# A Taylor Series Approach to Correct Localization Errors in Robotic Field Mapping using Gaussian Processes

Muzaffar Qureshi[1] Tochukwu Elijah Ogri[1] Kyle Volle[2] Rushikesh Kamalapurkar[1]

***Abstract*— Gaussian Processes (GPs) are powerful non-parametric Bayesian models for scalar-field regression, typically assuming perfectly known measurement locations and Gaussian measurement noise. In many real-world mapping applications, however, sensor-equipped mobile robots collect measurements under imperfect localization, causing discrepancies between estimated and true measurement locations that degrade GP mean and covariance estimates. To address this issue, we propose a method for updating GP models as improved location estimates become available. By exploiting the differentiability of the GP kernel, we develop a second-order correction algorithm based on precomputed Jacobians and Hessians of the GP mean and covariance, enabling real-time refinement from measurement-location discrepancy data. Simulation results show improved prediction accuracy and lower computational cost than full retraining.**

## I. Introduction

Mapping unknown fields is important in many scientific and engineering applications, including environmental monitoring, search and rescue, and autonomous underwater exploration [1]. Among available methods, Gaussian Processes (GPs) are widely used because they capture complex spatial correlations and provide uncertainty quantification [2]. Standard GP regression assumes known measurement locations and Gaussian measurement noise.

In many mapping applications, a sensor-equipped robot navigates the environment and collects field measurements to construct a map [3]. The true location of the robot is often unknown and needs to be estimated to ensure accurate mapping [4], [5]. Due to unmodeled dynamics and environmental disturbances, location estimates of the robot become corrupted, resulting in biased and noisy datasets for GP training [2], [6], [7]. Robots often rely on global positioning system (GPS) for localization during measurement collection. However, GPS signals can be unavailable or unreliable in environments with occlusions, interference, or other challenging conditions [8]–[10].

To model an unknown field, Girard [11] developed a GP framework for propagating input uncertainty by modeling the input as a Gaussian random variable and exploiting the analytical structure of the squared exponential (SE) kernel to obtain closed-form expressions for the predictive mean and variance. This approach is well-suited to discrete-time nonlinear models [12] and iterative $k$-step-ahead forecasting [11], where uncertainty accumulates over time. Although the SE kernel admits closed-form derivatives, the framework in [11], [12] is restricted to the SE kernel and assumes Gaussian uncertainty. When these assumptions are violated, one must rely on computationally expensive approximations, which further complicate hyperparameter estimation, particularly in high-dimensional settings.

As an alternative, McHutchon and Rasmussen introduced the noisy input Gaussian process (NIGP) model, which provides an approximate analytical treatment of input uncertainty [13]. NIGP uses a first-order Taylor expansion at each noisy input to recast input uncertainty as an additive output-variance term proportional to the squared gradient of the posterior mean, without requiring a specific kernel. Its hyperparameters, including the input noise variances, are learned jointly by maximizing the marginal likelihood. However, the first-order approximation may lose accuracy for strongly nonlinear functions [13], while the added hyperparameters increase training complexity and the assumption of constant input noise variance limits applicability to heteroscedastic settings.

Most existing methods assume that measurement-location errors are independent and identically distributed (i.i.d.) with Gaussian distributions and compensate for Gaussian uncertainty by modifying the covariance function or introducing a corrective variance term. In many practical mapping applications, however, location errors are dominated by systematic deterministic effects rather than random noise.

Correcting deterministic input errors in GP-based maps without full retraining is challenging because the GP mean and covariance depend on the inverse kernel matrix, so updating a measurement location generally costs $O(T^3)$, where $T$ is the number of measurements. To overcome this, this paper proposes a two-stage correction framework: an offline stage that precomputes the Jacobians and Hessians of the GP mean and covariance, and an online stage that uses them to apply real-time corrections as improved location estimates become available. The main contribution is a gradient-based method for correcting deterministic measurement-location errors in GP models.

## II. Problem Formulation

Consider a sensor-equipped robot operating in a domain $\mathcal{X} \subset \mathbb{R}^n$ that is tasked with visiting a set of measurement locations to create a map of a scalar field $f : \mathcal{X} \to \mathbb{R}$. The

This research was supported in part by the Air Force Research Laboratories under contract numbers FA8651-25-1-0019. Any opinions, findings, or recommendations in this article are those of the author(s), and do not necessarily reflect the views of the sponsoring agencies.

[1] Dept. of Mech. and Aero. Engg, University of Florida, Gainesville, Florida, USA, email: {muzaffar.qureshi, tochukwu.ogri, rkamalapurkar} @ufl.edu.

[2] Torch Technologies, Shalimar, Florida, USA, email: Kyle.Volle@torchtechnologies.com

field can be modeled by GP as

$$\hat{f} \sim \mathrm{GP}(\mu, \Sigma), \tag{1}$$

where $\mu : \mathbb{R}^n \to \mathbb{R}$ is the mean function and $\Sigma : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ is the covariance function. The robot is programmed to follow a planned trajectory, visiting planned measurement locations $\hat{\mathbf{X}} = \{\hat{\mathbf{x}}_i\}_{i=1}^T$, where $T \in \mathbb{N}$ is the total number of measurements. However, due to unmodeled dynamics or localization errors, the robot deviates from its planned trajectory and collects measurements at the actual locations $\mathbf{X} = \{\mathbf{x}_i\}_{i=1}^T$. The corresponding measurement vector is $\mathbf{Y} = [y_1, \ldots, y_T]^\top \in \mathbb{R}^{T\times 1}$, where $y_i = f(\mathbf{x}_i) + \epsilon_i$, and $\epsilon_i \sim \mathcal{N}(0, \sigma_y^2)$ is zero-mean Gaussian measurement noise.

The robot ignores localization error during training and incorrectly pairs the measurements $\mathbf{Y}$ with the planned locations $\hat{\mathbf{X}}$. As a result, the corrupted GP model is trained on the mismatched dataset $\{\hat{\mathbf{X}}, \mathbf{Y}\}$. The true and planned measurement locations satisfy

$$\mathbf{x}_i = \hat{\mathbf{x}}_i + \delta_i, \quad \forall i = 1, 2, \ldots, T, \tag{2}$$

where $\delta_i \in \mathbb{R}^n$ is the localization error for the $i$-th measurement.

The objective is to use the known errors $\delta_i$ to correct the corrupted GP model so that it approximates the predictive performance of an ideal GP trained on $\{\mathbf{X}, \mathbf{Y}\}$, without full retraining or additional measurements. The following assumptions are made to keep the problem tractable.

*Assumption 1:* $\epsilon_i$ is i.i.d. zero-mean Gaussian with variance $\sigma_y^2$, i.e., $\epsilon_i \sim \mathcal{N}(0, \sigma_y^2)$, and $\mathbb{E}[\epsilon_i \epsilon_j] = 0$ for $i \neq j$.

*Assumption 2:* The localization error satisfies $\|\delta_i\| \leq \delta_{\max}$ for all $i = 1, 2, \ldots, T$.

## III. Gaussian Process Regression

The unknown function $f$ is modeled as a GP with squared exponential (SE) kernel [2],

$$k(\hat{\mathbf{x}}, \hat{\mathbf{x}}') \coloneqq \alpha^2 \exp\left(-\frac{\|\hat{\mathbf{x}} - \hat{\mathbf{x}}'\|^2}{2\beta^2}\right), \tag{3}$$

where $\alpha^2$ and $\beta$ denote the signal variance and characteristic lengthscale, respectively, and $\hat{\mathbf{x}}, \hat{\mathbf{x}}' \in \mathcal{X}$. The SE kernel is selected due to its smoothness properties [2] and its universal approximation property [14]. The algorithm developed in this paper is compatible with other smooth kernels, such as Matérn-type kernels.

Let $\mathbf{X}_e = \{\mathbf{x}_{e,1}, \ldots, \mathbf{x}_{e,M}\} \subset \mathcal{X}$ denote the test locations for GP posterior evaluation. Because these locations are fixed, the test–test kernel matrix $K_{e-e} \in \mathbb{R}^{M\times M}$ is constant and given by

$$K_{e-e}[i, j] = k(\mathbf{x}_{e,i}, \mathbf{x}_{e,j}). \tag{4}$$

For an arbitrary set of $T$ training locations $\mathbf{Z} = \{\mathbf{z}_t\}_{t=1}^T \in \mathcal{X}^T$, where $\mathcal{X}^T \coloneqq \mathcal{X} \times \cdots \times \mathcal{X}$, define

$$K_{e-\mathrm{T}}(\mathbf{Z})[i, k] \coloneqq k(\mathbf{x}_{e,i}, \mathbf{z}_k), \tag{5}$$

$$K_{\mathrm{T}-\mathrm{T}}(\mathbf{Z})[j, k] \coloneqq k(\mathbf{z}_j, \mathbf{z}_k), \tag{6}$$

where $K_{e-\mathrm{T}} : \mathcal{X}^T \to \mathbb{R}^{M\times T}$ and $K_{\mathrm{T}-\mathrm{T}} : \mathcal{X}^T \to \mathbb{R}^{T\times T}$. For any $\mathbf{Z}$, these mappings produce the corresponding test–train and train–train kernel matrices.

The GP mean and covariance mappings $\mathbf{m} : \mathcal{X}^T \to \mathbb{R}^M$ and $\mathbf{S} : \mathcal{X}^T \to \mathbb{R}^{M\times M}$ are computed as

$$\mathbf{m}(\mathbf{Z}) \coloneqq K_{e-\mathrm{T}}(\mathbf{Z}) \left(K_{\mathrm{T}-\mathrm{T}}(\mathbf{Z}) + \sigma_y^2 \mathbf{I}_T\right)^{-1} \mathbf{Y}, \tag{7}$$

$$\mathbf{S}(\mathbf{Z}) \coloneqq K_{e-e} - K_{e-\mathrm{T}}(\mathbf{Z}) \left(K_{\mathrm{T}-\mathrm{T}}(\mathbf{Z}) + \sigma_y^2 \mathbf{I}_T\right)^{-1} K_{e-\mathrm{T}}(\mathbf{Z})^\top, \tag{8}$$

where $\mathbf{m}(\mathbf{Z}) \in \mathbb{R}^M$ and $\mathbf{S}(\mathbf{Z}) \in \mathbb{R}^{M\times M}$ denote the predictive mean and covariance, respectively. Evaluating these mappings at the planned measurement locations $\hat{\mathbf{X}}$ gives the corrupted GP posterior

$$\hat{\mathcal{M}} \coloneqq K_{e-\mathrm{T}}(\hat{\mathbf{X}}) \left(K_{\mathrm{T}-\mathrm{T}}(\hat{\mathbf{X}}) + \sigma_y^2 \mathbf{I}_T\right)^{-1} \mathbf{Y}, \tag{9}$$

$$\hat{\mathcal{S}} \coloneqq K_{e-e} - K_{e-\mathrm{T}}(\hat{\mathbf{X}}) \left(K_{\mathrm{T}-\mathrm{T}}(\hat{\mathbf{X}}) + \sigma_y^2 \mathbf{I}_T\right)^{-1} K_{e-\mathrm{T}}(\hat{\mathbf{X}})^\top. \tag{10}$$

A sensitivity analysis is then performed with respect to perturbations in the training locations. Specifically, a second-order Taylor expansion about $\hat{\mathbf{X}}$ is derived to approximate the corrected GP posterior under localization error.

## IV. Taylor Series Approximation of Mean and Covariance functions

In GP regression, the mean function $\mathbf{m}$ is formed as a linear combination of the observed measurements $\mathbf{Y}$, with weights determined by the correlation between the test and training locations. These weights depend on the test–train kernel matrix $K_{e-\mathrm{T}}$ in (5) and the inverse of the train–train kernel matrix $K_{\mathrm{T}-\mathrm{T}}$ in (6), both of which make the predictive mean sensitive to perturbations in the measurement locations. Similarly, the covariance $\mathbf{S}$, which quantifies predictive uncertainty, depends on $K_{e-e}$, $K_{e-\mathrm{T}}$, and $K_{\mathrm{T}-\mathrm{T}}$, and therefore also varies with changes in the correlation structure induced by the measurement locations.

To quantify the effect of the discrepancy between the true measurement locations $\mathbf{X}$ and the planned locations $\hat{\mathbf{X}}$ on the GP posterior, a second-order Taylor expansion is taken about $\hat{\mathbf{X}}$. Accordingly, the corrected mean vector $\mathcal{M} \coloneqq \mathbf{m}(\mathbf{X}) \in \mathbb{R}^{M\times 1}$ is approximated as

$$\mathcal{M} \approx \hat{\mathcal{M}} + \sum_{i=1}^{T} \mathbf{J}_M^i \delta_i + \frac{1}{2} \sum_{i=1}^{T} \sum_{j=1}^{T} \begin{bmatrix} \delta_i^\top \mathbf{H}_M^{i,j}[1, \cdot, \cdot] \delta_j \\ \vdots \\ \delta_i^\top \mathbf{H}_M^{i,j}[M, \cdot, \cdot] \delta_j \end{bmatrix}, \tag{11}$$

where $\delta_i \in \mathbb{R}^{n\times 1}$ is the error at location $\hat{\mathbf{x}}_i$. The Jacobian and Hessian matrices are defined as $\mathbf{J}_M^i \coloneqq \frac{\partial \mathbf{m}}{\partial \mathbf{z}_i}\Big|_{\mathbf{Z}=\hat{\mathbf{X}}} \in \mathbb{R}^{M\times n}$, and $\mathbf{H}_M^{i,j} \coloneqq \frac{\partial^2 \mathbf{m}}{\partial \mathbf{z}_i \partial \mathbf{z}_j^\top}\Big|_{\mathbf{Z}=\hat{\mathbf{X}}} \in \mathbb{R}^{M\times n\times n}$, respectively. Here, $\mathbf{H}_M^{i,j}[t, :, :] \in \mathbb{R}^{n\times n}$ denotes the Hessian matrix of the $t$-th component of $\mathbf{m}$ with respect to $\mathbf{z}_i$ and $\mathbf{z}_j$.

Similarly, the corrected covariance matrix $\mathcal{S} \coloneqq \mathbf{S}(\mathbf{X}) \in \mathbb{R}^{M\times M}$ is approximated using a second-order Taylor expansion around $\hat{\mathbf{X}}$ as

$$\mathcal{S} \coloneqq \hat{\mathcal{S}} + \sum_{i=1}^{T} \mathbf{J}_{\mathcal{S}}^{i} \delta_i + \frac{1}{2} \sum_{i=1}^{T} \sum_{j=1}^{T} \begin{bmatrix} \delta_i^\top \mathbf{H}_{\mathcal{S}}^{i,j}[1,1,:,:]\delta_j & \cdots & \delta_i^\top \mathbf{H}_{\mathcal{S}}^{i,j}[1,M,:,:]\delta_j \\ \vdots & \ddots & \vdots \\ \delta_i^\top \mathbf{H}_{\mathcal{S}}^{i,j}[M,1,:,:]\delta_j & \cdots & \delta_i^\top \mathbf{H}_{\mathcal{S}}^{i,j}[M,M,:,:]\delta_j \end{bmatrix}, \tag{12}$$

where $\mathbf{J}_{\mathcal{S}}^{i} \coloneqq \frac{\partial \mathbf{S}}{\partial \mathbf{z}_i}\Big|_{\mathbf{Z}=\hat{\mathbf{X}}} \in \mathbb{R}^{M\times M\times n}$ denotes the Jacobian and $\mathbf{H}_{\mathcal{S}}^{i,j} \coloneqq \frac{\partial^2 \mathbf{S}}{\partial \mathbf{z}_i \partial \mathbf{z}_j^\top}\Big|_{\mathbf{Z}=\hat{\mathbf{X}}} \in \mathbb{R}^{M\times M\times n\times n}$ denotes the Hessian tensor of the covariance function.

Although the corrected mean and covariance expressions in (11) and (12) involve first- and second-order derivatives with respect to every measurement location, their computational cost is mitigated by substantial sparsity in the Jacobian and Hessian tensors. As shown in the next section, this sparsity follows from the kernel structure. In particular, differentiating the kernel with respect to a specific measurement location $\mathbf{z}_k$ affects only the $k$-th column of $K_{e-\mathrm{T}}$ and the $k$-th row and column of $K_{\mathrm{T}-\mathrm{T}}$.

### A. Gradient of the Test-Train Kernel Matrix

To compute the gradient of the kernel matrix $K_{e-\mathrm{T}}$, first consider the gradient of the kernel function $k(\mathbf{x},\mathbf{z})$ with respect to its second argument $\mathbf{z}$. For the SE kernel in (3), the gradient, $\frac{\partial k(\mathbf{x},\mathbf{z})}{\partial \mathbf{z}} \in \mathbb{R}^n$, is computed as $\frac{\partial k(\mathbf{x},\mathbf{z})}{\partial \mathbf{z}} = \frac{1}{\beta^2}(\mathbf{x}-\mathbf{z})\,k(\mathbf{x},\mathbf{z})$.

The derivative of the test–train kernel matrix $K_{e-\mathrm{T}}$ with respect to a single training input $\mathbf{z}_k$ is a tensor in $\mathbb{R}^{M\times T\times n}$. Because only the $k$-th column of $K_{e-\mathrm{T}}$ depends on $\mathbf{z}_k$, the tensor is sparse, and computed as

$$\left[\frac{\partial K_{e-\mathrm{T}}}{\partial \mathbf{z}_k}\right]_{i,j,:}\Bigg|_{\mathbf{Z}=\hat{\mathbf{X}}} = \begin{cases} \left(\frac{\partial k(\mathbf{x}_{e,i},\mathbf{z})}{\partial \mathbf{z}}\big|_{\mathbf{z}=\hat{\mathbf{x}}_k}\right)^\top, & j=k, \\ \mathbf{0}_n^\top, & \text{otherwise,} \end{cases}$$

where $[\cdot]_{i,j,:}$ denotes the $1\times n$ vector at position $(i,j)$.

### B. Gradient of the Training Kernel Matrix

For the training kernel matrix $K_{\mathrm{T}-\mathrm{T}} \in \mathbb{R}^{T\times T}$, differentiation with respect to the training location $\mathbf{z}_k \in \mathbb{R}^n$ affects only the $k$-th row and $k$-th column. Thus, the gradient tensor $\frac{\partial K_{\mathrm{T}-\mathrm{T}}}{\partial \mathbf{z}_k}\Big|_{\mathbf{Z}=\hat{\mathbf{X}}} \in \mathbb{R}^{T\times T\times n}$ is sparse, with entries

$$\left[\frac{\partial K_{\mathrm{T}\text{-}\mathrm{T}}}{\partial \mathbf{z}_k}\right]_{j,l,:}\Bigg|_{\mathbf{Z}=\hat{\mathbf{X}}} = \begin{cases} \left(\frac{\partial k(\hat{\mathbf{x}}_j,\mathbf{z})}{\partial \mathbf{z}}\big|_{\mathbf{z}=\hat{\mathbf{x}}_k}\right)^\top, & l=k,\ j\neq k, \\ \left(\frac{\partial k(\mathbf{x},\hat{\mathbf{x}}_l)}{\partial \mathbf{x}}\big|_{\mathbf{x}=\hat{\mathbf{x}}_k}\right)^\top, & j=k,\ l\neq k, \\ \mathbf{0}_n^\top, & \text{otherwise.} \end{cases}$$

## V. Gradient of the GP Mean and Covariance

The Jacobians of the GP mean function $\mathbf{m}$ and covariance function $\mathbf{S}$ with respect to the training location $\mathbf{z}_i$, evaluated at $\hat{\mathbf{X}}$, define the terms $\mathbf{J}_M^i$ and $\mathbf{J}_{\mathcal{S}}^i$ in the Taylor expansions (11) and (12). These Jacobians simplify considerably because the gradients of the kernel matrices are sparse, with nonzero entries only in the $i$-th row or column.

### A. Gradient of the GP Mean Function

The Jacobian of $\mathbf{m}$ as defined in (7) with respect to the $i$-th training location $\mathbf{z}_i$, evaluated at $\mathbf{Z}=\hat{\mathbf{X}}$, is $\mathbf{J}_M^i \in \mathbb{R}^{M\times n}$. Let $\mathbf{K}(\mathbf{Z}) = \mathbf{K}_{\mathrm{T}-\mathrm{T}}(\mathbf{Z}) + \sigma_y^2 \mathbf{I}_{\mathrm{T}}$, and define $\mathbf{K} \coloneqq \mathbf{K}(\hat{\mathbf{X}})$ and $\mathbf{K}_{e-\mathrm{T}} \coloneqq \mathbf{K}_{e-\mathrm{T}}(\hat{\mathbf{X}})$ for brevity.

Applying the chain rule and the matrix inverse identity, $\frac{\partial \mathbf{K}(\mathbf{Z})^{-1}}{\partial \mathbf{z}_i} = -\mathbf{K}(\mathbf{Z})^{-1}\frac{\partial \mathbf{K}(\mathbf{Z})}{\partial \mathbf{z}_i}\mathbf{K}(\mathbf{Z})^{-1}$ yields

$$\mathbf{J}_M^i = \left(\frac{\partial \mathbf{K}_{e-\mathrm{T}}}{\partial \mathbf{z}_i}\Big|_{\hat{\mathbf{X}}}\right)\mathbf{K}^{-1}\mathbf{Y} - \mathbf{K}_{e-\mathrm{T}}\mathbf{K}^{-1}\left(\frac{\partial \mathbf{K}_{\mathrm{T}-\mathrm{T}}}{\partial \mathbf{z}_i}\Big|_{\hat{\mathbf{X}}}\right)\mathbf{K}^{-1}\mathbf{Y}.$$

Let $\mathbf{c} \coloneqq \mathbf{K}^{-1}\mathbf{Y} \in \mathbb{R}^{T\times 1}$ and $\mathbf{P} \coloneqq \mathbf{K}_{e-\mathrm{T}}\mathbf{K}^{-1} \in \mathbb{R}^{M\times T}$. Let $\mathbf{J}_{e-T}^i \coloneqq \frac{\partial \mathbf{K}_{e-\mathrm{T}}}{\partial \mathbf{z}_i}\big|_{\hat{\mathbf{X}}} \in \mathbb{R}^{M\times T\times n}$ and $\mathbf{J}_{T-T}^i \coloneqq \frac{\partial \mathbf{K}_{\mathrm{T}-\mathrm{T}}}{\partial \mathbf{z}_i}\big|_{\hat{\mathbf{X}}} \in \mathbb{R}^{T\times T\times n}$. Using the sparsity of $\mathbf{J}_{e-T}^i$ (non-zero only at column $i$), the $\mathbf{J}_M^i$ Jacobian simplifies to

$$\mathbf{J}_M^i = \mathbf{J}_{e-T}^i[\cdot,i,\cdot]\mathbf{c}[i] - \mathbf{P}\mathbf{J}_{T-T}^i\mathbf{c}, \tag{13}$$

where $\mathbf{J}_{e-T}^i[\cdot,i,\cdot] \in \mathbb{R}^{M\times n}$ is the $i$-th slice of the kernel gradient tensor, $\mathbf{c}[i]$ is the $i$-th element of $\mathbf{c}$, and the second term denotes a full tensor contraction resulting in an $M\times n$ matrix.

### B. Gradient of the GP Covariance Function

The Jacobian of $\mathbf{S}$ as defined in (8) with respect to $\mathbf{z}_i$, evaluated at $\mathbf{Z}=\hat{\mathbf{X}}$, is $\mathbf{J}_{\mathcal{S}}^i \in \mathbb{R}^{M\times M\times n}$. Since $K_{e-e}$ is constant, its gradient is zero. Let $\mathbf{A}(\mathbf{Z}) = \mathbf{K}_{e-\mathrm{T}}(\mathbf{Z}) \in \mathbb{R}^{M\times T}$ and $\mathbf{A} \coloneqq \mathbf{A}(\hat{\mathbf{X}})$. Applying the product rule for matrix gradients yields

$$\mathbf{J}_{\mathcal{S}}^i = -\left(\frac{\partial \mathbf{A}}{\partial \mathbf{z}_i}\right)\mathbf{K}^{-1}\mathbf{A}^\top - \mathbf{A}\left(\frac{\partial \mathbf{K}^{-1}}{\partial \mathbf{z}_i}\right)\mathbf{A}^\top - \mathbf{A}\mathbf{K}^{-1}\left(\frac{\partial \mathbf{A}^\top}{\partial \mathbf{z}_i}\right).$$

Using the identity $\frac{\partial \mathbf{K}^{-1}}{\partial \mathbf{z}_i} = -\mathbf{K}^{-1}\frac{\partial \mathbf{K}}{\partial \mathbf{z}_i}\mathbf{K}^{-1}$, and noting that the first and third terms are transposes of each other, the Jacobian $\mathbf{J}_{\mathcal{S}}^i$ is simplified. Let $\mathbf{P} \coloneqq \mathbf{A}\mathbf{K}^{-1} \in \mathbb{R}^{M\times T}$, $\mathbf{J}_{\mathbf{A}}^i \coloneqq \frac{\partial \mathbf{A}}{\partial \mathbf{z}_i}\big|_{\hat{\mathbf{X}}} \in \mathbb{R}^{M\times T\times n}$ and $\mathbf{J}_{\mathbf{K}}^i \coloneqq \frac{\partial \mathbf{K}}{\partial \mathbf{z}_i}\big|_{\hat{\mathbf{X}}} \in \mathbb{R}^{T\times T\times n}$.

$$\mathbf{J}_{\mathcal{S}}^i = -\left(\mathbf{J}_{\mathbf{A}}^i\mathbf{K}^{-1}\mathbf{A}^\top\right) - \left(\mathbf{J}_{\mathbf{A}}^i\mathbf{K}^{-1}\mathbf{A}^\top\right)^\top + \mathbf{P}\mathbf{J}_{\mathbf{K}}^i\mathbf{P}^\top, \tag{14}$$

where $\mathbf{J}\mathbf{A}^i$ and $\mathbf{J}\mathbf{K}^i$ denote the kernel gradient tensors, which are sparse and have non-zero entries only in column $i$ for $\mathbf{J}\mathbf{A}^i$ and in row and column $i$ for $\mathbf{J}\mathbf{K}^i$. The term $\mathbf{P}\mathbf{J}_{\mathbf{K}}^i\mathbf{P}^\top$ represents a full tensor contraction, yielding an output tensor of dimension $M\times M\times n$.

## VI. Offline Computation of GP Gradients for Efficient Online Updates

The structure of the GP moments reveals a critical property: the dependence on the measurement vector $\mathbf{Y}$ is separable from the dependence on the planned measurement locations $\hat{\mathbf{X}}$. This separation is key to avoiding the $\mathcal{O}(T^3)$ computational bottleneck of a full GP retraining when the measurement data set is corrected. The mean function $\mathbf{m}(\mathbf{Z})$, and in particular its evaluation $\hat{\mathcal{M}} = K_{e-\mathrm{T}}K^{-1}\mathbf{Y}$, is linear in the measurement vector $\mathbf{Y}$. Therefore, all derivatives of the mean with respect to the training locations, including the Jacobian $\mathbf{J}_M^i$ and Hessian $\mathbf{H}_M^{i,j}$, are also linear in $\mathbf{Y}$.

The GP covariance in (8) is independent of $\mathbf{Y}$. Therefore, its derivatives, including $\mathbf{J}_{\mathcal{S}}^i$ and $\mathbf{H}_{\mathcal{S}}^{i,j}$, depend only

on the training locations $\hat{\mathbf{X}}$, test locations $\mathbf{X}_e$, and the kernel hyperparameters. This observation motivates a two-stage framework. In the offline stage, the expensive location-dependent parts of the Jacobians and Hessians are precomputed using automatic differentiation. In the online stage, these precomputed tensors are combined with $\mathbf{Y}$ to assemble the required derivatives, enabling real-time GP correction.

### A. Mean Function Update

The Jacobian $\mathbf{J}_M^i \in \mathbb{R}^{M\times n}$ can be expressed as a tensor contraction $\mathbf{J}_M^i = \mathbf{F}^i(\hat{\mathbf{X}}, \mathbf{X}_e)\mathbf{Y}$, where $\mathbf{F}^i \in \mathbb{R}^{M\times n\times T}$ is the $\mathbf{Y}$-independent structural tensor that is precomputed offline. Similarly, the Hessian $\mathbf{H}_M^{i,j} \in \mathbb{R}^{M\times n\times n}$ is given by $\mathbf{H}_M^{i,j} = \mathbf{G}^{i,j}(\hat{\mathbf{X}}, \mathbf{X}_e)\mathbf{Y}$, where $\mathbf{G}^{i,j} \in \mathbb{R}^{M\times n\times n\times T}$ is the corresponding precomputed fourth-order tensor.

By substituting these tensors into the original Taylor series expansion in (11) and (12), we get

$$\mathcal{M} \coloneqq \hat{\mathcal{M}} + \sum_{i=1}^{T} \left(\mathbf{F}^i\mathbf{Y}\right)\delta_i + \frac{1}{2}\sum_{i=1}^{T}\sum_{j=1}^{T} \begin{bmatrix} \delta_i^\top \left(\mathbf{G}^{i,j}\mathbf{Y}\right)[1,\cdot,\cdot]\delta_j \\ \vdots \\ \delta_i^\top \left(\mathbf{G}^{i,j}\mathbf{Y}\right)[M,\cdot,\cdot]\delta_j \end{bmatrix}.$$

As a result, the online computation only involves tensor contractions between the precomputed structural components ($\mathbf{F}^i$, $\mathbf{G}^{i,j}$), $\mathbf{Y}$ and $\{\delta_i\}_{i=1}^T$.

### B. Covariance Function Update

An analogous correction is applied to the covariance function $\mathbf{S}$. Its value and derivatives, including the Jacobian $\mathbf{J}_{\mathcal{S}}^i \in \mathbb{R}^{M\times M\times n}$ and cross-Hessian $\mathbf{H}_{\mathcal{S}}^{i,j} \in \mathbb{R}^{M\times M\times n\times n}$, are independent of the measurement vector $\mathbf{Y}$. Instead, these tensors depend only on the planned measurement locations $\hat{\mathbf{X}}$, the test locations $\mathbf{X}_e$, and the kernel hyperparameters, and can therefore be precomputed offline.

The corrected covariance matrix $\mathcal{S}$ is then obtained online by combining these precomputed tensors with the localization errors $\{\delta_i\}_{i=1}^T$. Using a second-order Taylor expansion,

$$\mathcal{S} \coloneqq \hat{\mathcal{S}} + \sum_{i=1}^{T} \mathbf{J}_{\mathcal{S}}^i\delta_i + \frac{1}{2}\sum_{i=1}^{T}\sum_{j=1}^{T}\left(\delta_i^\top \mathbf{H}_{\mathcal{S}}^{i,j}\delta_j\right), \tag{15}$$

where $\hat{\mathcal{S}}$ is the covariance computed using the planned locations $\hat{\mathbf{X}}$. The term $\mathbf{P}\mathbf{J}_{\mathbf{K}}^i\mathbf{P}^\top$ denotes a full tensor contraction, resulting in an output tensor of dimension $M\times M\times n$, where the terms $\mathbf{J}_{\mathcal{S}}^i\delta_i$ and $\delta_i^\top\mathbf{H}_{\mathcal{S}}^{i,j}\delta_j$ denote the tensor contractions required to produce the $M\times M$ correction matrices.

By precomputing all structural components $(\mathbf{F}^i, \mathbf{G}^{i,j}, \mathbf{J}_{\mathcal{S}}^i, \mathbf{H}_{\mathcal{S}}^{i,j})$, online evaluation of both $\mathcal{M}$ and $\mathcal{S}$ is reduced to a series of tensor products, bypassing full GP recomputation.

### C. Computational Complexity Analysis

Full GP retraining, used as the baseline, is dominated by inversion of the $T\times T$ covariance matrix and thus requires $\mathcal{O}(T^3)$ operations. The developed framework avoids this online cost by precomputing the Jacobian and Hessian tensors offline.

Online, the first- and second-order mean corrections require $\mathcal{O}(TMn)$ and $\mathcal{O}(T^2Mn^2)$ operations, respectively. The corresponding covariance corrections require $\mathcal{O}(TM^2n)$ and $\mathcal{O}(T^2M^2n^2)$. Therefore, the worst-case online complexity is $\mathcal{O}(T^2M^2n^2)$. Since the derivative tensors are highly sparse, the practical cost is often much lower.

*Remark 1:* This bound corresponds to correcting all $T$ measurement locations simultaneously. In practical settings such as robotic SLAM or loop closure, only a small subset $K \ll T$ is usually updated. In that case, the online complexity reduces to $\mathcal{O}(KM^2n + K^2M^2n^2)$, which is substantially lower than both full retraining and the worst-case $\mathcal{O}(T^2M^2n^2)$ cost.

**Algorithm 1** GP Correction under Location Errors

**Input**: Planned locations $\hat{\mathbf{X}} \in \mathbb{R}^{T\times n}$, Actual measurements $\mathbf{Y} \in \mathbb{R}^{T\times 1}$, Test locations $\mathbf{X}_e \in \mathbb{R}^{M\times n}$, GP hyperparameters $\alpha, \beta$
**Offline Phase**:
  Compute and store gradient tensors
  Compute initial kernel matrices: $K_{\mathrm{T-T}}$, $K_{e-\mathrm{T}}$, $K_{e-e}$
  Compute gradient operators: $\{\mathbf{F}^i\}_{i=1}^T$, $\{\mathbf{G}^{i,j}\}_{i,j=1}^T$, $\{\mathbf{J}_{\mathcal{S}}^i\}_{i=1}^T$, $\{\mathbf{H}_{\mathcal{S}}^{i,j}\}_{i,j=1}^T$
**Online Phase**:
**Input**: Full set of location perturbations $\{\delta_i\}_{i=1}^T$
  Compute initial GP moments: $\hat{\mathcal{M}}$, $\hat{\mathcal{S}}$
  Instantiate Jacobians/Hessians using $\mathbf{F}^i$, $\mathbf{G}^{i,j}$ and current $\mathbf{Y}$.
  $\mathbf{J}_M^i \leftarrow \mathbf{F}^i\mathbf{Y}$ (for all $i = 1,\ldots,T$)
  $\mathbf{H}_M^{i,j} \leftarrow \mathbf{G}^{i,j}\mathbf{Y}$ (for all $i,j = 1,\ldots,T$)
  Update $\mathcal{M}$ using $\mathbf{J}_M^i, \mathbf{H}_M^{i,j}$ and $\{\delta_i\}_{i=1}^T$ (Eq. 11)
  Update $\mathcal{S}$ using $\mathbf{J}_{\mathcal{S}}^i, \mathbf{H}_{\mathcal{S}}^{i,j}$ and $\{\delta_i\}_{i=1}^T$ (Eq. 12)
**Output**: $\mathcal{M}$, $\mathcal{S}$

## VII. Convergence Analysis

In the preceding sections, the GP mean $\mathbf{m}$ was defined as a function of the training locations $\mathbf{Z} = \{\mathbf{z}_i\}_{i=1}^T$. For the convergence analysis, $\mathbf{m}$ is viewed instead as a function of the stacked vector $\boldsymbol{z} = \mathrm{vec}(\mathbf{z}_1,\ldots,\mathbf{z}_T) \in \mathbb{R}^{nT}$, with admissible training locations in a compact set $\mathcal{D} \subset \mathbb{R}^{nT}$. Likewise, define the stacked planned and actual locations as $\hat{\boldsymbol{x}} = \mathrm{vec}(\hat{\mathbf{x}}_1,\ldots,\hat{\mathbf{x}}_T) \in \mathbb{R}^{nT}$ and $\boldsymbol{x} = \mathrm{vec}(\mathbf{x}_1,\ldots,\mathbf{x}_T) \in \mathbb{R}^{nT}$, respectively. The resulting perturbation is $\boldsymbol{\delta} \in \mathbb{R}^{nT}$.

*Theorem 1:* Given the GP mean function $\mathbf{m} : \mathcal{D} \to \mathbb{R}^M$, if the kernel function $k$ is analytic on $\mathcal{X} \subset \mathbb{R}^n$, then $\mathbf{m}$ is analytic on $\mathcal{D}$. Hence, the true mean value $\mathcal{M} \coloneqq \mathbf{m}(\boldsymbol{x})$ admits the multivariate Taylor expansion about $\hat{\boldsymbol{x}}$, with perturbation $\boldsymbol{\delta} = \boldsymbol{x} - \hat{\boldsymbol{x}}$, given by $\mathcal{M} = \sum_{N=0}^{\infty}\frac{1}{N!}\nabla^N\mathbf{m}(\hat{\boldsymbol{x}})\,[\boldsymbol{\delta}^N]$, where $\nabla^N\mathbf{m}(\hat{\boldsymbol{x}})$ denotes the $N$-th order gradient tensor of $\mathbf{m}$ at $\hat{\boldsymbol{x}}$, and $[\boldsymbol{\delta}^N]$ denotes the $N$-fold tensor application of the perturbation vector $\boldsymbol{\delta}$. Furthermore, the Taylor series converges uniformly to $\mathbf{m}$ on any compact subset of $\mathcal{D}$.

*Proof:* The GP mean function $\mathbf{m}(\boldsymbol{z})$ is given by (7). Since $k$ is analytic on $\mathcal{X}$, each entry of the kernel matrices $\mathbf{K}_{e-\mathrm{T}}$ and $\mathbf{K}_{\mathrm{T-T}}$ is analytic in the components of $\boldsymbol{z}$. Matrix addition and multiplication preserve analyticity, and matrix inversion is analytic wherever the inverse exists. Therefore, the composite function $\mathbf{m}(\boldsymbol{z})$ is real analytic on $\mathcal{D}$.

By definition of real analyticity, $\mathbf{m}(\boldsymbol{z})$ admits a Taylor series expansion about any point $\hat{\boldsymbol{x}} \in \mathcal{D}$. Let $\mathcal{V} \subset \mathcal{D}$ be compact. Then the perturbation $\boldsymbol{\delta} = \boldsymbol{x} - \hat{\boldsymbol{x}}$ is uniformly bounded

on $\mathcal{V}$, that is, $\|\boldsymbol{x}-\hat{\boldsymbol{x}}\| \leq L_f$ for all $\boldsymbol{x} \in \mathcal{V}$. By the multivariate Taylor theorem, the remainder $R_N(\boldsymbol{x}) = \mathbf{m}(\boldsymbol{x}) - T_N(\boldsymbol{x})$, where $T_N$ is the $N$-th order Taylor polynomial, satisfies $\|R_N(\boldsymbol{x})\| \leq \frac{M_{N+1} L_f^{N+1}}{(N+1)!}$, where $M_{N+1} > 0$ bounds the $(N+1)$-th order gradient tensor on $\mathcal{V}$. Since $(N+1)!$ grows faster than $L_f^{N+1}$, it follows that $\|R_N(\boldsymbol{x})\| \to 0$ as $N \to \infty$ [15, Theorem 5.15]. ■

Theorem 1 shows that the mean function $\mathbf{m}$ is infinitely differentiable and admits a convergent Taylor series. Assumption 2 also yields a bound on the total perturbation vector $\boldsymbol{\delta} = \boldsymbol{x} - \hat{\boldsymbol{x}}$. In particular, if $\|\delta_i\| \leq \delta_{\max}$ for all $i = 1, \ldots, T$, then the $L_2$-norm of the stacked perturbation satisfies

$$\|\boldsymbol{\delta}\| \leq \sqrt{\sum_{i=1}^{T} (\delta_{\max})^2} = \delta_{\max}\sqrt{T}. \tag{16}$$

Building on Theorem 1 and the bound in (16), the following theorem establishes bounds on the number of higher-order gradients required to achieve a prescribed approximation accuracy on a compact subset of the input domain.

*Theorem 2:* Let $\mathcal{V} \subset \mathcal{D}$ be compact, and suppose $\|\boldsymbol{x} - \hat{\boldsymbol{x}}\| \leq \beta$ for all $\boldsymbol{x} \in \mathcal{V}$, where $\beta := \delta_{\max}\sqrt{T}$. Given a desired accuracy $\epsilon > 0$, the minimum order $N$ of the Taylor polynomial $T_N(\boldsymbol{x})$ required to ensure

$$\|\mathbf{m}(\boldsymbol{x}) - T_N(\boldsymbol{x})\| \leq \epsilon, \quad \forall \boldsymbol{x} \in \mathcal{V}, \tag{17}$$

is the smallest integer $N$ satisfying $\frac{M_{N+1}\beta^{N+1}}{(N+1)!} \leq \epsilon$, where $M_{N+1} \geq \sup_{\boldsymbol{\kappa}\in\mathcal{V}} \|\nabla^{N+1}\mathbf{m}(\boldsymbol{\kappa})\|$ bounds the $(N+1)$-th order gradient tensor.

*Proof:* From the proof of Theorem 1, the remainder $R_N(\boldsymbol{x}) = \mathbf{m}(\boldsymbol{x}) - T_N(\boldsymbol{x})$ satisfies $\|R_N(\boldsymbol{x})\| \leq \frac{M_{N+1}\|\boldsymbol{x}-\hat{\boldsymbol{x}}\|^{N+1}}{(N+1)!}$.

Using the bound $\|\boldsymbol{x}-\hat{\boldsymbol{x}}\| \leq \beta$, it follows that $\|R_N(\boldsymbol{x})\| \leq \frac{M_{N+1}\beta^{N+1}}{(N+1)!}$. Therefore, to guarantee $\|\mathbf{m}(\boldsymbol{x}) - T_N(\boldsymbol{x})\| \leq \epsilon$ for all $\boldsymbol{x} \in \mathcal{V}$, it suffices to choose $N$ as the smallest integer satisfying $\frac{M_{N+1}\beta^{N+1}}{(N+1)!} \leq \epsilon$. ■

## VIII. Simulation Results

Two simulations are used to validate the developed correction method for GP models trained on corrupted datasets. The first, in a 1D domain, introduces spatially varying perturbations in the measurement locations. The second, in a 2D domain, applies a constant offset $\delta$ to all locations, representing a uniform sensor bias.

### A. 1-Dimensional Example

Consider $f_1(x) = 2 + x\sin(6\pi x)$, with $x \in [0, 1]$. The planned measurement locations are $\hat{\mathbf{X}} = \{0, 0.1, 0.2, \ldots, 1\}$. An SE kernel is used with $\alpha = 1$ and $\beta = 0.1$. The Jacobians and Hessians in (13) and (14) are computed using CasADi [16]. Predictions are evaluated at $M = 100$ uniformly spaced query points. A baseline GP is trained on the original locations and measurements. The training locations are then corrupted by Gaussian noise $\epsilon \sim \mathcal{N}(0, 0.01^2)$. The corrupted and corrected GP models are shown in Figures 2 and 3, respectively. The error norm in Table I is

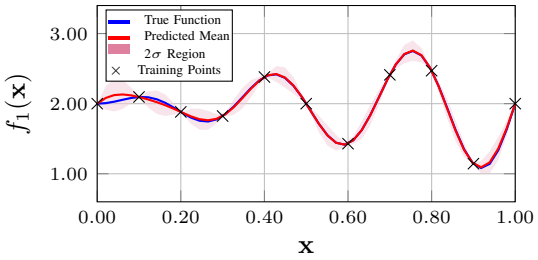


Fig. 1. Baseline GP model prediction trained on the original, non-corrupted 1D measurement locations. The true function is shown in blue.

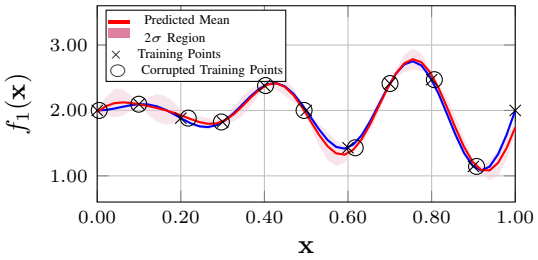


Fig. 2. GP model prediction trained on corrupted 1D measurement locations, demonstrating the model shift relative to the true function and original training points.

the $L_2$ prediction error over all query points $X_e$,$\|e\|_{\mathcal{M}} = \sqrt{\sum_{i=1}^{M} \left(f_1(x_i) - \mathcal{M}_i\right)^2}$, with $\|e\|_{\hat{\mathcal{M}}}$ defined similarly for the corrupted GP prediction $\hat{\mathcal{M}}$. Figure 4 compares the pointwise absolute errors of the corrupted and corrected GP models relative to the reference GP. Average error norms and computational times over 100 Monte Carlo trials are reported in Tables I and II, respectively.

### B. 2-Dimensional Example

The second simulation considers the scalar field $f_2(\mathbf{x}) = \sin(2\pi x)\cos(2\pi y)$, where $\mathbf{x} = (x, y) \in [0, 1]^2$.

To model a uniform sensor bias, the $x$-coordinates of all measurement locations are shifted by a constant offset $\delta_i = 0.1$, as illustrated in Figure 5. A GP model is then trained on these corrupted locations using the true field values. The corrected GP predictions are obtained using the gradient-based update. Figure 6 shows the error trajectories of the corrupted and corrected mean functions, demonstrating the reduction in error after correction.

TABLE I

Average Error Norm ($||e||$) from 100 Simulations.

| Improvement | 1D Sim | 2D Sim |
|---|---|---|
| $\|\|e\|\|_{\hat{\mathcal{M}}}$ | 0.65 | 2.04 |
| $\|\|e\|\|_{\mathcal{M}}$ | 0.20 | 0.50 |
| Improvement Percentage | 67.83 % | 75.38 % |

TABLE II

Average Computationsl Time for 100 Simulations.

| Method | Time (secs) 1D | Time (secs) 2D |
|---|---|---|
| Full Retraining | $1.16 \cdot 10^{-4}$ | $8.26 \cdot 10^{-4}$ |
| Gradient Correction | $3.17 \cdot 10^{-5}$ | $3.25 \cdot 10^{-5}$ |

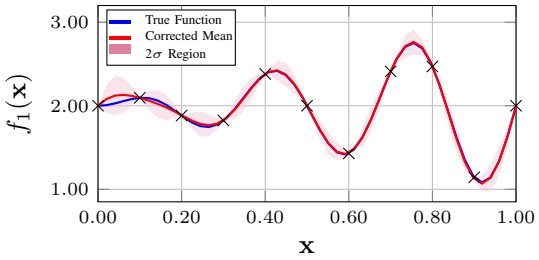


Fig. 3. Corrected GP prediction in the 1D simulation, showing the gradient-corrected mean (red) and $2\sigma$ confidence band with actual function (blue).

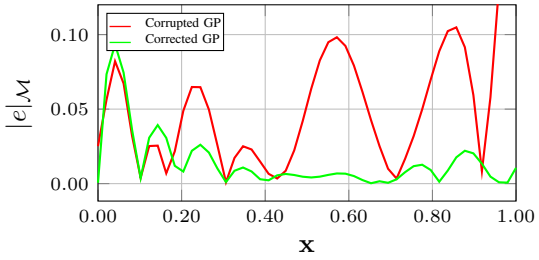


Fig. 4. Absolute error of the mean for the 1D simulation, comparing the corrupted GP against the corrected GP across the test domain.

## IX. Discussion

The developed framework corrects deterministic input errors in GP models without full retraining. By separating derivative computation from the online update, it enables real-time correction using precomputed Jacobian and Hessian tensors. As shown in Table II and Table I, this improves both computational efficiency and accuracy relative to GP models that ignore location error. While the results here use the SE kernel because of its infinite differentiability, the framework applies to any kernel that is at least twice differentiable. Its main limitation is memory scalability, since the stored derivative tensors grow with the number of training points and input dimensions. This motivates future work on compression and structure-exploiting approximations.

## X. Conclusion

A gradient-based correction framework has been developed for efficient real-time GP updates under deterministic input perturbations. By precomputing and storing derivative tensors offline, the method corrects the mean and covariance without recomputing the kernel inverse. This improves both accuracy and computational efficiency, making the framework well suited for robotic mapping, sensor fusion, and control in localization-uncertain or GPS-denied environments. Future work will address high-dimensional and large-scale settings using low-rank and sparsity-aware representations of the derivative tensors.

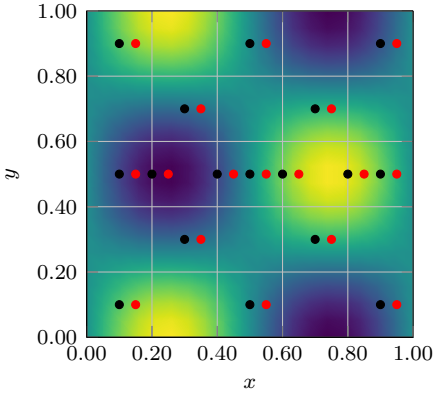


Fig. 5. Top-down view of the 2D scalar field, showing the true measurement locations (black dots) and the corrupted locations (red dots).

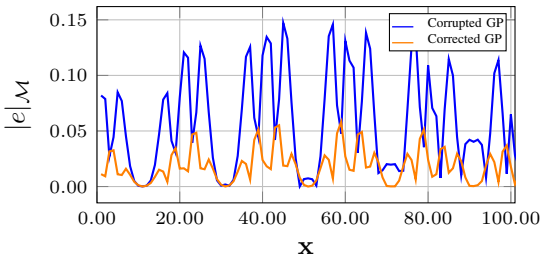


Fig. 6. Absolute mean error in the 2D simulation at 100 test points, comparing the corrupted and corrected GP predictions.

## References


[1] P. Sabella, "A rendering algorithm for visualizing 3D scalar fields," *Comput. Graph.*, vol. 22, no. 4, pp. 51–58, 1998.

[2] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. Cambridge, MA: MIT Press, 2006.

[3] M. Elhashash, H. Albanwan, and R. Qin, "A review of mobile mapping systems: From sensors to applications," *Sensors*, vol. 22, no. 11, 2022.

[4] H. Bai and C. N. Taylor, "Future uncertainty-based control for relative navigation in GPS-denied environments," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 56, no. 5, pp. 3491–3501, 2020.

[5] M. Qureshi, T. E. Ogri, H. Ramos, W. A. Makumi, Z. I. Bell, and R. Kamalapurkar, "A switched adaptive control approach to reduce sensing needs in trajectory tracking problems," *IEEE Control Syst. Lett.*, vol. 9, pp. 2777–2782, 2025.

[6] N. Uzzaman and H. Bai, "A novel variational Bayesian adaptive Kalman filter for systems with unknown state-dependent noise covariance matrices," in *Proc. Am. Control Conf.* IEEE, Jul. 2024, pp. 1192–1197.

[7] Y. Mao, T. Tan, X. Shen, W. E. Dixon, and R. Kamalapurkar, "Parallel OctoMapping: A scalable framework for enhanced path planning in autonomous navigation," arXiv:2603.22508, 2026.

[8] A. Bachrach, S. Prentice, R. He, and N. Roy, "RANGE-robust autonomous navigation in GPS-denied environments," *J. Field Robot.*, no. 5, pp. 644–666, 2011.

[9] M. Qureshi, T. E. Ogri, H. Ramos, Z. I. Bell, and R. Kamalapurkar, "Gaussian process-based scalar field estimation in GPS-denied environments," arXiv:2502.17584, 2025.

[10] M. Qureshi, T. E. Ogri, Z. I. Bell, and R. Kamalapurkar, "Scalar field mapping with adaptive high-intensity region avoidance," in *Proc. IEEE Conf. Control Technol. Appl.*, Aug. 2024, pp. 388–393.

[11] A. Girard, "Approximate methods for propagation of uncertainty with Gaussian process," Ph.D. dissertation, University of Glasgow (UK), 2004.

[12] J. Candela, A. Girard, J. Larsen, and C. Rasmussen, "Propagation of uncertainty in Bayesian kernel models - application to multiple-step ahead forecasting," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process*, vol. 2, 2003, pp. II–701.

[13] A. Mchutchon and C. Rasmussen, "Gaussian process training with input noise," in *In Proc. Adv. Neural Inf. Proc. Syst.*, vol. 24, 2011.

[14] J. Park and I. W. Sandberg, "Universal approximation using Radial-Basis-Function networks," *Neural Comput.*, vol. 3, no. 2, pp. 246–257, 1991.

[15] W. Rudin, *Principles of Mathematical Analysis*, 3rd ed. New York: McGraw-Hill, 1976.

[16] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, 2019.