Joint Coordinate Optimization in Fingerprint-Based Indoor Positioning

Author 1, Author 2, and Author 3

***Abstract*—Fingerprint-based indoor positioning estimates the users' locations in wireless local area network environments where satellite-based positioning methods cannot work properly. In this method, the location of a user is estimated by a pattern recognition algorithm (PRA). Traditionally, the training phase of PRA is conducted for x and y coordinates separately. However, the received signal strength from access points is a unique fingerprint for each measured point, not for x and y coordinates, independently. In this letter, we propose a novel PRA-based Gaussian process regression (GPR) method, named 2D-GPR, to jointly employ the x and y coordinates during the training phase. Experimental results show the superiority of 2D-GPR over conventional GPR (CGPR) and other competitors, especially in limited data samples. Also, the proposed method has a lower computation complexity compared with CGPR.**

***Index Terms*—wireless local area network, fingerprint-based positioning, machine learning, pattern recognition.**

I. Introduction

SATELLITE signals cannot penetrate in indoor environments, and due to the non-line-of-sight error, the accuracy of satellite-based positioning methods is not enough [1], [2]. The fingerprint-based wireless local area networks (WLANs) positioning is used in indoor environments to reach a high accuracy positioning [3]. In the training (offline) phase, received signal strength (RSS) vectors from several access points (APs) are captured at reference points (RPs) in the WLAN environment and then they are handed over to a database. A pattern recognition algorithm (PRA) is employed to recognize the statistical patterns of gathered data in the training phase. The trained PRA is utilized in the test (online) phase to convert users' RSS vectors to the Cartesian coordinates of the WLAN environment.

The conventional PRAs have been created for single output scenarios, and this limitation forces us to utilize these algorithms for x and y coordinates separately. These algorithms, such as conventional Gaussian process regression (CGPR) [4], [5], support vector regression (CSVR) [6], and random forest (CRF) [7] have been used for positioning purposes, in which the optimization is performed for x and y coordinates separately. Some GPR-based algorithms have been developed for multi-task learning problems [8]. In the context of multi-task learning, the input features are different for each task, and the complexity of these techniques is at a high level, which is not suitable for a positioning system due to the battery consumptions and delays. Also, the GPR-based multi-task learning algorithms have been designed for correlated outputs, and they have apriori assumption in the training phase, such as a linearity assumption between the tasks, which means that we have to accept a linearity relation between x and y coordinates. However, the x and y outputs in fingerprint localization do not have necessarily a linear dependency in the WLAN environment. Besides, the uniqueness of the RSS vector as a fingerprint corresponds to the RP location, not its x and y coordinates separately, and the PRAs can be modified by considering this fact.

Manuscript received ?? ??, 2020; revised ?? ??, 2020.

In this letter, we introduce a novel fingerprint-based GPR positioning algorithm via a *joint* coordinate optimization within the offline phase. The proposed method, named 2D-GPR, maximizes the multivariate probability of RSS samples for 2D coordinates during the training phase of PRA, which means that it is optimized with respect to RPs not based on x and y coordinates. It uses a shared covariance matrix (which is a similarity matrix for RPs not for x and y coordinates separately) with the same hyperparameters across x and y coordinates and leverages joint information from their locations at the same time. Also, it does not need a linearity assumption during the training phase. The proposed method not only achieves better accuracy compared with CGPR in limited data samples, but also has a lower computation complexity compared with its counterpart due to its one-time optimization accomplishment in the training phase and calculation of matrix elements in the test phase.

II. Conventional GPR Based Positioning

This section describes the structure of conventional Gaussian process regression (CGPR), which is needed to apprehend the proposed 2D-GPR algorithm. First, we consider a training dataset consists of RPs' fingerprints and corresponding locations as follows

$$\mathbf{S} = [\mathbf{s}_1, \mathbf{s}_2, \cdots, \mathbf{s}_N]^T, \quad \begin{matrix} \mathbf{x} = [x_1, x_2, \cdots, x_N]^T \\ \mathbf{y} = [y_1, y_2, \cdots, y_N]^T, \end{matrix} \tag{1}$$

where $\mathbf{s}_i \in \mathbb{R}^{M\times 1}$ is the fingerprint of the i^{th} RP, M is the number of APs, and N is the number of RPs. In CGPR the aim is to obtain two functions that can map any RSS vector $\forall \mathbf{s} \in \mathbb{R}^{M\times 1}$ to the 2D Cartesian coordinates as follows

$$x = \omega_x(\mathbf{s}) + \epsilon_x, \quad y = \omega_y(\mathbf{s}) + \epsilon_y \ \text{ and } \ \epsilon_x, \epsilon_y \sim \mathcal{N}(0, \sigma_n), \tag{2}$$

where ω_x and ω_y are pattern recognition functions that can convert RSS vectors to the x and y coordinates, respectively. The ω_x and ω_y should be optimized with the training dataset in (1). The optimization process of ω_x and ω_y are similar, and in the following, we only describe this procedure for ω_x. In the Gaussian processes perspective [9], each variable x can be defined by a mean and corresponding variance. Initially,

the mean values can be considered as zero, and therefore, the vector $\mathbf{x}$ has a multivariate distribution as follows

$$\mathbf{x} \sim \mathcal{GP}(\mathbf{0}, \mathbf{C}_x), \tag{3}$$

where $\mathbf{C}_x \in \mathbb{R}^{N\times N}$ is the covariance matrix, and each element of this matrix is calculated by a user-defined kernel function. The kernel functions capture the similarity between each pair of RSS samples. Here, we use a combination of three kernels as follows

$$\begin{aligned} c_{ij}^x &= \alpha_1^2 \exp\left(-\frac{(\mathbf{s}_i - \mathbf{s}_j)^T(\mathbf{s}_i - \mathbf{s}_j)}{\gamma^2}\right) + \alpha_2^2 \mathbf{s}_i^T \mathbf{s}_j + \sigma_n^2 \delta_{ij} \\ \delta_{ij} &= \{1 \text{ if } i = j, \ \ 0 \text{ o.w}\}, \end{aligned} \tag{4}$$

where c_{ij}^x represents the i^{th} row and j^{th} column element of matrix $\mathbf{C}_x$, and $\mathbf{s}_i$ is the i^{th} row of $\mathbf{S}$. In (4) the first and second terms are squared exponential and linear kernels that elicit non-linear and linear dependencies of $\mathbf{s}_i^{th}$, respectively. Finally, the third term models the variance of ϵ_x in (2). The vector $\boldsymbol{\theta} = [\alpha_1, \alpha_2, \gamma, \sigma_n]^T$ contains the hyperparameters that play a key role for PRA design and should be optimized in training phase to maximize the log-likelihood of multivariate probability density function (PDF)

$$\tilde{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \log(p(\mathbf{x})) = \arg\min_{\boldsymbol{\theta}} (-\log(p(\mathbf{x})), \tag{5}$$

where $p(\mathbf{x})$ is the multivariate PDF defined as follows

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{N/2}|\mathbf{C}_x|^{1/2}} \exp(-\frac{1}{2}\mathbf{x}^T \mathbf{C}_x^{-1}\mathbf{x}). \tag{6}$$

Therefore, the objective function of (5) is as follows

$$\mathcal{L}(\theta) = -\log p(\mathbf{x}) = \frac{1}{2}\log|\mathbf{C}_x| + \frac{N}{2}\log(2\pi) + \frac{1}{2}\mathbf{x}^T \mathbf{C}_x^{-1}\mathbf{x}. \tag{7}$$

The optimization problem in (5) is non-convex; however, it can be solved for a locally optimum point via gradient-based algorithms such as conjugate gradient [10]. The conjugate gradient algorithm (CGA) needs the first-order gradient of $\mathcal{L}(\theta)$ to optimize the hyperparameters where for the j^{th} hyperparameter can be calculated as follows

$$\nabla\mathcal{L}(\theta_j) = -\frac{1}{2}\text{tr}((\mathbf{q}_x\mathbf{q}_x^T - \mathbf{C}_x^{-1})\frac{\partial \mathbf{C}_x}{\partial \theta_j}) \ \text{ where } \ \mathbf{q}_x = \mathbf{C}_x^{-1}\mathbf{x}. \tag{8}$$

The CGA is iterated till convergence, and then it is needed to derive the posterior distribution from the joint distribution to estimate the users' locations. Assume that the joint distribution of test and train samples is as follows

$$\begin{bmatrix} \mathbf{x} \\ \hat{\mathbf{x}} \end{bmatrix} \sim \mathcal{N}\left[\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \begin{pmatrix} \mathbf{C}_x & \mathbf{C}_x(\mathbf{s}, \hat{\mathbf{s}}) \\ \mathbf{C}_x(\hat{\mathbf{s}}, \mathbf{s}) & \mathbf{C}_x(\hat{\mathbf{s}}, \hat{\mathbf{s}}) \end{pmatrix}\right], \tag{9}$$

where $\hat{\mathbf{x}} \in \mathbb{R}^{\hat{N}\times 1}$ is the vector of users' locations that should be estimated with conditioning over the training samples $\mathbf{x}$, $\hat{N}$ is the number of test samples (users), $\mathbf{C}_x \in \mathbb{R}^{N\times N}$ is the covariance matrix between the training samples, $\mathbf{C}_x^T(\hat{\mathbf{s}}, \mathbf{s}) = \mathbf{C}_x(\mathbf{s}, \hat{\mathbf{s}}) \in \mathbb{R}^{N\times\hat{N}}$ is the covariance matrix between the test and training samples, and $\mathbf{C}_x(\hat{\mathbf{s}}, \hat{\mathbf{s}}) \in \mathbb{R}^{\hat{N}\times\hat{N}}$ is the covariance matrix between the test samples. The elements of these matrices are calculated by the optimized hyperparameters, and then, the posterior distribution can be derived as follows

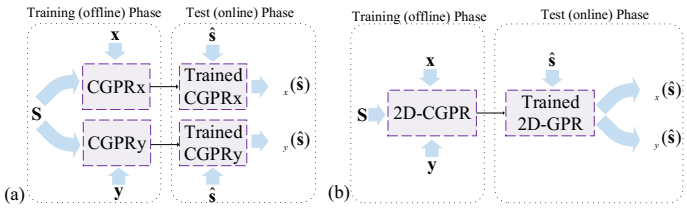

Fig. 1. Difference between (a) Conventional GPR (CGPR) and (b) Proposed 2D-GPR algorithms.

$$\begin{aligned} &\hat{\mathbf{x}}|\mathbf{x} \sim \mathcal{N}(\boldsymbol{\omega}_x, \boldsymbol{\Phi}_x) \\ &\boldsymbol{\omega}_x = \mathbf{C}_x(\hat{\mathbf{s}}, \mathbf{s})\, \mathbf{C}_x^{-1}\mathbf{x} \\ &\boldsymbol{\Phi}_x = \mathbf{C}_x(\hat{\mathbf{s}}, \hat{\mathbf{s}}) - \mathbf{C}_x(\hat{\mathbf{s}}, \mathbf{s})\mathbf{C}_x^{-1}\mathbf{C}_x(\mathbf{s}, \hat{\mathbf{s}}), \end{aligned} \tag{10}$$

where $\boldsymbol{\omega}_x$ is the pattern recognition function whose output is equal to estimated locations of (2), $\boldsymbol{\Phi}_x$ is the corresponding covariance matrix where the diagonal elements of $\boldsymbol{\Phi}_x$ are the estimated variances correspond to the variance of ϵ_x.

III. 2D-GPR Based Positioning

In this section, we present the proposed optimization procedure of hyperparameters for a 2D scenario. Fig. 1 shows the difference between the proposed 2D-GPR and CGPR algorithms. Here, each pair of x and y coordinates are considered as the characteristics of a given RP, $r = (x, y)$. Therefore, $\mathbf{x}$ and $\mathbf{y}$ have the same covariance matrix as depicted in (11)

$$(\mathbf{x}, \mathbf{y}) \sim \mathcal{GP}((\mathbf{0}, \mathbf{0}), \mathbf{C}_r), \tag{11}$$

where $\mathbf{C}_r \in \mathbb{R}^{N\times N}$ is the shared covariance matrix between x and y, and elements of this matrix are calculated by kernel functions similar to $\mathbf{C}_x$ in the previous section. Considering a shared covariance matrix $\mathbf{C}_r$ for both coordinates, in the optimization process, means that the similarities of RSS vectors are captured for the RPs, not for x and y coordinates separately. The proposed optimization problem employs both coordinates during the training phase as follows

$$\tilde{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \log(p(\mathbf{x}, \mathbf{y})) = \arg\min_{\boldsymbol{\theta}} (-\log(p(\mathbf{x}, \mathbf{y}))), \tag{12}$$

where $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}|\mathbf{y})p(\mathbf{y})$. To derive the $p(\mathbf{x}|\mathbf{y})$ we first can use the joint distribution of training coordinates

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{x} \end{bmatrix} \sim \mathcal{N}\left[\begin{pmatrix} \mathbf{0} \\ \mathbf{0} \end{pmatrix}, \underbrace{\begin{pmatrix} \mathbf{C}_y(\mathbf{s}, \mathbf{s}) & \mathbf{C}_{x,y}(\mathbf{s}, \mathbf{s}) \\ \mathbf{C}_{y,x}(\mathbf{s}, \mathbf{s}) & \mathbf{C}_x(\mathbf{s}, \mathbf{s}) \end{pmatrix}}_{\boldsymbol{\Sigma}_{x,y}}\right], \tag{13}$$

where the joint covarince matrix $\boldsymbol{\Sigma}_{x,y}$ in (13) is a singular matrix and is not invertible, because the input features and hyperparameters are the same for both coordinates. In other word, in (13) we have $\mathbf{C}_x(\mathbf{s}, \mathbf{s}) = \mathbf{C}_y(\mathbf{s}, \mathbf{s}) = \mathbf{C}_{x,y}(\mathbf{s}, \mathbf{s}) = \mathbf{C}_{y,x}(\mathbf{s}, \mathbf{s})$. Therefore, the posterior distribution cannot be derived. To avoid this problem, we can assume that $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y})$.

In the following, it can be seen that both of $\mathbf{x}$ and $\mathbf{y}$ take part to calculate the objective function $\mathcal{L}(\boldsymbol{\theta})$ and corresponding gradient $\nabla\mathcal{L}(\boldsymbol{\theta})$. First, the objective function can be calculated as follows

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}) &= -\log(p(\mathbf{x},\mathbf{y}))\\ &= \frac{1}{2}\log|\mathbf{C}_r| + \frac{N}{2}\log(2\pi) + \frac{1}{2}\mathbf{x}^T\mathbf{C}_r^{-1}\mathbf{x} + \\ &\frac{1}{2}\log|\mathbf{C}_r| + \frac{N}{2}\log(2\pi) + \frac{1}{2}\mathbf{y}^T\mathbf{C}_r^{-1}\mathbf{y}.\end{aligned} \tag{14}$$

It can be seen that the objective function $\mathcal{L}(\boldsymbol{\theta})$ depends on both training coordinates $\mathbf{x}$ and $\mathbf{y}$ with the same covariance matrix $\mathbf{C}_r$. The gradient of (14) for the j^{th} hyperparameter can be derived as follows

$$\begin{aligned}\nabla\mathcal{L}(\theta_j) &= \frac{\partial(-\log(p(\mathbf{x},\mathbf{y})))}{\partial\theta_j}\\ &= -\frac{1}{2}\mathrm{tr}((\mathbf{q}_x\mathbf{q}_x^T - \mathbf{C}_r^{-1})\frac{\partial\mathbf{C}_r}{\partial\theta_j}) - \frac{1}{2}\mathrm{tr}((\mathbf{q}_y\mathbf{q}_y^T - \mathbf{C}_r^{-1})\frac{\partial\mathbf{C}_r}{\partial\theta_j})\\ &= -\frac{1}{2}\mathrm{tr}((\mathbf{q}_x\mathbf{q}_x^T + \mathbf{q}_y\mathbf{q}_y^T - 2\mathbf{C}_r^{-1})\frac{\partial\mathbf{C}_r}{\partial\theta_j}),\end{aligned} \tag{15}$$

where $\mathbf{q}_x = \mathbf{C}_r^{-1}\mathbf{x}$ and $\mathbf{q}_y = \mathbf{C}_r^{-1}\mathbf{y}$ take both coordinates during the optimization of PRA. As explained in the previous section, CGA can be applied to update the hyperparameters till convergence. The joint distribution of test and train samples in this 2D scenario can be written as follows

$$\begin{bmatrix}(\mathbf{x},\mathbf{y})\\(\hat{\mathbf{x}},\hat{\mathbf{y}})\end{bmatrix} \sim \mathcal{N}\left[\begin{pmatrix}(\mathbf{0},\mathbf{0})\\(\mathbf{0},\mathbf{0})\end{pmatrix}, \begin{pmatrix}\mathbf{C}_r & \mathbf{C}_r(\mathbf{s},\hat{\mathbf{s}})\\ \mathbf{C}_r(\hat{\mathbf{s}},\mathbf{s}) & \mathbf{C}_r(\hat{\mathbf{s}},\hat{\mathbf{s}})\end{pmatrix}\right], \tag{16}$$

where $\hat{\mathbf{x}},\hat{\mathbf{y}} \in \mathbb{R}^{\hat{N}\times 1}$ are the vectors of users' locations that should be estimated with conditioning over the training samples $\mathbf{x},\mathbf{y}$, $\hat{N}$ is the number of test observations (users), $\mathbf{C}_r \in \mathbb{R}^{N\times N}$ is the covariance matrix between the training observations, $\mathbf{C}_r^T(\hat{\mathbf{s}},\mathbf{s}) = \mathbf{C}_r(\mathbf{s},\hat{\mathbf{s}}) \in \mathbb{R}^{N\times\hat{N}}$ is the covariance matrix between the test and training observations, and $\mathbf{C}_r(\hat{\mathbf{s}},\hat{\mathbf{s}}) \in \mathbb{R}^{\hat{N}\times\hat{N}}$ is the covariance matrix between the test observations. The posterior distribution can be derived as follows

$$\begin{aligned}&(\hat{\mathbf{x}},\hat{\mathbf{y}})|(\mathbf{x},\mathbf{y}) \sim \mathcal{N}((\omega_x,\omega_y),\boldsymbol{\Phi}_r)\\ &\boldsymbol{\omega}_x = \mathbf{C}_r(\hat{\mathbf{s}},\mathbf{s})\,\mathbf{C}_r^{-1}\mathbf{x}\\ &\boldsymbol{\omega}_y = \mathbf{C}_r(\hat{\mathbf{s}},\mathbf{s})\,\mathbf{C}_r^{-1}\mathbf{y}\\ &\boldsymbol{\Phi}_r = \mathbf{C}_r(\hat{\mathbf{s}},\hat{\mathbf{s}}) - \mathbf{C}_r(\hat{\mathbf{s}},\mathbf{s})\mathbf{C}_r^{-1}\mathbf{C}_r(\mathbf{s},\hat{\mathbf{s}}).\end{aligned} \tag{17}$$

Proof: To derive the posterior distribution, we explain the process for the $\boldsymbol{\omega}_x$ with respect to the shared covariance matrix that can similarly be employed for $\boldsymbol{\omega}_y$. Initially, from the Bayes rule, we have

$$\begin{aligned}p(\hat{\mathbf{x}}|\mathbf{x}) &= \frac{p(\hat{\mathbf{x}},\mathbf{x})}{p(\mathbf{x})} \sim \frac{\exp(-\frac{1}{2}\bar{\mathbf{x}}^T\Sigma_r^{-1}\bar{\mathbf{x}})}{\exp(-\frac{1}{2}\mathbf{x}^T\mathbf{C}_r^{-1}\mathbf{x})}\\ &= \exp(-\frac{1}{2}(\bar{\mathbf{x}}^T\Sigma_r^{-1}\bar{\mathbf{x}} - \mathbf{x}^T\mathbf{C}_r^{-1}\mathbf{x})),\end{aligned} \tag{18}$$

where,

$$\bar{\mathbf{x}} = \begin{bmatrix}\mathbf{x}\\ \hat{\mathbf{x}}\end{bmatrix},\quad \Sigma_r = \begin{bmatrix}\mathbf{C}_r & \mathbf{C}_r(\mathbf{s},\hat{\mathbf{s}})\\ \mathbf{C}_r(\hat{\mathbf{s}},\mathbf{s}) & \mathbf{C}_r(\hat{\mathbf{s}},\hat{\mathbf{s}})\end{bmatrix}, \tag{19}$$

where $\boldsymbol{\Sigma}_r^{-1}$ is the inverse of 2×2 block matrix that can be calculated as below

$$\begin{aligned}&\boldsymbol{\Sigma}_r^{-1} =\\ &\begin{bmatrix}\mathbf{C}_r^{-1} + \mathbf{C}_r^{-1}\mathbf{C}_r(\mathbf{s},\hat{\mathbf{s}})\mathbf{H}\mathbf{C}_r(\hat{\mathbf{s}},\mathbf{s})\mathbf{C}_r^{-1} & -\mathbf{C}_r^{-1}\mathbf{C}_r(\mathbf{s},\hat{\mathbf{s}})\mathbf{H}\\ -\mathbf{H}\mathbf{C}_r(\hat{\mathbf{s}},\mathbf{s})\mathbf{C}_r^{-1} & \mathbf{H}\end{bmatrix}\\ &\text{where,}\quad \mathbf{H} = (\mathbf{C}_r(\hat{\mathbf{s}},\hat{\mathbf{s}}) - \mathbf{C}_r(\hat{\mathbf{s}},\mathbf{s})\mathbf{C}_r^{-1}\mathbf{C}_r(\mathbf{s},\hat{\mathbf{s}}))^{-1}.\end{aligned} \tag{20}$$

the above equation can be substituted to inner term in (18) which can be simplified as follows

$$\begin{aligned}&\bar{\mathbf{x}}^T\boldsymbol{\Sigma}_r^{-1}\bar{\mathbf{x}} - \mathbf{x}^T\mathbf{C}_r^{-1}\mathbf{x} = \begin{bmatrix}\mathbf{x}^T & \hat{\mathbf{x}}^T\end{bmatrix}\boldsymbol{\Sigma}_r^{-1}\begin{bmatrix}\mathbf{x}\\ \hat{\mathbf{x}}\end{bmatrix} - \mathbf{x}^T\mathbf{C}_r^{-1}\mathbf{x}\\ &= \big(\mathbf{x}^T\mathbf{C}_r^{-1}\mathbf{C}_r(\mathbf{s},\hat{\mathbf{s}})\mathbf{H}\mathbf{C}_r(\hat{\mathbf{s}},\mathbf{s})\mathbf{C}_r^{-1}\mathbf{x} - \mathbf{x}^T\mathbf{C}_r^{-1}\mathbf{C}_r(\mathbf{s},\hat{\mathbf{s}})\mathbf{H}\hat{\mathbf{x}}\big) +\\ &\qquad\big(-\hat{\mathbf{x}}^T\mathbf{H}\mathbf{C}_r(\hat{\mathbf{s}},\mathbf{s})\mathbf{C}_r^{-1}\mathbf{x} + \hat{\mathbf{x}}^T\mathbf{H}\hat{\mathbf{x}}\big)\\ &= (\mathbf{x}^T\mathbf{C}_r^{-1}\mathbf{C}_r(\mathbf{s},\hat{\mathbf{s}})\mathbf{H})(\mathbf{C}_r(\hat{\mathbf{s}},\mathbf{s})\mathbf{C}_r^{-1}\mathbf{x} - \hat{\mathbf{x}}) +\\ &\qquad\hat{\mathbf{x}}^T\mathbf{H}(-\mathbf{C}_r(\hat{\mathbf{s}},\mathbf{s})\mathbf{C}_r^{-1}\mathbf{x} + \hat{\mathbf{x}})\\ &= (-\mathbf{x}^T\mathbf{C}_r^{-1}\mathbf{C}_r(\mathbf{s},\hat{\mathbf{s}})\mathbf{H} + \hat{\mathbf{x}}^T\mathbf{H})(\hat{\mathbf{x}} - \mathbf{C}_r(\hat{\mathbf{s}},\mathbf{s})\mathbf{C}_r^{-1}\mathbf{x})\\ &= (\hat{\mathbf{x}} - \mathbf{C}_r(\hat{\mathbf{s}},\mathbf{s})\mathbf{C}_r^{-1}\mathbf{x})^T\mathbf{H}(\hat{\mathbf{x}} - \mathbf{C}_r(\hat{\mathbf{s}},\mathbf{s})\mathbf{C}_r^{-1}\mathbf{x}).\end{aligned} \tag{21}$$

therefore, Eq. (18) can be simplified as follows

$$p(\hat{\mathbf{x}}|\mathbf{x}) \sim \exp\left(\frac{-1}{2}\left(\hat{\mathbf{x}} - \mathbf{C}_r(\hat{\mathbf{s}},\mathbf{s})\mathbf{C}_r^{-1}\mathbf{x}\right)^T\mathbf{H}\left(\hat{\mathbf{x}} - \mathbf{C}_r(\hat{\mathbf{s}},\mathbf{s})\mathbf{C}_r^{-1}\mathbf{x}\right)\right). \tag{22}$$

from (22) it can be seen that the posterior mean vector and covariance matrix are as follows

$$\begin{cases}\boldsymbol{\omega}_x = \mathbf{C}_r(\hat{\mathbf{s}},\mathbf{s})\mathbf{C}_r^{-1}\mathbf{x}\\ \boldsymbol{\Phi}_r = \mathbf{H}^{-1} = \mathbf{C}_r(\hat{\mathbf{s}},\hat{\mathbf{s}}) - \mathbf{C}_r(\hat{\mathbf{s}},\mathbf{s})\mathbf{C}_r^{-1}\mathbf{C}_r(\mathbf{s},\hat{\mathbf{s}}).\end{cases} \tag{23}$$

Note that $\boldsymbol{\Phi}_r$ is equal for both coordinates because it does not depend on x and y.

Complexity Analysis: In the training phase, 2D-GPR needs the inverse of $\mathbf{C}_r$ in each iteration of CGA for (15) that causes $O(N^3)$ calculations. This process is done two times to calculate the $\mathbf{C}_x$ and $\mathbf{C}_y$ for CGPR. In the test phase, 2D-GPR calculates the $\mathbf{C}_r(\hat{\mathbf{s}},\mathbf{s})\,\mathbf{C}_r^{-1}$ in (17) which is equal for $\boldsymbol{\omega}_x$ and $\boldsymbol{\omega}_y$, whereas CGPR performs this process two times for $\mathbf{C}_x(\hat{\mathbf{s}},\mathbf{s})\,\mathbf{C}_x^{-1}$ and $\mathbf{C}_y(\hat{\mathbf{s}},\mathbf{s})\,\mathbf{C}_y^{-1}$. Therefore, the proposed method has lower complexity compared with CGPR during the training and test phases.

IV. Experimental Results

The proposed method has been implemented using the R programming language, and we use a dataset introduced in [5] to evaluate it over competitors. This dataset consists of 250 points, and each point has 75 samples from 27 Wi-Fi APs where the area size is $120m^2$. Here, 9 out of 27 APs that are more powerful in the environment have been selected. We conduct several experiments to compare the proposed 2D-GPR algorithm with three of the most popular baseline PRAs including CGPR [4], [5], CSVR [6], and CRF [7]. We use the Monte-Carlo cross-validation method [11] to evaluate these algorithms in which the train and test samples from 250 points are randomly selected K times, and the average of test errors is reported to reduce the error bias. The mean average error (MEA) over these K times is calculated as follows

$$\mathrm{MAE} = \frac{1}{K\hat{N}}\sum_{k=1}^{K}\sum_{\hat{n}=1}^{\hat{N}}\sqrt{(\hat{\mathbf{x}}_{\hat{n}}^k - [\omega_x]_{\hat{n}}^k)^2 + (\hat{\mathbf{y}}_{\hat{n}}^k - [\omega_y]_{\hat{n}}^k)^2}, \tag{24}$$

TABLE I
THE MAE(m) OF DIFFERENT ALGORITHMS VIA DIFFERENT EXPERIMENT SETUP FOR DIFFERENT NUMBER OF RPS, APS, AND SAMPLES.

RP (N) ⇒	40									80									150								
AP (M) ⇒	3			5			9			3			5			9			3			5			9		
Sample ⇒	3	5	15	3	5	15	3	5	15	3	5	15	3	5	15	3	5	15	3	5	15	3	5	15	3	5	15
2D-GPR	3.28	2.97	2.58	2.93	2.62	2.24	2.91	2.60	2.23	2.78	2.49	2.11	2.61	2.30	1.92	2.48	2.16	1.78	2.67	2.37	1.99	2.40	2.10	1.74	2.21	1.91	1.56
CGPR [4], [5]	3.76	3.42	3.01	3.84	3.46	3.00	3.07	2.73	2.53	2.97	2.64	2.24	2.64	2.32	1.95	2.63	2.37	1.96	2.77	2.46	2.07	2.42	2.12	1.76	2.25	1.94	1.59
CSVR [6]	3.78	3.52	3.22	3.57	3.30	3.01	3.42	3.15	2.87	3.03	2.77	2.43	2.84	2.55	2.24	2.69	2.40	2.10	2.83	2.52	2.15	2.60	2.30	1.97	2.40	2.09	1.78
CRF [7]	4.35	4.05	3.72	3.84	3.53	3.18	3.35	3.03	2.68	3.67	3.36	2.98	3.26	2.92	2.53	2.89	2.58	2.19	3.29	2.94	2.53	2.90	2.54	2.15	2.49	2.16	1.80

where $\hat{N}$ is the number of test samples, $\hat{\mathbf{x}}_{\hat{n}}^{k}$ and $\hat{\mathbf{y}}_{\hat{n}}^{k}$ are the location of $\hat{n}^{th}$ user within the k^{th} iteration, and $[\boldsymbol{\omega}_x]_{\hat{n}}^{k}$ and $[\boldsymbol{\omega}_y]_{\hat{n}}^{k}$ are the estimated location of the $\hat{n}^{th}$ user within the k^{th} iteration. In our experiments, we choose $K = 10$ and $\hat{N} = (250 - N)(75/\text{Sample})$, where "Sample" is the number of available samples for each user. We reported the MAE of different experiment setup in Table I, for a different number of RPs, APs, and samples. To reduce the effect of small-scale fading, the RSS samples obtained from multiple times can be averaged out [4]. Here, all of the 75 training samples are averaged out. The number of samples in Table I refers to the available samples for each user that are also averaged out. Since the environment area is $120m^2$, for 40, 80, and 150 RPs, we have one RP per 3, 1.5, and 0.8 m^2, respectively. With the same reasoning, when the number of APs is 9, 5, and 3, we have one AP per 13.3, 24, and 40 m^2, respectively. In the following, we use (RP, AP, Sample) to describe Table I. For instance, (40, 5, 3) refers to 40 RPs, 5 APs, and 3 available samples for users. It can be seen that when the number of RPs and APs is limited, 2D-GPR is more effective than other PRAs. The best record of 2D-GPR compared with other PRAs occurs in (40, 5, 15). It means that when there is one RP per $3m^2$ and one AP per $24m^2$ the proposed method is more effective. On the other side, in ideal scenarios where the number of RPs and APs are enough (e.g., (150, 9, 15)), all PRAs have similar performance, particularly 2D-GPR and CGPR. This observation does not reduce the strength of the proposed 2D-GPR algorithm, because several studies on PRAs show that they are saturated by increasing the number of training data [1], [12].

In Fig 2 we have illustrated the boxplot of K times iteration for a low RPs and APs density scenario (40, 5, 15), and a high RPs and APs density scenario (150, 9, 15) where the horizontal lines in this figure show the median error. As can be seen, 2D-GPR in the low-density scenario has less median error and variance. On the other hand, in the high-density scenario, 2D-GPR is similar to CGPR and a little bit better than CRF and CSVR. It can be concluded that in large scale areas with low densities of RPs and APs, our proposed method is more effective. Besides, the proposed method is preferable due to its less calculation cost compared with CGPR.

V. CONCLUSION

We proposed a novel fingerprint-based positioning method, named 2D-GPR, which is an evolved version of CGPR for a two-dimensional scenario. We conducted several experiments with different experiment setups, and numerical results proved the superiority of this algorithm over baseline algorithms, especially when the densities of RPs and APs are limited. The proposed method also has less complexity compared with CGPR.

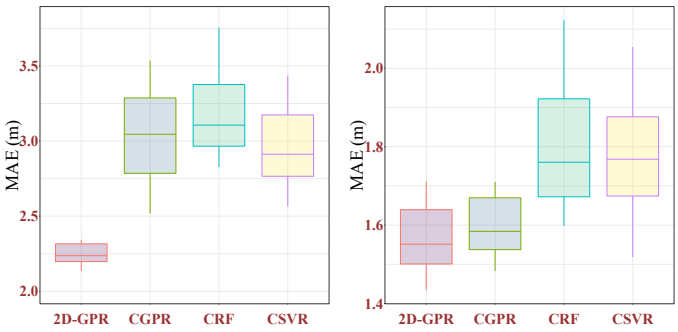

Fig. 2. Boxplot for two scenarios of Table I where (a) is a low RPs and APs density scenario (40, 5, 15) and (b) is a high RPs and APs density scenario (150, 5, 15).

REFERENCES

[1] M. Nabati, H. Navidan, R. Shahbazian, S. A. Ghorashi, and D. Windridge, "Using synthetic data to enhance the accuracy of fingerprint-based localization: A deep learning approach," *IEEE Sensors Letters*, vol. 4, no. 4, pp. 1–4, 2020.

[2] Y. Tao and L. Zhao, "Fingerprint localization with adaptive area search," *IEEE Communications Letters*, vol. 24, no. 7, pp. 1446–1450, 2020.

[3] A. Khalajmehrabadi, N. Gatsis, and D. Akopian, "Modern wlan fingerprinting indoor positioning methods and deployment challenges," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1974–2002, 2017.

[4] K. N. R. S. V. Prasad, E. Hossain, and V. K. Bhargava, "Machine learning methods for rss-based user positioning in distributed massive mimo," *IEEE Transactions on Wireless Communications*, vol. 17, no. 12, pp. 8402–8417, 2018.

[5] E. Homayounvala, M. Nabati, R. Shahbazian, S. A. Ghorashi, and V. Moghtadaiee, "A novel smartphone application for indoor positioning of users based on machine learning," in *Adjunct Proceedings of the ACM International Joint Conference on UbiComp/ISW*, 2019, p. 430–437.

[6] W. Kim, J. Park, J. Yoo, H. J. Kim, and C. G. Park, "Target localization using ensemble support vector regression in wireless sensor networks," *IEEE Transactions on Cybernetics*, vol. 43, no. 4, pp. 1189–1198, 2013.

[7] X. Guo, N. Ansari, L. Li, and H. Li, "Indoor localization by fusing a group of fingerprints based on random forests," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4686–4698, 2018.

[8] H. Liu, J. Cai, and Y.-S. Ong, "Remarks on multi-output gaussian process regression," *Knowledge-Based Systems*, vol. 144, pp. 102–121, 2018.

[9] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press Cambridge, MA, 2006, vol. 2, no. 3.

[10] J. Nocedal and S. Wright, *Numerical optimization*. Springer Science & Business Media, 2006.

[11] Q.-S. Xu and Y.-Z. Liang, *Monte Carlo cross validation*. Elsevier, 2001, vol. 56, no. 1.

[12] X. Zhu, C. Vondrick, C. C. Fowlkes, and D. Ramanan, "Do we need more training data?" *International Journal of Computer Vision*, vol. 119, no. 1, pp. 76–92, 2016.