Kernel-Based Machine Learning Using Radio-Fingerprints for Localization in WSNs

SANDY MAHFOUZ
FARAH MOURAD-CHEHADE
PAUL HONEINE
Université de Technologie de Troyes
Troyes, France

JOUMANA FARAH
Lebanese University
Roumieh, Lebanon

HICHEM SNOUSSI
Université de Technologie de Troyes
Troyes, France

This paper introduces an original method for sensors localization in WSNs. Based on radio-location fingerprinting and machine learning, the method consists of defining a model whose inputs and outputs are, respectively, the received signal strength indicators and the sensors locations. To define this model, several kernel-based machine-learning techniques are investigated, such as the ridge regression, support vector regression, and vector-output regularized least squares. The performance of the method is illustrated using both simulated and real data.

Manuscript received January 24, 2014; revised July 21, 2014; released for publication November 25, 2014.

DOI. No. 10.1109/TAES.2015.140061.

Refereeing of this contribution was handled by S. Marano.

This work is supported by the Champagne-Ardenne Region in France, grant WiDiD: Wireless Diffusion Detection.

Authors' addresses: S. Mahfouz, F. Mourad-Chehade, P. Honeine, and H. Snoussi, Institut Charles Delaunay (CNRS), Université de Technologie de Troyes, 12 rue Marie Curie, Troyes 10004, France, Email: (sandy.mahfouz@utt.fr); J. Farah, Faculty of Engineering, Lebanese University, Roumieh, Mount Lebanon.

I. INTRODUCTION

Recent advances in radio and embedded systems have enabled the proliferation of wireless sensor networks (WSNs). These networks are composed of a large number of low-cost, low-power, multifunctional sensor nodes that are small in size and spatially distributed, and can exchange data [1]. Nowadays, WSNs are being widely implemented in many different scenarios to perform a number of tasks, ranging from home monitoring [2] to, e.g., environmental monitoring [3], covering industrial monitoring [4], battlefield surveillance [5], and medical applications [6]. In most applications, the data collected by the sensors are only meaningful if they are coupled with the correct locations of the corresponding sensors. Therefore, sensor localization has become a fundamental issue, especially in WSNs, where sensors lack a fixed infrastructure and are able to move in an uncontrollable manner. A direct way to locate the nodes is the use of Global Positioning System (GPS) devices. However, this solution is not practical because of the high energy consumption of GPS receivers and their limited performance in indoor environments. An alternative solution is to develop localization algorithms that estimate the unknown sensors locations with respect to others having fixed known locations. Two types of sensors are thus defined: those with known positions, called anchors, and the others having unknown locations, called nodes. Anchors positions can be obtained, for instance, by setting them at fixed locations with known coordinates, whereas nodes are localized by exchanging information with anchors.

Many localization algorithms using anchors have been proposed. They are mainly based on estimating the distances between the anchors and the nodes. Such methods are either time based using the time-of-arrival (ToA) or time-difference-of-arrival (TDoA) techniques [7], or by using the angle-of-arrival (AoA) technique [8] or received signal strength indicator (RSSI)-based methods [9]. Distance estimates are then combined using, e.g., triangulation [10] or trilateration [11] to find nodes positions. Compared to ToA, TDoA, and AoA, RSSI-based methods are being widely used due to their low-power consumption and cost competitiveness because no extra devices are needed. However, methods that estimate exact distances using RSSI are not efficient because the measurements of signals' powers could be significantly altered by the presence of additive noise, multipath fading, shadowing, and other interferences. Alternative RSSI-based methods for sensors localization employ connectivity measurements [12]. Instead of estimating exact distances, these techniques compare the RSSIs of a considered node to fixed power thresholds in order to detect all its neighboring anchors. Such an approach assumes that all anchors have circular transmission ranges with known radii and that signals powers decrease monotonically with the increase of the nodes traveled distances. Localizing a given sensor

consists then on computing the intersection of the communication disks of its neighboring anchors using, e.g., a particle filter [13], interval analysis [14, 15], polar-interval analysis [16], and a variational filter [17]. For instance, in [14], the authors use intervals to perform an outer approximation of the solution areas, leading to boxes guaranteed to include the actual locations of the nodes.

Other more reliable techniques for RSSI-based localization are based on fingerprinting [18]. Compared to connectivity-based methods, these techniques do not make any assumption on the communications in the network. However, they need a preconfiguration phase where a collection of fingerprints is performed. Indeed, a typical fingerprinting location-sensing system consists of two phases: an offline phase and an online phase. In the offline phase, a set of reference locations is generated within the surveillance area, and the RSSIs of the signals sent by the anchors and received at these reference positions are collected. A database of fingerprints is then obtained; it is composed of the set of reference positions with their associated RSSIs. In the online phase, nodes move and collect RSSI information, which is then combined with the collected fingerprints database to compute their locations. The advantage of location fingerprinting technique is that it takes into account the stationary characteristics of the environment, such as multipath propagation and wall attenuation. A well-known algorithm based on location fingerprinting is the weighted K-nearest neighbor (WKNN) algorithm [19]. In this algorithm, the RSSI values collected by the nodes are compared to the samples in the database, and the nodes positions are given by weighted combinations of the K nearest neighboring positions; the nearness indicator for this method is based on the Euclidean distance between RSSIs. Motivated by the complexity of the RSSI patterns, the authors of [20] propose a kernelized combination algorithm that compares the RSSI values collected by the nodes with the samples in the database. Moreover, they provide a new method for choosing the training samples to be used in the weighting process instead of using the K nearest neighboring positions as in traditional WKNN. Another fingerprinting localization method is proposed in [21], where the authors consider location estimation as a machine-learning problem and use support vector regression (SVR) for localization. However, instead of using the RSSIs of the signals exchanged in a WSN to construct a database of fingerprints, the authors use signals from the Global System for Mobile communication (GSM) network. Such methods can only be used in areas where GSM coverage is available, and a GSM receiver is then necessary, limiting thus the possible range of applications of the method.

In this paper, we propose a new localization technique that combines location fingerprinting and learning methods. Being a fingerprinting-based method, the proposed technique is composed of offline and online phases. In addition to fingerprints collection, the offline phase consists of a training phase where a model is defined, associating to the RSSIs of the database the exact reference positions where they are collected. In the online phase, the RSSI measures collected by the nodes while moving are used with the computed model to estimate node positions. The definition of the model is done using several kernel-based machine-learning algorithms [22], such as ridge regression or least-squares support vector machine (LS SVM) [23], SVR [24], and vector-output regularized least squares (vo-RLS) [25]. This paper provides a framework for all these machine-learning techniques and thus extends naturally our previous work [26], where the bias-free ridge regression was only considered. We then develop new theoretical results, using all the above techniques and provide a thorough comparison between them in terms of accuracy and time complexity with many new experimental scenarios and results.

TABLE I
List of the Used Variables Along with Their Respective Sizes

Variable	Size
$\boldsymbol{a}_i, \boldsymbol{x}, \boldsymbol{p}_\ell, \boldsymbol{\psi}(\cdot), \boldsymbol{P}_{\ell,*}, \boldsymbol{\alpha}_{\ell,*}, \boldsymbol{b}$	1-by-D
$\boldsymbol{\rho}, \boldsymbol{\rho}_\ell$	N_a-by-1
$\boldsymbol{P}, \boldsymbol{\alpha}$	N_p-by-D
$\boldsymbol{P}_{*,d}, \boldsymbol{\alpha}_{*,d}, \boldsymbol{\beta}, \boldsymbol{v}$	N_p-by-1
$\boldsymbol{K}, \boldsymbol{G}$	N_p-by-N_p

The rest of the paper is organized as follows. Section II outlines the proposed localization technique. Section III describes the use of kernel methods and introduces the different machine-learning algorithms that can be used. In Section IV, we study the effectiveness of these algorithms. Finally, Section V concludes the paper.

II. DESCRIPTION OF THE LOCALIZATION METHOD

We consider an environment of D dimensions, with $D = 2$ for a two-dimensional environment or $D = 3$ for a three-dimensional environment. Also, two types of sensors are considered: anchors and mobile nodes. Anchors have known fixed locations, denoted by $\boldsymbol{a}_i$, $i \in \{1, \ldots, N_a\}$, whereas mobile nodes are moving with unknown positions, denoted by $\boldsymbol{x}_j$, $j \in \{1, \ldots, N_x\}$, and hence they need to be regularly localized. To avoid confusion, it is worth noting that all coordinates are D-dimension row vectors. Without loss of generality, because nodes are localized independently from each other, using only anchors information, we will withdraw the j index in the sequel. Therefore, only one mobile node with the unknown position $\boldsymbol{x}$ is considered, knowing that the proposed method could be applied to all nodes to be localized similarly. In Table I, all the variables that will be used in the following are listed, along with their respective sizes. Note that N_p will be defined in the following, and $\mathbf{0}$ and $\mathbf{1}$ are row vectors of, respectively, zeros and ones of appropriate sizes. Note also that ℓ and d are indices taking values, respectively, in $\{1, \ldots, N_p\}$ and $\{1, \ldots, D\}$.

A. Problem Statement

The proposed algorithm is based on fingerprinting, and hence it consists of two phases: offline and online phases. In the offline phase, N_p offline positions, denoted by $\boldsymbol{p}_\ell$, $\ell \in \{1, \ldots, N_p\}$, are generated uniformly or randomly in the studied region. Then, anchors broadcast signals in the network at a fixed initial power. Meanwhile, a sensor is temporarily placed at each offline position $\boldsymbol{p}_\ell$ to detect the anchors signals and to measure their RSSIs. All anchors signals are assumed to be received at all offline positions. Let $\boldsymbol{\rho}_\ell = (\rho_{\boldsymbol{a}_1,\boldsymbol{p}_\ell} \ldots \rho_{\boldsymbol{a}_{N_a},\boldsymbol{p}_\ell})^\top$ be the column vector of RSSIs sent by all N_a anchors and received at the position $\boldsymbol{p}_\ell$. This way, a database of N_p pairs $(\boldsymbol{\rho}_\ell, \boldsymbol{p}_\ell)$ is obtained, where $\ell \in \{1, \ldots, N_p\}$. Based on the information from the database, the objective is now to define a function

$$\boldsymbol{\psi}(\cdot) : \mathbb{R}^{N_a} \mapsto \mathbb{R}^D$$

that associates to each RSSI vector $\boldsymbol{\rho}_\ell$ the corresponding position $\boldsymbol{p}_\ell$. Kernel methods [22, 23] provide an elegant framework to find the model $\boldsymbol{\psi}(\cdot)$, where " $\cdot$ " is the model's input as will be shown in the following paragraph. In the online phase, the defined model $\boldsymbol{\psi}(\cdot)$ is used to estimate the node's position. Indeed, the node receives signals from the N_a anchors, at a given time, measures their RSSI values, and stores them into a vector $\boldsymbol{\rho}$. Its estimated coordinates are then given by:

$$\hat{\boldsymbol{x}} = \boldsymbol{\psi}(\boldsymbol{\rho}).$$

Note that the database construction and the computation of the model $\boldsymbol{\psi}(\cdot)$ are performed only once at a computation center, in the offline phase. The model is then communicated to the moving node that performs all subsequent computations in the phase.

B. Definition of $\boldsymbol{\psi}(\cdot)$ Using Kernel Methods

Univariate regression has been one of the biggest concerns of researchers in the field of system modeling. Regression analysis divides up into linear regression and nonlinear regression. It is used to predict a continuous dependent variable or response from a number of independent variables or input variables. Here, the objective is to find the model $\boldsymbol{\psi}(\cdot)$ that associates to each entry $\boldsymbol{\rho}_\ell$ the corresponding output $\boldsymbol{p}_\ell$. Consider a set $\{(\boldsymbol{\rho}_\ell, p_{\ell,d})\}_{\ell=1}^{N_p}$, where $d \in \{1, \ldots, D\}$, and $p_{\ell,d}$ is an element of $\boldsymbol{p}_\ell = (p_{\ell,1} \ldots p_{\ell,D})$. Let $\boldsymbol{\psi}(\cdot) = (\psi_1(\cdot) \ldots \psi_D(\cdot))$, where $\psi_d(\cdot) : \mathbb{R}^{N_a} \mapsto \mathbb{R}$ estimates the dth coordinate in $\boldsymbol{p}_\ell = (p_{\ell,1} \ldots p_{\ell,D})$, for an input $\boldsymbol{\rho}_\ell$. The functions $\psi_d(\cdot)$ to be determined are now univariate.

This section takes advantage of the kernel-based machine learning to determine the functions $\psi_d(\cdot)$. Consider a reproducing kernel (i.e., positive definite) $\kappa : \mathbb{R}^{N_a} \times \mathbb{R}^{N_a} \mapsto \mathbb{R}$, and denote by $\mathcal{H}$ its reproducing kernel Hilbert space (RKHS) with inner product $\langle \cdot, \cdot \rangle_{\mathcal{H}}$. From the reproducing property [22], every $\psi_d(\cdot)$ of $\mathcal{H}$ can be evaluated at any $\boldsymbol{\rho}_\ell \in \mathbb{R}^{N_a}$ by $\psi_d(\boldsymbol{\rho}_\ell) = \langle \psi_d(\cdot), \kappa(\cdot, \boldsymbol{\rho}_\ell) \rangle_{\mathcal{H}}$. Table II shows some of the most commonly used kernel functions.

TABLE II
Some Commonly Used Reproducing Kernels with Parameters $c, \sigma > 0$, and $q \in \mathbb{N}_+$

Type	General Form
Polynomial	$\kappa(\boldsymbol{\rho}_s, \boldsymbol{\rho}_{s'}) = \left(c + \boldsymbol{\rho}_s^\top \boldsymbol{\rho}_{s'}\right)^q$
Exponential	$\kappa(\boldsymbol{\rho}_s, \boldsymbol{\rho}_{s'}) = \exp\left(\frac{1}{\sigma} \boldsymbol{\rho}_s^\top \boldsymbol{\rho}_{s'}\right)$
Gaussian	$\kappa(\boldsymbol{\rho}_s, \boldsymbol{\rho}_{s'}) = \exp\left(-\frac{1}{2\sigma^2} \left\| \boldsymbol{\rho}_s - \boldsymbol{\rho}_{s'} \right\|^2\right)$

The function $\psi_d(\cdot)$ is obtained by minimizing the error between the model's outputs $\psi_d(\boldsymbol{\rho}_\ell)$ and the desired outputs $p_{\ell,d}$, namely, by minimizing the following regularized empirical risk:

$$\mathscr{C}((p_{1,d}, \psi_d(\boldsymbol{\rho}_1)), \ldots, (p_{N_p,d}, \psi_d(\boldsymbol{\rho}_{N_p}))) + \eta \mathscr{R}(\|\psi_d\|_{\mathcal{H}}^2), \tag{1}$$

where $\mathscr{C}$ is an arbitrary cost function, such as the mean squared error, $\mathscr{R}$ is a strictly monotonically increasing real-valued function on $[0, \infty)$, and $\|\cdot\|_{\mathcal{H}}$ is the norm in the RKHS. Here, the second term is a regularization term, with η a positive tunable parameter that controls the trade-off between the fitness error and the complexity of the solution as measured by the norm in the RKHS.

All machine-learning algorithms share the same fundamental foundation: the representer theorem [23, 27]. It is a key property that underlines the success of the kernel methods by allowing one to conduct all optimization in a space whose dimension does not exceed the size of the training set. According to this theorem, it turns out that the problem can be reduced, without loss of generality, to an optimization problem that is much more computation friendly. Hence, the minimizer of the regularized empirical risk (1) is of the following form:

$$\psi_d(\cdot) = \sum_{\ell=1}^{N_p} \alpha_{\ell,d} \kappa(\boldsymbol{\rho}_\ell, \cdot). \tag{2}$$

In this way, the problem is described in a simpler form such as to determine the $\alpha_{\ell,d}$, changing it into a finite dimensional one, and the minimizer $\psi_d(\cdot)$ is expressed as a finite linear combination of the kernels centered at the training samples. In the following section, we show how the weights in the defined models can be computed using several forms of the cost function $\mathscr{C}$ and the regularization term $\mathscr{R}$.

For the sake of clarity, we introduce the notations used in the following section. Let $\boldsymbol{P} = (\boldsymbol{p}_1^\top \ldots \boldsymbol{p}_{N_p}^\top)^\top$. The matrix $\boldsymbol{P}$ is then of size N_p-by-D having $p_{\ell,d}$ for the (ℓ,d)th entry and $\boldsymbol{p}_\ell$ for the ℓth row. In the following, we denote $\boldsymbol{p}_\ell$ by $\boldsymbol{P}_{\ell,*}$ and the dth column of $\boldsymbol{P}$ by $\boldsymbol{P}_{*,d}$. The vector $\boldsymbol{P}_{*,d}$ holds now all N_p points for a fixed coordinate d. In a similar manner, we define the matrix $\boldsymbol{\alpha}$ whose (ℓ,d)th entry is $\alpha_{\ell,d}$. The dth column of $\boldsymbol{\alpha}$ is now denoted by $\boldsymbol{\alpha}_{*,d}$ and the ℓth row of $\boldsymbol{\alpha}$ by $\boldsymbol{\alpha}_{\ell,*}$.

III. LEARNING ALGORITHMS

As mentioned in the previous section, the objective is to find a set of functions $\psi_d(\cdot)$, where $d \in \{1, \ldots, D\}$, that

associate to each RSSI vector $\boldsymbol{\rho}_\ell$ the corresponding coordinate $p_{\ell,d}$. According to the representer theorem, this function should have the form given in (2) in order to minimize (1). The fitness term $\mathscr{C}$ and the regularization term $\mathscr{R}$ of the general problem (1) can be written in different forms, thus, leading to different optimization problems and different solutions, as it will be shown in the following. In the first two subsections, we consider two different cases of the ridge regression in which the cost function is based on mean squared loss. SVR is discussed in the third subsection, where the cost function is taken using hinge loss. Finally, a multitasking approach is considered in the fourth subsection with the vo-RLS algorithm.

A. Ridge Regression or LS SVM

The solution is first defined using the original form of the optimization problem given in [23], i.e., the ridge regression. In its original form, the fitness term of (1) is taken using the mean squared loss, namely,

$$\mathscr{C}\left((p_{1,d}, \psi_d(\boldsymbol{\rho}_1)), \ldots, (p_{N_p,d}, \psi_d(\boldsymbol{\rho}_{N_p}))\right) = \frac{1}{N_p}\sum_{\ell=1}^{N_p}(p_{\ell,d} - \psi_d(\boldsymbol{\rho}_\ell))^2.$$

This way, the function $\psi_d(\cdot)$ minimizes the mean quadratic error between the model's outputs $\psi_d(\boldsymbol{\rho}_\ell)$ and the desired outputs $p_{\ell,d}$. As for the regularization term $\mathscr{R}(\|\psi_d\|^2_{\mathcal{H}})$, it is taken in its simplest form as $\|\psi_d\|^2_{\mathcal{H}}$. Now that we have defined the optimization problem, we can write it as follows:

$$\min_{\psi_d \in \mathcal{H}} \frac{1}{N_p}\sum_{\ell=1}^{N_p}((p_{\ell,d} - \psi_d(\boldsymbol{\rho}_\ell))^2 + \eta \|\psi_d\|^2_{\mathcal{H}}. \quad (3)$$

By injecting (2) in (3), we get the following dual optimization problem in terms of $\boldsymbol{\alpha}_{*,d}$:

$$\min_{\boldsymbol{\alpha}_{*,d}} \left(\boldsymbol{P}_{*,d} - \boldsymbol{K}\boldsymbol{\alpha}_{*,d}\right)^\top \left(\boldsymbol{P}_{*,d} - \boldsymbol{K}\boldsymbol{\alpha}_{*,d}\right) + \eta N_p \boldsymbol{\alpha}_{*,d}^\top \boldsymbol{K}\boldsymbol{\alpha}_{*,d},$$

where $\boldsymbol{K}$ is the N_p-by-N_p matrix whose (i, j)th entry is $\kappa(\boldsymbol{\rho}_i, \boldsymbol{\rho}_j)$, for $i, j \in \{1, \ldots, N_p\}$. This is a classical quadratic regression problem, whose solution is given by taking its derivative with respect to $\boldsymbol{\alpha}_{*,d}$ and setting it to zero:

$$-\boldsymbol{K}\boldsymbol{P}_{*,d} + \boldsymbol{K}^2\boldsymbol{\alpha}_{*,d} + \eta N_p \boldsymbol{K}\boldsymbol{\alpha}_{*,d} = \boldsymbol{0}^\top.$$

One can easily find the following form of the solution:

$$\boldsymbol{\alpha}_{*,d} = (\boldsymbol{K} + \eta N_p \mathrm{I})^{-1}\boldsymbol{P}_{*,d}, \quad (4)$$

where I is the N_p-by-N_p identity matrix. For an appropriate value of the regularization parameter η, the matrix between the parentheses is always nonsingular.

Equation (4) shows that the same matrix $(\boldsymbol{K} + \eta N_p \mathrm{I})$ needs to be inverted in order to estimate each coordinate. Nevertheless, it is reasonable to collect all D estimations (D being the space's dimension) in a single matrix inversion problem to reduce the computational complexity to $\mathcal{O}(N_p^3)$, by writing:

$$\boldsymbol{\alpha} = (\boldsymbol{K} + \eta N_p \mathrm{I})^{-1}\boldsymbol{P}. \quad (5)$$

Using (2) and the definition of the vector of functions $\boldsymbol{\psi}(\cdot)$, we now define a model that allows us to estimate all D coordinates at once, as follows:

$$\boldsymbol{\psi}(\cdot) = \sum_{\ell=1}^{N_p}\boldsymbol{\alpha}_{\ell,*}\kappa(\boldsymbol{\rho}_\ell, \cdot). \quad (6)$$

Now that we have considered a bias-free model, we may also consider an offset in the model as in [28], with

$$\boldsymbol{\psi}(\cdot) = \sum_{\ell=1}^{N_p}\boldsymbol{\alpha}_{\ell,*}\kappa(\boldsymbol{\rho}_\ell, \cdot) + \boldsymbol{b}, \quad (7)$$

where $\boldsymbol{b}$ is the bias parameter of size 1-by-D. We obtain the following linear equations, with $\boldsymbol{\alpha}$ and $\boldsymbol{b}$ being the unknown variables:

$$\begin{pmatrix} 0 & 1 \\ \boldsymbol{1}^\top & \boldsymbol{K} + \eta N_p \mathrm{I} \end{pmatrix} \times \begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{\alpha} \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{P} \end{pmatrix}.$$

B. Ridge Regression by Regularizing the $\boldsymbol{\alpha}_{*,d}$

In this subsection, we define the function $\psi_d(\cdot) \in \mathcal{H}$ in such a way to minimize the following regularized risk:

$$\min_{\psi_d \in \mathcal{H}} \frac{1}{N_p}\sum_{\ell=1}^{N_p}\left((p_{\ell,d} - \psi_d(\boldsymbol{\rho}_\ell)\right)^2 + \eta \left\|\boldsymbol{\alpha}_{*,d}\right\|^2. \quad (8)$$

The relationship between the ridge regression using the optimization problem in (3) and the optimization problem in (8) will be shown in the following remark. Substituting (2) into (8), the problem can be written in terms of $\boldsymbol{\alpha}_{*,d}$ as follows:

$$\min_{\boldsymbol{\alpha}_{*,d}} \left(\boldsymbol{P}_{*,d} - \boldsymbol{K}\boldsymbol{\alpha}_{*,d}\right)^\top \left(\boldsymbol{P}_{*,d} - \boldsymbol{K}\boldsymbol{\alpha}_{*,d}\right) + \eta N_p \boldsymbol{\alpha}_{*,d}^\top \boldsymbol{\alpha}_{*,d}.$$

The solution is obtained by taking the derivative with respect to $\boldsymbol{\alpha}_{*,d}$ and setting it to zero, namely:

$$-\boldsymbol{K}\boldsymbol{P}_{*,d} + \boldsymbol{K}^2\boldsymbol{\alpha}_{*,d} + \eta N_p \boldsymbol{\alpha}_{*,d} = \boldsymbol{0}^\top.$$

This leads to the following form of $\boldsymbol{\alpha}_{*,d}$:

$$\boldsymbol{\alpha}_{*,d} = (\boldsymbol{K}^2 + \eta N_p \mathrm{I})^{-1}\boldsymbol{K}\boldsymbol{P}_{*,d}.$$

By collecting the D estimations into one problem as in the previous subsection, we avoid inverting D times the matrix $(\boldsymbol{K}^2 + \eta N_p \mathrm{I})$. We then get:

$$\boldsymbol{\alpha} = (\boldsymbol{K}^2 + \eta N_p \mathrm{I})^{-1}\boldsymbol{K}\boldsymbol{P}.$$

Here, we may also consider an offset in the model as in (7). We obtain the following linear system:

$$\begin{pmatrix} 0 & \boldsymbol{1} \\ \boldsymbol{1}^\top & \boldsymbol{K}^2 + \eta N_p \mathrm{I} \end{pmatrix} \times \begin{pmatrix} \boldsymbol{b} \\ \boldsymbol{\alpha} \end{pmatrix} = \begin{pmatrix} \boldsymbol{0} \\ \boldsymbol{K}\boldsymbol{P} \end{pmatrix}.$$

REMARK 1 In this paragraph, we show how the two optimization problems in (3) and (8) are connected. In fact, both problems have the form given in (1), the first

term being identical; the difference then resides in the second term, i.e., the regularization. From (3), we have:

$$\begin{aligned}\mathscr{R}(\|\psi_d\|_{\mathcal{H}}^2) &= \|\psi_d\|_{\mathcal{H}}^2 \\ &= \boldsymbol{\alpha}_{*,d}^\top \boldsymbol{K}\boldsymbol{\alpha}_{*,d}.\end{aligned}$$

Using Rayleigh's inequalities, we write:

$$\lambda_{\min}\boldsymbol{\alpha}_{*,d}^\top\boldsymbol{\alpha}_{*,d} \leq \boldsymbol{\alpha}_{*,d}^\top \boldsymbol{K}\boldsymbol{\alpha}_{*,d} \leq \lambda_{\max}\boldsymbol{\alpha}_{*,d}^\top\boldsymbol{\alpha}_{*,d},$$

where $\lambda_{\min}$ and $\lambda_{\max}$ are, respectively, the smallest and the largest eigenvalues of K. Therefore, we have:

$$\lambda_{\min}\|\boldsymbol{\alpha}_{*,d}\|^2 \leq \|\psi_d\|_{\mathcal{H}}^2 \leq \lambda_{\max}\|\boldsymbol{\alpha}_{*,d}\|^2.$$

By considering these inequalities, one can see that minimizing $\|\boldsymbol{\alpha}_{*,d}\|^2$ results in minimizing $\|\psi_d\|_{\mathcal{H}}^2$ and vice versa. In fact, on the one hand, the optimization problem in (8) constrains the norm $\|\boldsymbol{\alpha}_{*,d}\|^2$ which leads to an upper bound on $\|\psi_d\|_{\mathcal{H}}^2$ because $\|\psi_d\|_{\mathcal{H}}^2 \leq \lambda_{\max}\|\boldsymbol{\alpha}_{*,d}\|^2$. On the other hand, in (3), the norm $\|\psi_d\|_{\mathcal{H}}^2$ is constrained, providing an upper bound on $\|\boldsymbol{\alpha}_{*,d}\|^2$ as well because $\lambda_{\min}\|\boldsymbol{\alpha}_{*,d}\|^2 \leq \|\psi_d\|_{\mathcal{H}}^2$.

C. SVR

In this subsection, SVR and the ϵ-insensitive loss function, introduced by Vapnik in [24], are used to find the model $\psi_d(\cdot)$. In ϵ-support vector regression (ϵ-SVR), the goal is to find $\psi_d(\boldsymbol{\rho})$ that has at most ε deviation from the target $p_{\ell,d}$ for all the training data, and that is, at the same time, as flat as possible. In other words, errors are accepted as long as they are less than ϵ. For this reason, the optimization problem is taken as in (1), with the cost term given by:

$$\begin{aligned}&\mathscr{C}((p_{1,d}, \psi_d(\boldsymbol{\rho}_1)), \ldots, (p_{N_p,d}, \psi_d(\boldsymbol{\rho}_{N_p}))) \\ &\quad = \frac{1}{2N_p}\sum_{\ell=1}^{N_p}\max(0, |p_{\ell,d} - \psi_d(\boldsymbol{\rho}_\ell)| - \epsilon).\end{aligned}$$

As for the regularization term, it is set to $\mathscr{R}(\|\psi_d\|_{\mathcal{H}}^2) = \|\psi_d\|_{\mathcal{H}}^2$. The two quantities $\eta > 0$ and $\epsilon \geq 0$ are tunable parameters that determine the trade-off between the regularization and the fit to the training set.

The optimization problem can be solved more easily in its dual formulation; for details, refer to [29, 30]. We get the following dual optimization problem in terms of the Lagrange multipliers $\boldsymbol{\alpha}_{*,d}$ and $\tilde{\boldsymbol{\alpha}}_{*,d}$:

$$\begin{aligned}\max_{\boldsymbol{\alpha}_{*,d},\,\tilde{\boldsymbol{\alpha}}_{*,d}} &\sum_{\ell=1}^{N_p}\tilde{\alpha}_{\ell,d}(p_{\ell,d} - \epsilon) - \alpha_{\ell,d}(p_{\ell,d} + \epsilon) \\ &-\frac{1}{2}\sum_{\ell=1}^{N_p}\sum_{j=1}^{N_p}(\tilde{\alpha}_{\ell,d} - \alpha_{\ell,d})(\tilde{\alpha}_{j,d} - \alpha_{j,d})\kappa(\boldsymbol{\rho}_\ell, \boldsymbol{\rho}_j),\end{aligned}$$

with the following constraints

$$\begin{aligned}&0 \leq \alpha_{\ell,d}, \tilde{\alpha}_{\ell,d} \leq \frac{1}{2\eta N_p}, \ell \in 1, \ldots, N_p \\ &\sum_{\ell=1}^{N_p}(\alpha_{\ell,d} - \tilde{\alpha}_{\ell,d}) = 0. \qquad (9)\end{aligned}$$

Solving the dual optimization problem with the constraints (9) determines the Lagrange multipliers, and therefore the regression function is given by:

$$\psi_d(\cdot) = \sum_{\ell=1}^{N_p}(\alpha_{\ell,d} - \tilde{\alpha}_{\ell,d})\kappa(\boldsymbol{\rho}_\ell, \cdot) + b. \qquad (10)$$

The offset term b in (10) is obtained after solving the optimization problem by exploiting the so-called Karush-Kuhn-Tucker conditions [31, 32]. See [29] for methods to compute the offset b.

Finally, the dual problem with the constraints in (9) can be written as,

$$\begin{aligned}\max_{\boldsymbol{\alpha}_{*,d},\tilde{\boldsymbol{\alpha}}_{*,d}} &\; 2\begin{pmatrix}-\epsilon\mathbf{1}^\top + \boldsymbol{P}_{*,d} \\ -\epsilon\mathbf{1}^\top - \boldsymbol{P}_{*,d}\end{pmatrix}^\top\begin{pmatrix}\boldsymbol{\alpha}_{*,d} \\ \tilde{\boldsymbol{\alpha}}_{*,d}\end{pmatrix} \\ &-\begin{pmatrix}\boldsymbol{\alpha}_{*,d} \\ \tilde{\boldsymbol{\alpha}}_{*,d}\end{pmatrix}^\top\begin{pmatrix}\boldsymbol{K} & -\boldsymbol{K} \\ -\boldsymbol{K} & \boldsymbol{K}\end{pmatrix}\begin{pmatrix}\boldsymbol{\alpha}_{*,d} \\ \tilde{\boldsymbol{\alpha}}_{*,d}\end{pmatrix}\end{aligned}$$

subject to

$$\begin{aligned}&\begin{pmatrix}\mathbf{1} & -\mathbf{1}\end{pmatrix}\begin{pmatrix}\boldsymbol{\alpha}_{*,d} \\ \tilde{\boldsymbol{\alpha}}_{*,d}\end{pmatrix} = 0 \\ &\boldsymbol{\alpha}_{*,d}, \tilde{\boldsymbol{\alpha}}_{*,d} \in \left[0, \tfrac{1}{2\eta N_p}\right]^{N_p}.\end{aligned}$$

We now have a quadratic programming problem, whose solution is found using an off-the-shelf optimization technique. For instance, one can use the Matlab function quadprog.

D. vo-RLS

In the previous algorithms, D optimization problems are set separately to define D univariate models $\psi_d(\cdot)$, one per coordinate. In this subsection, a single optimization problem is considered to estimate simultaneously all D coordinates. To this end, a vector-output model $\boldsymbol{\psi}(\cdot)$ is determined by exploring multitask learning. Recently, such learning has become essential for solving a variety of practical problems that necessitate the estimation of vector-output functions. For instance, in [33, 34], Micchelli et al. presented a framework to study the problem of learning vector-output functions, and the goal was to extend the single-task kernel methods, which have been successfully used in recent years, to multitask learning. In [35], the authors showed that the representer theorem could still be used in the case of multitask learning. In this subsection, we take advantage of multitask learning by using the vo-RLS algorithm [25] to estimate all D coordinates at once. Therefore, we now determine the function $\boldsymbol{\psi}(\cdot)$, whose output is a position vector of dimension D, without having to determine the set of functions $\psi_d(\cdot)$.

In multitask learning, $\boldsymbol{\psi}(\cdot)$ takes the form:

$$\boldsymbol{\psi}(\cdot) = \sum_{\ell=1}^{N_p} \beta_\ell \boldsymbol{P}_{\ell,*} \kappa(\boldsymbol{\rho}_\ell, \cdot),$$

where β_ℓ, $\ell \in \{1, \ldots, N_p\}$, are parameters to be defined. We then consider the following optimization problem:

$$\min_{\boldsymbol{\psi}} \frac{1}{N_p} \sum_{\ell=1}^{N_p} \left\| \boldsymbol{P}_{\ell,*} - \boldsymbol{\psi}(\boldsymbol{\rho}_\ell) \right\|^2 + \eta \left\| \boldsymbol{\beta} \right\|^2, \qquad (11)$$

where $\boldsymbol{\beta} = \left(\beta_1 \ldots \beta_{N_p}\right)^\top$. By substituting the expression of $\boldsymbol{\psi}(\cdot)$ in the above optimization problem, we get, in matrix form, the following problem formulation:

$$\min_{\boldsymbol{\beta}} \operatorname{tr}(\boldsymbol{P}\boldsymbol{P}^\top) - 2v^\top \boldsymbol{\beta} + \boldsymbol{\beta}^\top \boldsymbol{G} \boldsymbol{\beta} + \eta N_p \boldsymbol{\beta}^\top \boldsymbol{\beta}, \qquad (12)$$

where $\operatorname{tr}(\cdot)$ is the matrix trace operator, $\boldsymbol{G}$ is the N_p-by-N_p matrix whose (j, k)th entry is

$$\boldsymbol{P}_{j,*} \boldsymbol{P}_{k,*}^\top \sum_{i=1}^{N_p} \kappa(\boldsymbol{\rho}_j, \boldsymbol{\rho}_i) \kappa(\boldsymbol{\rho}_k, \boldsymbol{\rho}_i),$$

and v is the N_p-by-1 vector whose jth entry is

$$\sum_{k=1}^{N_p} \boldsymbol{P}_{j,*} \boldsymbol{P}_{k,*}^\top \kappa(\boldsymbol{\rho}_j, \boldsymbol{\rho}_k).$$

By taking the gradient of the objective function in (12) with respect to $\boldsymbol{\beta}$, namely, $-v + \boldsymbol{G}\boldsymbol{\beta} + \eta N_p \boldsymbol{\beta}$, and setting it to zero, we obtain the final solution:

$$\boldsymbol{\beta} = (\boldsymbol{G} + \eta N_p \mathrm{I})^{-1} \boldsymbol{v}.$$

Having calculated the vector $\boldsymbol{\beta}$, it is now possible to determine the needed model $\boldsymbol{\psi}(\cdot)$. It is worth noting that N_p unknown variables need to be computed in this algorithm, while $N_p \times D$ unknown variables need to be found for the previous algorithms.

IV. PRACTICAL SIMULATIONS AND RESULTS

In this section, we highlight the performance of our method applied for different learning algorithms. To this end, we show the results obtained when using each of the learning algorithms described in this paper for different scenarios. In the first subsection, the proposed method is tested on simulated data, and the results are compared for different machine-learning techniques. In the second subsection, we test the performance of our method using real data gathered in a 10 m $\times$ 10 m real indoor environment [36]. In the final subsection, the results obtained with the proposed method are compared to the ones obtained when performing localization using connectivity information as well as the WKNN algorithm and the method that uses kernelized calculation proposed in [20].

It is important to mention that in all of the following sections, the choice of the regularization parameter η and the kernel parameters are performed using the cross-validation technique. This approach is a statistical method that consists of dividing data into two segments: one for training the model and the other one for validating it [37]. The k-fold cross-validation, which is the basic form of cross-validation, is used here; it consists of partitioning the data into k roughly equally sized folds. Subsequently, k iterations of training and validation are performed such that, within each iteration, $k-1$ folds are used for learning and the remaining one for validation. In each iteration, the error on the validation set is computed for different values of the tuning parameters. Then, the values for the parameters that give a minimum error for all iterations are retained.

A. Evaluation of the Method on Simulated Data

We consider a 100 m $\times$ 100 m area and generate 16 static anchors and 100 offline positions uniformly distributed over the area. The RSSI values needed to construct the database of N_p pairs $(\boldsymbol{\rho}_\ell, \boldsymbol{p}_\ell)$, where $\ell \in \{1, \ldots, N_p\}$ ($N_p = 100$), are generated using the well-known Okumura-Hata model [38] given by:

$$\rho_{\boldsymbol{a}_i, \boldsymbol{p}_\ell} = \rho_0 - 10 n_P \log_{10} \left\| \boldsymbol{a}_i - \boldsymbol{p}_\ell \right\| + \varepsilon_{i,\ell}, \qquad (13)$$

where $\rho_{\boldsymbol{\alpha}_i, \boldsymbol{p}_\ell}$ (in dB m) is the power received from the anchor $\boldsymbol{a}_i$ by the node at position $\boldsymbol{p}_\ell$, i.e., the ith entry of the vector $\boldsymbol{\rho}_\ell$, ρ_0 is the initial power (in dB m) set to 150, n_P is the path-loss exponent set to 4, $\|\boldsymbol{\alpha}_i - \boldsymbol{p}_\ell\|$ is the Euclidian distance between the position $\boldsymbol{p}_\ell$ of the considered node and the anchor position $\boldsymbol{a}_i$, and $\varepsilon_{i,\ell}$ is the noise affecting the RSSI measures. We also generate a trajectory and calculate the RSSI values collected by the moving node using this RSSI model. In this paper, we consider the Gaussian kernel given by:

$$\kappa(\boldsymbol{\rho}_s, \boldsymbol{\rho}_{s'}) = \exp\left(\frac{-\left\| \boldsymbol{\rho}_s - \boldsymbol{\rho}_{s'} \right\|^2}{2\sigma^2} \right),$$

where σ is the bandwidth of the Gaussian kernel. This quantity together with the regularization parameter η control the degree of smoothness, noise tolerance, and generalization of the solution. The cross-validation technique allows us to find the proper values for the parameters η and σ, considering $\eta N_p = 2^s$ with $s \in \{-20, -19, \ldots, -1\}$ and $\sigma = 2^{s'}$ with $s' \in \{1, 2, \cdots, 10\}$. Fig. 1 shows the generated trajectory and the estimated one using noiseless RSSIs for different machine-learning algorithms (in the case of a bias-free model), using a 10-fold cross-validation to find the optimal σ and η. The estimation error, measured by the root mean squared distance between the exact positions and the estimated ones, as well as the parameters σ and η, are shown in Table III. We notice that the ridge regression and the vo-RLS yield close results, and that the best result is obtained when using the ridge regression from Section IIIA; we obtain an estimation error of 0.17 m for $\sigma = 2^7$ and $\eta N_p = 2^{-20}$. To evaluate the robustness of our method against noise, we add a zero-mean Gaussian white noise of standard deviation 1 dB to the RSSI values; the results are shown in Table IV for the different learning algorithms.

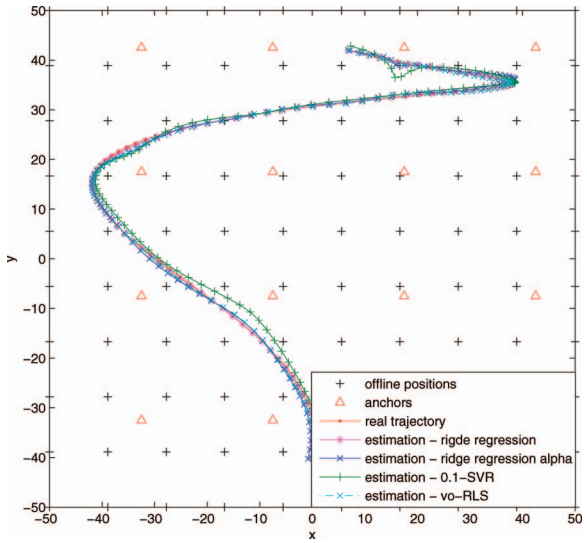

Fig. 1. Estimation of trajectory in absence of noise: uniform distribution.

TABLE III
Estimation Error (in Meters) for Different Techniques: Simulated Data without Noise

Learning algorithm	η N_p	σ	Error
Ridge regression	2^{-20}	2^{7}	0.17
Ridge regression with bias	2^{-20}	2^{7}	0.17
Ridge regression (α)	2^{-18}	2^{5}	0.59
Ridge regression (α) with bias	2^{-20}	2^{5}	0.70
0-SVR	2^{-8}	2^{6}	2.22
0.1-SVR	2^{-10}	2^{5}	1.10
0.25-SVR	2^{-8}	2^{6}	1.72
0.5-SVR	2^{-10}	2^{5}	3.13
1-SVR	2^{-18}	2^{5}	1.37
2-SVR	2^{-9}	2^{6}	2.23
vo-RLS	2^{-13}	2^{6}	0.45

TABLE IV
Estimation Error (in Meters) for Different Techniques: Simulated Data with Noise

Learning algorithm	ηN_p	σ	Error
Ridge regression	2^{-7}	2^{6}	1.88
Ridge regression with bias	2^{-7}	2^{6}	1.87
Ridge regression (α)	2^{-14}	2^{6}	1.92
Ridge regression (α) with bias	2^{-16}	2^{6}	1.95
0-SVR	2^{-8}	2^{6}	2.46
0.1-SVR	2^{-8}	2^{6}	2.69
0.25-SVR	2^{-7}	2^{6}	2.67
0.5-SVR	2^{-8}	2^{6}	2.35
1-SVR	2^{-8}	2^{6}	2.42
2-SVR	2^{-9}	2^{6}	2.36
vo-RLS	2^{-10}	2^{7}	2.09

We notice that the best estimations are also obtained when using the ridge regression and the vo-RLS.

As for the evaluation of the time complexity of our method when using each of the aforementioned learning algorithms, we measure the elapsed time for the training phase for each one of them. Simulations are run on version 7.10.0.499 of Matlab on a Dell laptop with Windows 7 and Intel® Core™ i7 CPU. For all the cases of the ridge regression, the elapsed time for the training phase is around 8 ms; it is around 25 ms for the vo-RLS and around 15 s for the ϵ-SVR. It is worth noting that finding the optimal values for the tuning parameters η and σ is the most complex part in terms of computations and time because one has to repeat the training phase depending on the number of folds considered and on the range of the variables. Indeed, finding the optimal values of η and σ, using a 10-fold cross-validation in our case, takes around 16 s for the ridge regression, around 50 s for the vo-RLS, and more than 5 h for the ϵ-SVR. Hence, it is easy to see that the ϵ-SVR has the worst performance when it comes to studying time complexity and also estimation error. We note that, in the online phase, the elapsed time to estimate a trajectory point is around 0.3 ms when using the ridge regression, around 1.7 ms with the vo-RLS, and around 1.5 ms with the ϵ-SVR. As for the evaluation of the memory consumption of the method, we consider the number of necessary variables in the online phase. For the bias-free ridge regression, $N_p(N_a + D) + 1$ variables are needed, whereas $N_p(N_a + D) + D + 1$ variables are needed for the ridge regression with bias. Finally, N_p $(N_a + D + 1) + 1$ variables are needed for the vo-RLS, and N_p $(N_a + D) + D + 1$ for the ϵ-SVR. One can see that the bias-free ridge regression necessitates the smallest number of variables in the online phase, while the highest number of variables needed is for the vo-RLS because N_p is always higher than D.

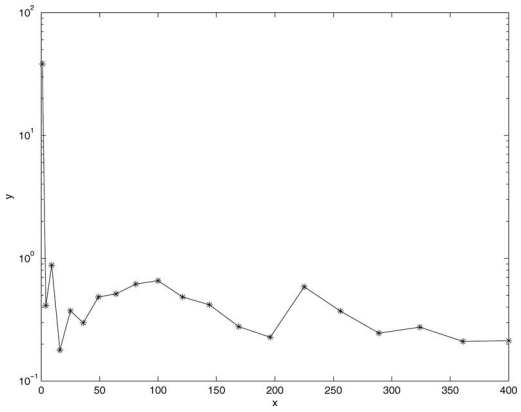

Fig. 2. Estimation error as function of number of anchors.

We now study the effect of the number of anchors and the number of offline positions on the performance of the method. We consider noiseless data and the same scenario as the one used in Fig. 1, but we vary the number of anchors ($N_a = 1^2, \ldots, 20^2$) while keeping a fixed number for the offline positions ($N_p = 100$). In this paragraph, we consider a bias-free model using the ridge regression problem of Section IIIA with noiseless data because it yields the best results in this case as shown before. Fig. 2 shows the evolution of the estimation error in terms of the number of anchors. For the results in Fig. 3, we consider the same settings, and fix the number of anchors to 16, but

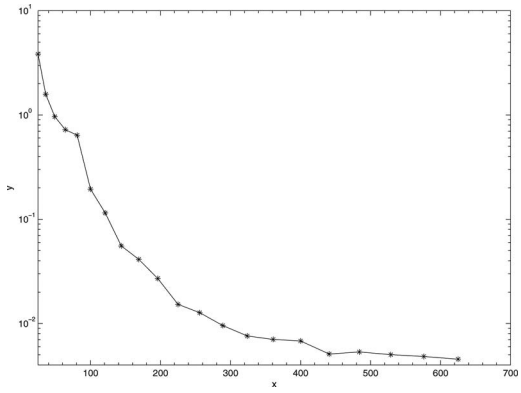

Fig. 3. Estimation error as function of number of offline positions.

TABLE V
Estimation Error (in Meters) for Different Techniques, Random Distribution, and Noiseless Simulated Data

Learning algorithm	$\eta\ N_p$	σ	σ_{MSE}	Error
Ridge regression	2×10^{-8}	63.36	0.23	0.33
Ridge regression with bias	2.08×10^{-8}	62.72	0.40	0.45
Ridge regression (α)	1.30×10^{-8}	39.68	0.61	1.13
Ridge regression (α) with bias	3.57×10^{-8}	40.96	0.52	1.14
0.1-SVR	2^{-16}	2^4	-	2.62
vo-RLS	1.46×10^{-3}	72.96	0.27	1.25

vary the number of offline positions, $N_p = 5^2, \ldots, 25^2$. By comparing the obtained results, one can notice that the changes in the number of anchors do not affect the estimation error as much as the changes in the number of offline positions do. Indeed, increasing the number of offline positions allows a better coverage and a better knowledge of the environment, which explains the improvement in the results. For instance, for $N_p = 25^2 = 625$ and $N_a = 16$, we get a low estimation error of 0.0045 m, but with a significant increase in the algorithm's complexity. Therefore, depending on the practical system constraints, a trade-off should be found between the algorithm's accuracy and the computational load.

We now consider a random distribution of the 16 anchors and the 100 offline positions instead of the uniform distribution. We repeat the experiment 50 times for the ridge regression algorithms and only once for the 0.1-SVR. The 0.1-SVR is chosen for comparison because it yields one of the lowest estimation errors in the noiseless setup (Table III). However, the experiment is done only one time because of the huge time complexity of this algorithm. The mean estimation error, as well as the mean values of σ and η, are shown in Table V in the case of noiseless data and in Table VI in the case of noisy data (zero-mean Gaussian white noise with standard deviation 1 dB). In both tables, σ_{MSE} is the standard deviation of the mean estimation error. Fig. 4 shows the generated trajectory and the estimated one using noiseless RSSIs and a random distribution. Compared to the results obtained when having uniform distributions of the anchors and offline positions, one can see that the estimation error

TABLE VI
Estimation Error (in Meters) for Different Techniques, Random Distribution, and Simulated Data With Noise

Learning algorithm	$\eta\ N_p$	σ	σ_{MSE}	Error
Ridge regression	7.61×10^{-5}	89.28	1.09	2.71
Ridge regression with bias	1×10^{-4}	72.00	0.82	2.55
Ridge regression (α)	5.40×10^{-6}	48.32	1.63	2.87
Ridge regression (α) with bias	2.60×10^{-6}	47.36	0.57	2.55
0.1-SVR	2^{-6}	2^6	–	2.83
vo-RLS	2.09×10^{-2}	85.76	0.82	3.46

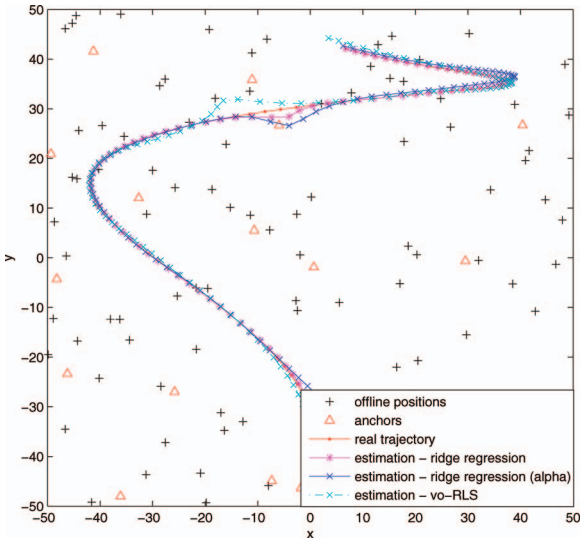

Fig. 4. Estimation of trajectory in absence of noise: random distribution.

increases with the use of random distributions. This can be explained by the fact that a uniform distribution allows a better coverage of the region of interest, while a random distribution does not always guarantee a good coverage of the region. Nevertheless, the results are still satisfactory, and random distributions can still be used for accurate localization when uniform distributions are not applicable.

B. Evaluation of the Method on Real Data

In this subsection, we study the performance of the method in the case of real collected data for different machine-learning algorithms. The set of collected measurements used in this study are available from [36]. The measurements are performed in a room of approximately 10 m $\times$ 10 m, where 48 uniformly distributed EyesIFX nodes are deployed. Furniture and people in the room cause multipath interferences affecting the collected RSSI values. Five nodes are chosen to be anchors, leaving us with 43 nodes to use as offline positions. However, to get better results, we generate 57 additional offline positions in order to get a total of 100 offline positions. This is done using a weighting function that relies on the Euclidian distance between the existing points and the new ones. We generate the trajectory in the same manner and apply the proposed method to estimate

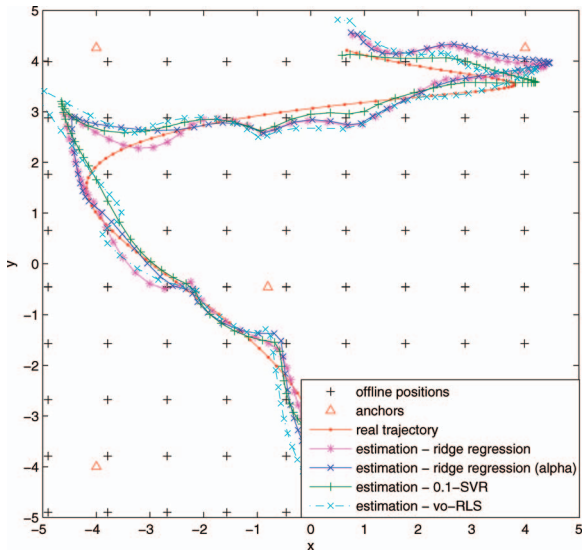

Fig. 5. Estimation of trajectory: real data.

TABLE VII
Estimation Error (in Meters) for Different Techniques and Real Data

Learning algorithm	$\eta\, N_p$	σ	Error
Ridge regression	2^{-14}	2^4	0.34
Ridge regression with bias	2^{-14}	2^4	0.35
Ridge regression (α)	2^{-14}	2^3	0.42
Ridge regression (α) with bias	2^{-20}	2^4	0.56
0.1-SVR	2^{-3}	2^3	0.41
0.25-SVR	2^{-3}	2^3	0.53
0.5-SVR	2^{-3}	2^3	0.62
vo-RLS	2^{-18}	2^3	0.41

the node's position as shown in Fig. 5. Table VII shows the estimation error for different learning algorithms. The lowest estimation error is again obtained with the ridge regression of Section IIIA. The other results are also satisfactory. However, the ε-SVR does not allow a rapid localization because of its high complexity in terms of computation time as explained previously.

C. Comparison to Other Localization Methods

In this subsection, we provide a comparison of our localization method with respect to two well-known localization techniques: localization by connectivity and localization using the WKNN algorithm. Comparisons are made toward the results obtained with the ridge regression of Section IIIA because it yields the best results as shown previously.

Localization by connectivity [12] is a simple technique that only uses neighboring anchors information and RSSI measures. All anchors are considered to have the same transmission range, which is an ideal disk with an equal transmission radius. A node's position is given by finding the intersection of all the range disks of the anchors detected by this node. However, this technique can only provide a coarse-grained estimate of each node's location. In addition, the localization error is highly dependent on the node density of the network, the number of anchors,

TABLE VIII
Comparison of the Estimation Error of the Proposed Method to the Connectivity-Based One With Noiseless Data

	No. of anchors (N_a)				
	16	64	81	100	196
Connectivity	7.44	3.20	3.02	2.61	1.92
Proposed method	0.17	0.51	0.61	0.65	0.22

and the network topology [39]. The results in Table VIII shows the strong influence of the number of anchors on the connectivity-based method, while our proposed method yields good results in all cases. Moreover, our method outperforms the connectivity-based method in all cases, with a decrease of the estimation error of more than 75%. In terms of computation time, the connectivity-based method does not need offline computations. However, if we compare the online computation time in the case of 16 anchors, finding the position of a trajectory point with the ridge regression takes about 0.3 ms as we already stated in Subsection IVA, whereas the connectivity algorithm takes around 3.8 ms. Furthermore, when estimating the whole trajectory, that is 100 positions, the time needed for the connectivity is even higher than the total complexity of the online and offline phases together, using the ridge regression or the vo-RLS. For all these reasons, the ridge regression method seems to be the most convenient one to the considered context because it globally presents the best performance with the lowest overall complexity. Moreover, when 196 anchors are deployed, the connectivity algorithm takes about 150 ms to estimate a point of the trajectory, while the ridge regression only needs 0.5 ms. Therefore, we conclude that the ridge regression method takes less time for the trajectory estimation, compared to connectivity, once the model has been defined.

The WKNN algorithm [19] relies on the Euclidean distances between the RSSI values received by the node to be localized and the fingerprints in the database to give an estimation of the node's position. Let $\delta_\ell = \|\boldsymbol{\rho} - \boldsymbol{\rho}_\ell\|$ be the Euclidean distance between the RSSI vector $\boldsymbol{\rho}$ of the mobile node and $\boldsymbol{\rho}_\ell$ of the database where $\ell \in \{1, \ldots, N_p\}$. Also, let I be the set of indices of $\boldsymbol{\rho}_\ell$ yielding the K smallest distances (i.e., K nearest neighbors) δ_ℓ. Then, the node's position is estimated by:

$$\hat{\boldsymbol{x}} = \sum_{k \in I} w_k \boldsymbol{p}_k,$$

where $\boldsymbol{p}_k$ is one of the nearest offline positions neighboring the node and w_k is a normalized weight factor depending on the kth minimal distance. The weight factor is important in contributing to the position accuracy; in fact, nearer neighbors should have higher weights, in order to contribute more to the position coordinates compared to farther neighbors (i.e. with lower weights). Therefore, weights should be chosen in a decreasing manner with respect to the distances. Weight values often considered in

TABLE IX
List of Selected Weight Factors w_k

Type	Expression of w_k
A	$1/K$
B	$\frac{1/\delta_k}{\sum_{v\in I} 1/\delta_v}$
C	$\frac{1/\delta_k^2}{\sum_{v\in I} 1/\delta_v^2}$
D	$\frac{1/\delta_k^3}{\sum_{v\in I} 1/\delta_v^3}$
E	$\frac{\exp(-\delta_k)}{\sum_{v\in I}\exp(-\delta_v)}$
F	$\frac{\frac{1}{(\sqrt{2\pi}\varsigma)^\tau}\exp(-\frac{\delta_k^2}{2\varsigma^2})}{\sum_{v\in S}\frac{1}{(\sqrt{2\pi}\varsigma)^\tau}\exp(-\frac{\delta_v^2}{2\varsigma^2})}$

TABLE X
Estimation Error (in Meters) for Different Weight Models: Simulated Data Without Noise

WKNN					KBP	Proposed Method
A	B	C	D	E	F	
4.36	2.74	2.13	2.34	3.61	4.66	0.17

TABLE XI
Estimation Error (in Meters) for Different Weight Models: Simulated Data With Noise

WKNN					KBP	Proposed Method
A	B	C	D	E	F	
4.48	2.42	2.39	2.41	3.67	4.98	1.87

TABLE XII
Estimation Error (in Meters) for Different Weight Models: Real Data

WKNN					KBP	Proposed Method
A	B	C	D	E	F	
0.72	0.64	0.61	0.58	0.62	0.75	0.35

the literature are listed in Table IX and assigned as types A–E. Table X shows the estimation error using WKNN for different weight functions and the proposed method using ridge regression with bias, in the case of noiseless simulated data. On the other hand, in Table XI, the results shown are obtained in the case of RSSI with additive zero-mean Gaussian white noise of standard deviation 1 dB. In Table XII, estimation errors are presented for the case of real data. Values of K are found using the 10-fold cross validation for $K \in 1, 2, \ldots, 15$. We notice that in all the presented scenarios, the proposed method outperforms the WKNN method. Also note that the WKNN algorithm takes about 0.2 ms to estimate a trajectory point. It outperforms the ridge regression in terms of time complexity by little, but with less estimation accuracy.

As for the localization method proposed in [20], it also uses weighted combinations of a set of fingerprints from the training database. However, instead of using the K nearest neighbors, the authors present a new method for choosing this set of fingerprints. Indeed, they propose a spatially localized averaging strategy wherein anchors coverage information is used to retain a subset of spatially relevant offline positions. This is done based on the premise that offline positions close to the node at a given step are covered by a similar set of anchors. They also propose a real-time anchor selection algorithm, where they first select the anchors yielding the strongest RSSIs. Then, the correlation between RSSIs is considered in order to obtain a representation with minimal redundancy, reducing again the number of anchors. It is worth noting that in the method in [20], many samples of RSSI are collected for each offline position. However, since in our simulations only one RSSI vector per offline position is considered, the number of samples becomes equal to 1. Then, the weight used in [20] is given by F in Table IX, where S is the set of fingerprints obtained after the spatial filtering, τ is the number of selected anchors, and δ_k is the Euclidean distance between the RSSI vectors of the node and the considered offline position, while only taking into account the RSSI components corresponding to the selected anchors. The parameter ς is chosen according to the authors' definition. The anchor selection is performed on the five strongest anchors, then three of these anchors are chosen using the redundant information. The smallest estimation errors obtained for the different proposed scenarios are given in Table X–Table XII. One can see that our proposed method clearly outperforms the one in [20]. As for the time complexity of the latter, we found that the elapsed time for the estimation of a trajectory point is about 1.9 ms, which is much higher than the ridge regression execution time.

V. CONCLUSION

In this paper, we proposed a new localization method using radio-location fingerprinting and kernel-based machine learning. Different machine-learning algorithms have been applied, namely, ridge regression or LS SVM, vo-RLS, and SVR. Simulation results show that the proposed framework allows accurate localization in both cases of simulated and real data. Moreover, our method outperforms other localization techniques such as the WKNN algorithm, for different weight models, and localization by connectivity, for different numbers of anchors. Future work will handle further improvements of this method by correcting the estimated trajectory using additional information such as past known location information and acceleration. Improvements may also include the introduction of a decentralized version of the proposed method, making it more robust to failure in imperfect environments. The use of multiple kernels in the learning process may also be considered.

REFERENCES

[1] Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. A survey on sensor networks. *IEEE Communications Magazine*, **40**, 8 (2002), 102–114.

[2] Gomez, C., and Paradells, J.
Wireless home automation networks: A survey of architectures and technologies.
Communications Magazine, IEEE, **48**, 6 (2010), 92–101.

[3] De Vito, S., and Fattoruso, G
Wireless chemical sensor networks for air quality monitoring.
In *14th International Meeting on Chemical Sensors—IMCS 2012*, May 2012, 641–644.

[4] Salvadori, F., De Campos, M., Sausen, P., De Camargo, R., Gehrke, C., Rech, C., Spohn, M., and Oliveira, A.
Monitoring in industrial systems using wireless sensor network with dynamic power management.
IEEE Transactions on Instrumentation and Measurement, **58**, 9 (2009), 3104–3111.

[5] Lee, S. H., Lee, S., Song, H., and Lee, H. S.,
Wireless sensor network design for tactical military applications: Remote large-scale environments.
In *Military Communications Conference*, Boston, MA, Oct. 2009, 1–7.

[6] Honeine, P., Mourad, F., Kallas, M., Snoussi, H., Amoud, H., and Francis, C.
Wireless sensor networks in biomedical: body area networks.
In *Proceedings of the 7th International Workshop on Systems, Signal Processing and their Applications*, Algeria, May 9–11, 2011.

[7] Okello, N., Fletcher, F., Musicki, D., and Ristic, B.
Comparison of recursive algorithms for emitter localisation using TDoA measurements from a pair of uavs.
IEEE Transactions on Aerospace and Electronic Systems, **47**, 3 (2011), 1723–1732.

[8] Rong, P., and Sichitiu, M.
Angle of arrival localization for wireless sensor networks.
In *3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks. SECON*, **1**, Sep. 2006, 374–382.

[9] Gholami, M., Vaghefi, R., and Strom, E.
Rss-based sensor localization in the presence of unknown channel parameters.
IEEE Transactions on Signal Processing, **61**, 15 (2013), 3752–3759.

[10] Esteves, J., Carvalho, A., and Couto, C.
Generalized geometric triangulation algorithm for mobile robot absolute self-localization.
In *IEEE International Symposium on Industrial Electronics, ISIE '03*, **1**, 2003, 346–351.

[11] Manolakis, D.
Efficient solution and performance analysis of 3-D position estimation by trilateration.
IEEE Transactions on Aerospace and Electronic Systems, **32**, 4 (1996), 1239–1248.

[12] Shang, Y., Ruml, W., Zhang, Y., and Fromherz, M. P. J.
Localization from mere connectivity.
MobiHoc 03: Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking and Computing, 2003, 201–212.

[13] Ozdemir, O., Niu, R., and Varshney, P.
Tracking in wireless sensor networks using particle filtering: Physical layer considerations.
IEEE Transactions on Signal Processing, **57**, 5 (2009), 1987–1999.

[14] Mourad, F., Snoussi, H., Abdallah, F., and Richard, C.
Anchor-based localization via interval analysis for mobile ad-hoc sensor networks.
IEEE Transactions on Signal Processing, **57**, 8 (2009), 3226–3239.

[15] Mourad, F., Snoussi, H., and Richard, C.
Interval-based localization using RSSI comparison in MANETs.
IEEE Transactions on Aerospace and Electronic Systems, **47**, 4 (2011), 2897–2910.

[16] Mourad, F., Honeine, P., and Snoussi, H.
Polar-interval-based localization in mobile sensor networks.
IEEE Transactions on Aerospace and Electronic Systems, **49**, 4 (2013), 2310–2322.

[17] Teng, J., Snoussi, H., and Richard, C.
Decentralized variational filtering for target tracking in binary sensor networks.
IEEE Transactions on Mobile Computing, **9**, 10 (2010), 1465–1477.

[18] Lin, T.-N., and Lin, P.-C.
Performance comparison of indoor positioning techniques based on location fingerprinting in wireless networks.
In *International Conference on Wireless Networks, Communications and Mobile Computing*, **2**, 2005, 1569–1574.

[19] Koyuncu, H., and Yang, S. H.
A 2D positioning system using WSNs in indoor environment.
International Journal of Electrical and Computer Sciences IJECS-IJENS, **11**, 3 (2011), 70–77.

[20] Kushki, A., Plataniotis, K., and Venetsanopoulos, A.
Kernel-based positioning in wireless local area networks.
IEEE Transactions on Mobile Computing, **6**, 6 (2007), 689–705.

[21] Wu, Z.-L., Li, C.-H., Ng, J.-Y., and Leung, K.
Location estimation via support vector regression.
IEEE Transactions on Mobile Computing, **6**, 3 (2007), 311–321.

[22] Aronszajn, N.
Theory of reproducing kernels.
Transactions of the American Mathematical Society, **68**, 3 (1950), 337–404.

[23] Kimeldorf, G., and Wahba, G.
Some results on Tchebycheffian spline functions.
Journal of Mathematical Analysis and Applications, **33**, 1 (1971), 82–95.

[24] Vapnik, V. N.
The nature of statistical learning theory.
New York, NY: Springer-Verlag, 1995.

[25] Honeine, P., Noumir, Z., and Richard, C.
Multiclass classification machines with the complexity of a single binary classifier.
Signal Processing, **93**, 5 (May 2013), 1013–1026.

[26] Mahfouz, S., Mourad-Chehade, F., Honeine, P., Snoussi, H., and Farah, J.
Kernel-based localization using fingerprinting in wireless sensor networks.
In *IEEE 14th Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2013, 744–748.

[27] Schölkopf, B., Herbrich, R., and Smola, A. J.
A generalized representer theorem.
In *Proceedings of the 14th Annual Conference on Computational Learning Theory and 5th European Conference on Computational Learning Theory*, London, United Kingdom, 2001, 416–426.

[28] Suykens, J. A. K., and Vandewalle, J.
Least squares support vector machine classifiers.
Neural Processing Letters, **9**, 3 (Jun. 1999), 293–300.

[29] Smola, A. J., and Schölkopf, B.
A tutorial on support vector regression.
Statistics and Computing, **14**, 3 (Aug. 2004), 199–222.

[30] McCormick, G. P.
Nonlinear programming: Theory, algorithms, and applications.
New York, NY: John Wiley and Sons, 1983.

[31] Karush, W.
Minima of functions of several variables with inequalities as side conditions.
Master's thesis, University of Chicago, 1939.

[32] Kuhn, H., and Tucker, A.
Nonlinear programming.
In *Proceedings of the Second Berkeley Symposium on Mathematical Statistics and Probability*, 1951, 481–492.

[33] Micchelli, C. A., and Pontil, M.
On learning vector-valued functions.
Neural Computation, **17**, 1 (2005), 177–204.

[34] Evgeniou, T., Micchelli, C. A., and Pontil, M.
Learning multiple tasks with kernel methods.
Journal of Machine Learning Research, **6**, (2005), 615–637.

[35] Argyriou, A., Micchelli, C. A., and Pontil, M.
When is there a representer theorem? Vector versus matrix regularizers.
Journal of Machine Learning Research, **10**, (2009), 2507–2529.

[36] Zanca, G., Zorzi, F., and Zanella, A.
RSSI measurements in indoor wireless sensor networks.
[Online].
Available: http://telecom.dei.unipd.it/pages/read/59/.

[37] Stone, M.
Cross-validatory choice and assessment of statistical predictions.
Journal of the Royal Statistical Society, **36**, 2 (1974), 111–147.

[38] Medeisis, A., and Kajackas, A.
On the use of the universal Okumura-Hata propagation prediction model in rural areas.
In *Vehicular Technology Conference Proceedings*, May 2000, 1815–1818.

[39] Mao, G., Fidan, B., and Anderson, B. D. O.
Wireless sensor network localization techniques.
Computer Networks, **51**, 10 (2007), 2529–2553.

Sandy Mahfouz was born in Fidar, Lebanon, on November 27, 1989. She received the diploma degree in computer and communication engineering, majoring in telecommunications, in 2012, from the Holy Spirit University of Kaslik, Lebanon. She is currently working toward the Ph.D. degree in systems optimisation and security from the University of Technology of Troyes, France. Her current research interests include wireless and mobile sensor networks, machine learning, and signal processing.

Farah Mourad-Chehade was born on January 15, 1984. She received the diploma degree in electrical engineering from the Lebanese University, Faculty of Engineering, Tripoli, Lebanon, in 2006. She also received the Master degree, in 2007, and the Ph.D. degree in 2010, in systems optimization and security from the University of Technology of Troyes (UTT), France. Since September 2011, she has been an associate professor at the UTT. She has since supervised two Ph.D. theses. She serves as a reviewer for several journals (*IEEE Transactions on Signal Processing*, *IEEE Transactions on Robotics*, *IEEE Transactions on Vehicular Technology*, and *Elsevier Signal Processing*) and conferences (EUSIPCO, WOSSPA, and ROADEF). Her research interests include wireless and mobile sensor networks, nonlinear signal analysis, machine learning, and biomedical applications.

Paul Honeine (M'07) was born in Beirut, Lebanon, on October 2, 1977. He received the Dipl.Ing. degree in mechanical engineering in 2002 and the M.Sc. degree in industrial control in 2003, both from the Faculty of Engineering, the Lebanese University, Lebanon. In 2007, he received the Ph.D. degree in systems optimisation and security from the University of Technology of Troyes, France, and was a postdoctoral research associate with the Systems Modeling and Dependability Laboratory from 2007 to 2008. Since September 2008, he has been an assistant Professor at the University of Technology of Troyes, France. His research interests include nonstationary signal analysis and classification, nonlinear and statistical signal processing, sparse representations, and machine learning. Of particular interest are applications to (wireless) sensor networks, biomedical signal processing, hyperspectral imagery and nonlinear adaptive system identification. He is the coauthor (with C. Richard) of the 2009 Best Paper Award at the IEEE Workshop on Machine Learning for Signal Processing. Over the past 5 years, he has published more than 100 peer-reviewed papers.

Joumana Farah received the B.E. degree in electrical engineering from the Lebanese University in 1998, the M.E. degree in signal, image, and speech processing in 1999, and the Ph.D. degree in third-generation mobile communication systems, in 2002, from the Polytechnic Institute of Grenoble, France. Since January 2010, she holds an Accreditation to Supervise Research (HDR) from the University Pierre and Marie Curie (Paris VI), France. She is currently a full-time faculty member of the faculty of engineering at the Lebanese University. She has supervised a large number of Master and Ph.D. theses. She has been the recipient of several research grants from the Lebanese National Council for Scientific Research and the Franco-Lebanese CEDRE program. She has four registered patents and software and has coauthored a research book and more than seventy papers in international journals and conferences. She was the General Chair of the 19th International Conference on Telecommunications (ICT 2012), and serves as TPC member and reviewer for several journals (*IEEE Journal on Selected Areas in Communications*, *IEEE Communications Letters*, *Signal Processing: Image Communication*, *Digital Signal Processing*, *Annals of Telecommunications*, etc.) and conferences (IEEE VTC, IEEE Globecom, IEEE ICECS, EUSIPCO, ICT, etc). Her current research interests include channel coding techniques, distributed video coding, multicarrier systems, cooperative and wireless sensor networks, and resource allocation techniques.

Hichem Snoussi was born in Bizerta, Tunisia, in 1976. He received the diploma degree in electrical engineering from the Ecole Superieure d'Electricite (Supelec), Gif-sur-Yvette, France, in 2000. He also received the DEA degree and the Ph.D. degree in signal processing from the University of Paris-Sud, Orsay, France, in 2000 and 2003, respectively. He has obtained the HDR from the University of Technology of Compiègne in 2009. Between 2003 and 2004, he was a postdoctoral researcher at Institut de Recherches en Communications et Cybernétiques de Nantes (IRCCyN). He has spent short periods as visiting scientist at the RIKEN Brain Science Institute, Japan, and the Olin Neuropsychiatry Research Center at the Institute of Living. USA. Between 2005 and 2010, he has been associate professor at the University of Technology of Troyes. Since September 2010, he has been appointed a full professor at the same university. He is in charge of the regional research program S3 (System Security and Safety) of the CPER 2007–2013 and the CapSec plateform (wireless embedded sensors for security). He is the principal investigator of an ANR-Blanc project (mv-EMD), a CRCA project (new partnership and new technologies), and a GDR-ISIS young researcher project. He is partner of many ANR projects, GIS, and strategic UTT programs. He obtained the national doctoral and research supervising award PEDR 2008–2012 and the PES (Scientific Excellence Award) 2013–2017.