A machine learning approach for user localization exploiting connectivity data

Article

Accepted version

P. Cottone, S. Gaglio, G. Lo Re, M. Ortolani

In Journal of Engineering Applications of Artificial Intelligence

Publisher: Elsevier

A Machine Learning Approach for User Localization Exploiting Connectivity Data

Pietro Cottone, Salvatore Gaglio, Giuseppe Lo Re, Marco Ortolani*

DICGIM, University of Palermo, Viale delle Scienze, ed. 6 - 90128 Palermo, Italy

Abstract

The growing popularity of Location-Based Services (LBSs) has boosted research on cheaper and more pervasive localization systems, typically relying on such monitoring equipment as Wireless Sensor Networks (WSNs), which allow to re-use the same instrumentation both for monitoring and for localization without requiring lengthy off-line training.

This work addresses the localization problem, exploiting knowledge acquired in sample environments, and extensible to areas not considered in advance. Localization is turned into a learning problem, solved by a statistical algorithm; additionally, parameter tuning is fully automated thanks to its formulation as an optimization problem based only on connectivity information.

Performance of our approach has been thoroughly assessed based on data collected in simulation as well as in actual deployment.

Keywords: Wireless Sensor Networks, range-free localization, Support Vector Machines.

1. Introduction

Currently available technology for environmental monitoring makes it practically feasible to design and deploy a controlled network of sensors and actuators throughout the surroundings of an intelligent system; besides the obvious intended purpose, the interconnected sensing devices indirectly enable advanced functionalities: for instance, they are suitable for making the system aware of the presence of users so that it can capture their preferences and eventually act on the environment to adapt it to their needs [1]. Such vision falls within the scope of Ambient Intelligence (AmI), whose aim is to support the user in carrying out everyday activities in an unobtrusive way, and in this context Location-Based

*Corresponding author.

Email addresses: pietro.cottone@unipa.it (Pietro Cottone), salvatore.gaglio@unipa.it (Salvatore Gaglio), giuseppe.lore@unipa.it (Giuseppe Lo Re), marco.ortolani@unipa.it (Marco Ortolani)

Services (LBSs) play a central role as location is a very valuable piece of information in order to deliver the correct service to the user. A very common choice for the sensing equipment consists in Wireless Sensor Networks (WSNs), whose characteristics of pervasiveness and inconspicuity meet two of the most specific requirements of the AmI philosophy [2].

In this work we propose an approach to the design of a WSN-based localization module, so as to allow the user to assess the accuracy obtainable when taking into account the practical limitations arising from the actual deployment; the aim is clearly to minimize the amount of nodes scattered through the environment, without excessively losing precision. Our proposal is characterized by a fully automated parameter selection, so that it may be easily adapted to different scenarios without user intervention; the corresponding module is fed with online data elaborated during the localization process. As regards localization, specifically, our proposal was inspired by a state-of-art algorithm, which was deeply modified in order to operate on different kinds of connectivity data, and to take advantage of our automated parameter selection process. Moreover, new kernels were used for the learning algorithm, in order to make it more resilient to a change in the external conditions, and, finally, we introduced a new mechanism to make the system aware of changes in settings and adapt to them.

The underlying idea is that information about connectivity, such as Received Signal Strength Indicator (RSSI) or Link Quality Indicator (LQI), can be used to get an approximated location of a sensor node whose position varies slowly within the borders of the covered area, as widely reported in literature [3, 4, 5]. A common assumption is that the node is carried by a user moving through the monitored area. For the sake of generality, however, we adopt a novel approach with respect to the majority of works in this field, and we avoid making use of any a priori information, such as network topology or the layout of the monitored area; rather, we just require knowledge about the actual position of a limited subset of nodes in the network.

It is well known that link quality estimators are characterized by very poor reliability [6], so the performance of any system based on them must be carefully assessed; a typical approach consists in arranging ad-hoc testbeds, which is often impractical as it depends on expensive, and hard to maintain deployment of nodes in complex scenarios [7]. Moreover, this does not allow to test for application scalability, nor to evaluate the system behavior across different configurations, so an alternative strategy involves simulating the environment, the sensing equipment, and the interaction with the system in order to get an estimate of the overall performance [8].

The concern regards in this case the reliability of the simulation, so we chose to rely for our analysis on models generated by a *hybrid* simulator for WSNs, where virtual nodes coexist with real ones [9]. The use of simulated nodes allows to limit the deployment to just a minimal set of real nodes, which may serve as realistic data model generators to steer the behavior of their virtual counterpart. This is useful to allow the simulation of environmental scenarios wider than those actually at researchers' disposal, as long as "tunable" models may be created for data, as discussed in [10]; models can be instantiated according to real past

sensory readings, and tuned with respect to on-line incoming ones. We tested our approach on simulated as well as on empirical data, demonstrating that localization can be abstracted from the specific information source, so that the parameters of the localization system can be directly inferred from the statistical model describing the monitored area. Thus, the problem of localization can be easily adapted to the available information, independently of a specific signal model, and can take advantage of data from multiple information sources [11].

This aspect sets our approach apart from a great number of localization systems that are designed and tuned for some specific signal process only. Those systems often outperform more general ones, but they are typically overadapted to the specific environment and WSN; even a little change in the monitoring system or the environment can significantly degrade the localization accuracy. On the other hand, accuracy of our system is not heavily affected by the particular setting or scenario, as our experiments demonstrate.

The remainder of the paper is organized as follows. Section 2 provides some background on documented methods for localization, and Section 3 describes our proposal focusing on obtaining a generalizable approach. Section 4 presents our experimental results on simulated and actual data, and finally Section 5 states our conclusions.

2. Related Work

Localization has been a widely studied topic during the last decades.

Range-based schemes can be considered the earliest approaches to localization in WSN, according to the taxonomy developed in [12]. They compute location by estimating the absolute pairwise distance between two nodes, using RSSI, Time Difference of Arrival (TDoA), or Angle of Arrival (AoA). For example, the system proposed in [13], uses RSSI registered at multiple base stations combined with signal propagation models in order to estimate node location. The main drawback of these approaches is the difficulty in obtaining a reliable model of the signal transmission for the ranging process; as a consequence, an accurate estimate requires very expensive hardware. Often, the localization accuracy needed by most WSN applications does not justify the high costs due to range-based methods; thus, research has shifted toward alternatives providing a coarser yet sufficient accuracy, while requiring fewer hardware and software resources.

As a result, *range-free* algorithms have been conceived; they do not assume that a direct distance or angle estimation is available, thus avoiding the creation of a model of the signal propagation. However, they use signal strength information to narrow the area the sensor might be in (the so-called *area-based* approach).

The range-free methods can be parted into two major classes: optimization approaches and machine-learning based ones. Systems belonging to the former class transform localization into an optimization problem, whose constraints arise from physical characteristics (Radio Frequency (RF) transmission system,

topology, etc). On the contrary, machine-learning based approaches correlate positions to the input, without using any a-priori model. The first class of algorithms can provide more accurate results, but is less adaptable to different scenarios, unless user intervention is done to accurately adapt the model to the new environment. Machine-learning localization schemes are more flexible and, if carefully designed, can adapt to new scenarios without any user intervention, implicitly capturing the new main physical features.

One of the first examples of range-free optimization systems is presented in [14], whose authors propose *APIT*, a localization scheme where few nodes (beacons) are aware of their own position (via GPS or other mechanism) and are equipped with high-powered transmitters. The environment is parted into overlapping triangular regions, whose vertices are the beacons. Since the attenuation of the signal strength is monotonically decreasing in a certain direction, in absence of obstacles, then each node can determine whether it resides within a triangle by using the signal strength received from beacon nodes, and comparing it to the one coming from its neighbors; location is estimated as the center of gravity of the most probable areas the sensor belongs to (i.e., intersections of triangles the node is assigned to).

In [15], the authors solve the localization problem modeling the underlying network as a set of geometric constraints on the node position; the global solution, yielded by convex optimization, provides reliable estimate of the unknown positions. Constraints derive from connections, RF transmitter model and receiver model.

A similar approach was used in [16], where a range-free localization problem is solved by finding a feasible solution to a class of nonlinear inequalities defined on a graph representing the WSN. In particular a recurrent neural network is used to address the problem, enabling a distributed real-time computation over the nodes of the network.

The authors of [17] propose a Maximum Likelihood (ML) formulation of the localization problem, solved through a class of convex relaxations, depending on the noise probability density function of the collected measurements. A distributed algorithm is used to solve the optimization problem, enabling each sensor to estimate its position by communicating only with its neighbors.

One of the most significant contribution of the range-free leaning-based algorithms is the idea to address localization as a classification: due to the lack of a very reliable signal propagation model, signal strength can be used as feature to decide whose regions a sensor belongs to, giving up locating the sensor directly from the ranging information and turning to powerful machine learning approaches. In [18], signal strength information from sensor nodes is stored in a kernel matrix and regression methods are employed to estimate the location of unknown nodes. The authors propose to code locations as complex numbers, using a linear weighted sum of kernel entries as estimating function. A different approach based on a regression model is presented in [19], where a very accurate cost function based on regression tools, devoid of local minima, is proposed. A common drawback of regression tools is the intrinsic difficulty related to accurate parameter tuning. The authors of [20] propose a localization approach

based on the Harmony Search (HS) algorithm and a local search procedure, in order to tackle the problem of a low connectivity degree of the network.

A classification-based scheme is implemented in [21], whose authors propose an area-based system exploiting RSSI and Support Vector Machines (SVMs) [22]. Its two main steps are called *coarse-* and *fine-grained* localization: the monitored area is covered by overlapping regions, whose shape (circular, diamond, elliptical, etc) is chosen by the designer; for each region, a binary classification problem is set up: the training set is made up by pairs composed by the RSSI reading vector that each beacon receives from the others (basically indicating whether the node location falls inside the region or not). A SVM is trained for each region and used in the coarse-grained localization phase, in order to find out the regions a sensor falls into. In the fine-grained phase, the location of the node is computed as the barycenter of the set of regions the sensor was classified in. An improved version of this work is proposed in [23], addressing the localization problem in large networks, and relying only on connectivity information (i.e., the hop-count length of the shortest path between two nodes), rather than RSSI. Similarly to the previous work, a bound on localization error is provided and a Modified Mass-Spring Optimization is suggested in order to improve localization accuracy. In [24, 25] localization schemes very similar to the one presented in [21] are proposed, analyzing the influence of SVM parameters and other factors on localization accuracy. However, kernel-based approaches are applicable only in presence of very dense networks, in order to obtain acceptable accuracy.

Nowadays, a common technique for localization in wireless network is based on the so-called *fingerprints*: in the online phase, a map is constructed, associating to each location the RSSI vector perceived from deployed sensors (the so-called fingerprint); in the off-line phase, the location of a sensor can be inferred comparing the current registered RSSI vector with those stored into the map [26]. In [27], Fuzzy C-Means clustering is used to improve quality of the fingerprints and deal with uncertainty embedded into the data; for the on-line phase a Nearest Neighbor algorithm is used. The use of fingerprints, however, requires dense sampling and may not be reliable in complex environments where few sensors are deployed.

Table 1 summarizes the main differences between the approaches presented in this section.

3. A Generalizable Approach to Localization

A reliable estimate of the user location is of utmost importance for an intelligent system whose goal is the control of the surrounding environment. In a practical scenario, a network of pervasively distributed sensors collects measurements about typical physical quantities (such as temperature, humidity, light exposure), coupled with the corresponding actuators that allow to modify the environment. This is quite a common setting in indoor applications for home or office automation, where even large areas may be subject to monitoring, and

Table 1: Comparison between different localization approaches.

	Centralized vs distributed	**Input data**	**Technique**
Bahl *et al.* [13]	Centralized	Signal strength, directions	Triangulation
He *et al.* [14]	Distributed	Signal strength	Point-in-Triangulation Test
Doherty *et al.* [15]	Centralized	Signal strength	Convex optimization
Li *et al.* [16]	Distributed	Signal strength	Recurrent neural networks
Simonetto *et al.* [17]	Distributed	Distance estimate, noise probability density function	ML estimation
Kuh *et al.* [18]	Centralized	RSSI	Least Squares Kernel Regression
Vanheel *et al.* [19]	Centralized	RSSI	Linear Regression
Manjarres *et al.* [20]	Centralized	Distance estimated through RSSI	Heuristic Optimization
Nguyen *et al.* [21] Afzal *et al.* [24] Wu *et al.* [25]	Centralized (model), with local position computation	RSSI	SVM
Haque *et al.* [26]	Centralized	RSSI	Fingerprint k-Nearest Neighbor (k-NN)
Suroso *et al.* [27]	Hybrid	RSSI	Fingerprint Fuzzy C-Means
Our proposal (SVMloc^{gen})	Centralized (model), with local position computation	RSSI, LQI, Received Power (Pr)	SVM

may be characterized by the presence of clearly identifiable sub-regions, distinguishable by their intended use and purpose [28]. For instance, control of the sources of artificial lighting is a typical goal of an AmI system, in order to provide a better user experience, by trying to infer the occupant's preferences (e.g. by building user profiles) without an excessive impact on energy consumption. The key functionality is the ability to detect user presence in selected areas, triggering the actuators only when needed. As already mentioned, our underlying assumption regards the availability of an adequate amount of sensor nodes deployed for monitoring purposes, which we want to employ also to implement LBSs based on user presence detection. A node moving within the sensor field may well represent the user crossing the monitored area at moderate speed, and our aim is to become aware of the user's presence, without necessarily tracking him or her completely. Our assumption thus implies that the user position does not change significantly between two consecutive scans, so the sampling frequency needed to follow them on their path does not need to be high. Moreover, we are not interested in the user's route through the monitored field, but only in detecting their presence in specific areas. In our view, a path is just a sequence of uncorrelated locations; although this choice may appear simplistic, it does allow to use a clear model to estimate the user's position; additionally, the computation does not require storing previous positions, which is advantageous given the constrained memory resources of the node.

In this scenario, we define the problem of localization as follows:

Statement 1. *Given a wireless sensor network of N nodes $S = \{s_1, \dots, s_N\}$ and a coordinate system for their locations, the set S can be parted into two classes:*

- *a subset $B = (b_1, \dots, b_n)$ including n beacon nodes, whose coordinates $X^{(b)} = \left\{(x_1^{(b)}, y_1^{(b)}), \dots, (x_n^{(b)}, y_n^{(b)})\right\}$ are known;*
- *a subset $T = (t_1, \dots, t_m)$ composed of m target nodes, whose coordinates $X^{(t)} = \left\{(x_1^{(t)}, y_1^{(t)}), \dots, (x_m^{(t)}, y_m^{(t)})\right\}$ are unknown, and possibly vary over time;*

with $m \gg n$ and $N = n + m$. Moreover, there exists a distinct node, known as base station, whose computing power is assumed to be greater with respect to the other nodes. Finally, let $\mathbf{r}_i \in \mathbb{R}^n$ indicate the connectivity information registered by node i from the beacons.

The localization problem consists in estimating the set $X^{(t)}$, given only the beacon connectivity information matrix $\mathbf{R} \triangleq (\mathbf{r}_i)_{i=1}^n \in \mathbb{R}^{n \times n}$ (i.e., whose rows are the $\mathbf{r}_i$ vectors) and the target connectivity information matrix $\mathbf{R}_T \triangleq (\mathbf{r}_j)_{j=1}^m \in \mathbb{R}^{m \times n}$ (i.e., whose rows are the $\mathbf{r}_j$ vectors).

Statement 1 is a common statement of the localization problem, allowing a formulation in the context of a machine learning framework; in particular, we follow the proposal by [21], and regard the monitored area as partitioned into H partially overlapping regions. We assume that the monitored environment can be adequately covered by a medium-size network, so the large majority of the beacon pairs will be formed by nodes within the communication range of each other. The base station, on the other hand, is a central node delegated to collecting and preprocessing sensor data, and carries out the calculation needed to compute our model.

Furthermore, we assume that the measurements for our testbed are provided by a *hybrid* WSN, where a small set of nodes actually deployed in the field coexist with a larger amount of virtual ones, thanks to a specific *simulator* that allows to regard them as part of the same network [9]. Such tool provides a simple way to test the application behavior in qualitatively different environments, with the freedom to choose the shape and size for the setting; the adherence to reality is granted by the inclusion of the minimal set of actual nodes, whose sensed data are used as *probes* to generate predictive models for the actual physical quantities; flexibility is also taken into account by adapting the obtained models to simulate the behavior of *virtual* nodes. The basic idea is that when a new area needs to be monitored, a few probe sensors will be deployed so that sample data may be collected in order to infer the connectivity information characterizing the target environment, as depicted in Figure 1. Models in this case will consist of a connectivity matrix, $\mathbf{R}$, that can be interpreted as a descriptor of the network and the environment; it is an adjacency matrix, whose elements are the weights associated to the edges of an undirected and connected graph, and basically conveys information both about the topology of the monitoring network and about the specific structure of the environment. Training such model requires

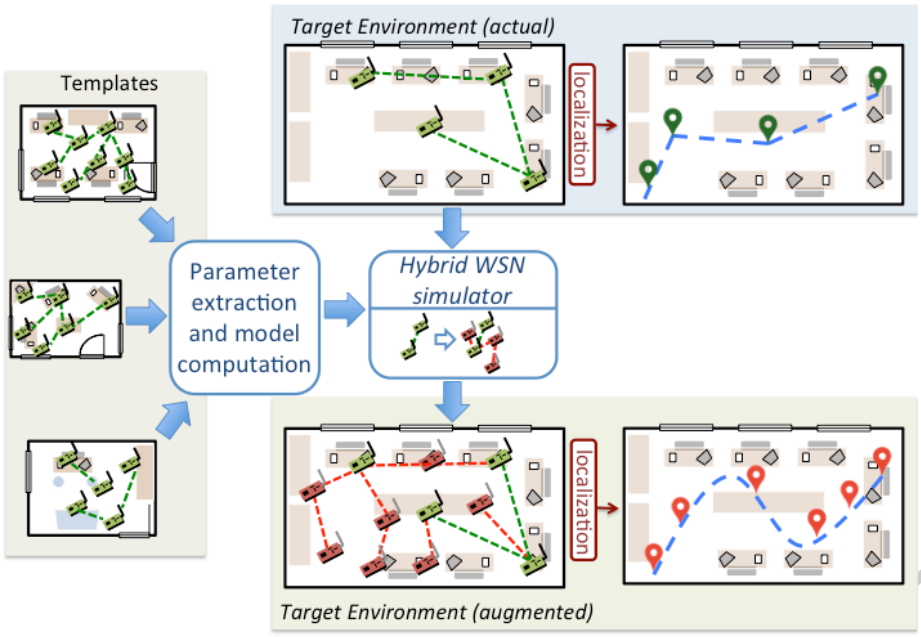

Figure 1: Using the hybrid WSN simulator to get a better estimate on the localization error. Connectivity models computed over template environments are used to simulate a dense network over a target environment, where the actual WSN is augmented with virtual nodes. Data collected from probe nodes (green) are used to simulate the behavior of virtual ones (red).

collecting connectivity data from the sensor field; this involves only beacon nodes, with no overhead for the test node, and the corresponding protocol is composed of 4 main steps, as outlined in the Localization Algorithm on page 9. We assume here that the position of the beacon nodes is known a priori to the base station. Once the model has been computed, it is broadcast over the network, so that target nodes can receive and store it. After acquiring the model, target nodes are able to calculate their position independently, until a new model will be distributed. The impact on the number of sent and received messages, and on the amount of computation required of the target node is minimized; moreover, if the target node does not need to known its own position, it can just temporarily turn off the localization module.

In literature, several approaches have been proposed in order to turn a centralized algorithm into a distributed one [29, 30, 31], but they do not apply to our context, due to the limited computational resources of beacon nodes.

It is worth noting that the proposed algorithm is intrinsically adaptive to novel environmental conditions: when beacon nodes receive the data in the localization step, they compute their position and compare it to their actual position, and notify the base station about the resulting localization. As soon as the mean error, computed among all beacons, exceeds a prefixed threshold the base station triggers a new training phase, by re-initializing the protocol. The beacons nodes act like validator for the accuracy of the model, allowing an on-line control on the performance of the localization system.

Localization Algorithm

Precondition:

- 1 Base Station (**BS**), n Beacon Nodes (**BN**s), H regions.
- The Base Station (BS) knows the exact locations, $\mathbf{p}_i$ $\forall i = 1, \dots, n$, of Beacon Nodes (BNs).
- θ_{err} is a threshold on the Mean Localization Error (MLE) over the BN set
- Each BN knows the region centers $\mathbf{c}^{(h)}$, $\forall h = 1, \dots, H$

procedure
  loop
  //The BS computes the MLE for the BN set
    $MLE \Leftarrow (1/n) \sum_{i=1}^{n} d(\mathbf{p} - \hat{\mathbf{p}})$
    if $E > \theta_{err}$ **then**
    //The BS sends a message to BNs, in order to put them into listening mode
      BS : broadcast()
    //The BS starts a timer T
      BS : T.start()
    //Each BN sends an ack to the BS
      $\mathbf{BN}_i$: ack(BS), $\forall i = 1, \dots, n$
    //Each BN switches to the listening mode
      $\mathbf{BN}_i$: listen(), $\forall i = 1, \dots, n$
    //After reception of all ACKs, or timer expiration, all BNs are triggered
      BS : wait_until(all_ACK_received() **OR** T.has_expired())
      BS : broadcast()

    1^{st} **phase: Flooding**
    //A pre-set time slice is assigned to every node
      for $i = 1 : n$ **do**
      //Each BN sends a broadcast message.
        $\mathbf{BN}_i$: broadcast()
        //All the other beacons store the connectivity information.
        $\mathbf{BN}_j$: store($\mathbf{r}_j$), $\forall j = 1, \dots, n$, $j \neq i$
      end for

    2^{nd} **phase: Collection**
    //Each node sends the connectivity data.
      $\mathbf{BN}_i$: $send(\mathbf{BS}, \mathbf{r})$, $\forall i = 1, \dots, n$
    //The BS trains a SVM for each region
      for $h = 1 : H$ **do**
        $(\boldsymbol{\alpha}^{(h)}, b^{(h)}, \mathbf{R}_v^{(h)}) \Leftarrow \mathbf{BS}$: trainSVM($\mathbf{h}, \mathbf{R}$)
      end for

    3^{rd} **phase: Model Distribution**
    //Models are broadcast to all nodes
      BS : broadcast($\boldsymbol{\alpha}^{(h)}, b^{(h)}, \mathbf{R}_v^{(h)}$)
    end if

  4^{th} **phase: Localization**
  //BNs compute their estimated position according to the models
    $\hat{p} \Leftarrow \mathbf{BN}_i : est\_pos(\boldsymbol{\alpha}^{(h)}, b^{(h)}, \mathbf{R}_v^{(h)}), \forall i = 1, \dots, n$
  //BNs send their estimated position to the BS
    $\mathbf{BN}_i$: $send(\mathbf{BS}, \hat{p})$, $\forall i = i, \dots, n$
  end loop
end procedure

3.1. Localization through Classification

Our approach is based on the work presented in [21], where a two-phase localization algorithm is proposed. We modified some parts of this algorithm in order to overcome some limitations and known issues, and to add a fully automated procedure for parameter tuning, allowing it to adapt to several different environments. In the following, we refer to the original approach as SVMloc, and to our generalization as SVMloc^{gen}.

We retain the original structure of SVMloc and the same approach to localization as a classification problem, solved by a set of SVM classifiers.

The first step of the localization scheme consists in what is called *coarse-grained localization*, and is formulated as a binary classification problem: given a node s, its RSSI reading vector $\mathbf{r}$ and a region A_h, classification will consist in determining whether s falls within region A_h or not. This problem is solved by means of SVMs; in particular, a decision function is associated to each region:

$$f^{(h)}(\mathbf{r}) = \text{sign}\left(\sum_{i=1}^{n} y_i \alpha_i^{(h)} \kappa\left(\mathbf{r}, \mathbf{r}_i\right) + b^{(h)}\right), \tag{1}$$

where $\kappa(\cdot,\cdot)$ is the chosen kernel function defining the SVM. Eq. (1) represents the hyperplane separating the two classes in the feature space, i.e. a higher dimensional space induced by the kernel function κ on input data where points are linearly separable.

Several different shapes for H were proposed in [21], and considered during our experiments. Our conclusion, in accordance to what is discussed in [24], is that circular regions are the most suitable for our setting; thus, in the following, we considered the H regions as circles.

The training set for each SVM is composed by pairs $(\mathbf{r}_i, y_i)_{i=1}^{n}$, where y_i is set to 1 if the beacon s is inside the region, and to -1 otherwise. Once trained, each SVM is completely defined by the 3-tuple

$$\left(\boldsymbol{\alpha}^{(h)}, b^{(h)}, \mathbf{R}_v^{(h)}\right),$$

where $\boldsymbol{\alpha}^{(h)} = \{\alpha_i^{(h)} : \alpha_i^{(h)} \neq 0\}$ and $\mathbf{R}_v^{(h)} = \{\mathbf{r}_i : \alpha_i^{(h)} \neq 0\}$ is the set of the so-called *support vectors* for the h^{th} SVM (i.e. region). In other words, each target node can establish the region(s) it belongs to by Eq. (1), provided the set $\left\{(\boldsymbol{\alpha}^{(1)}, b^{(1)}, \mathbf{R}_v^{(1)}), \cdots, (\boldsymbol{\alpha}^{(H)}, b^{(H)}, \mathbf{R}_v^{(H)})\right\}$.

The second phase of the algorithm is the *fine-grained localization*, whose aim is to estimate the position $(x_j^{(t)}, y_j^{(t)})$ of the j^{th} target node as the center of mass of the regions the node is sensed to be in. The method originally proposed by SVMloc for this step presents some drawbacks that make it not entirely suitable for use in an actual context; we thus slightly modified the selection of the centers for overlapping-regions by choosing appropriate kernels for the SVMs. The authors of [23, 24], for instance, analyze the so-called "border effect", consisting in a bias of the estimated positions of the nodes close to the border of the monitored field towards the center of the area. This may be an inconvenience,

as in our context the border region may represent a considerable part of the monitored area, when a limited set of nodes is used. In SVMloc^{gen}, we deal with this phenomenon by following the suggestion in [25]; some extra regions whose centers are placed outside the monitored area are artificially added, resulting into increased accuracy along the borders, at the expenses of an additional computational cost.

In SVMloc, other issues may arise when the number of overlapping regions is not adequate: for example, if two nodes are assigned to the same regions, they will eventually get the same estimated position, even if they actually reside in very different parts of the intersection. This phenomenon is not particularly significant when the intersection area is sufficiently small, which could be obtained by increasing the number of overlapping regions, with an obvious increase also in the computational cost; in SVMloc^{gen}, we chose instead to compute the node's position by a weighted sum of the region centers, as opposed to a simple arithmetic mean, where each weight expresses a degree of membership of the nodes to a specific region, i.e. an estimate of the *reliability* of the classification.

A possibility is to express the *belief* about the correctness of the output by computing a probabilistic output from the SVM [32]:

$$P(t_j \in A_h | \mathbf{r}_j) = P(y_j^{(h)} = 1 | \mathbf{r}_j) = p^{(h)}(\mathbf{r}_j). \tag{2}$$

and to perform the computation of fine-grained localization $\mathbf{x}_j$ for the j^{th} node as follows:

$$\mathbf{x}_j = \frac{\sum_{h \in I_j} w_j^{(h)} \mathbf{c}^{(h)}}{\sum_{h \in I_j} w_j^{(h)}}, \tag{3}$$

where $w_j^{(h)}$ is set to the probability expressed by (2), $\mathbf{c}^{(h)}$ indicates the coordinates of the center of the h^{th} region, and I_j is the set of the indices of the regions the node belongs to.

In an actual context, however, computational resources may be at a premium in order to extend each node's lifetime. The distance $d_j^{(h)}$ of a point $\mathbf{r}_j$ from the hyperplane characterizing the j^{th} SVM does not entail any relevant additional cost, yet it provides a rough approximation of a membership function and is an acceptable candidate for a reliable membership function, which may be used as $w_j^{(i)}$ in Eq. (3):

$$d_j^{(h)} = \frac{\left| \sum_{i=1}^{n} \alpha_i^{(h)} \kappa(\mathbf{r}_j, \mathbf{r}_i) + b^{(h)} \right|}{\left(\sum_{i=1}^{n} \alpha_i^{(h)} \right)^{1/2}}. \tag{4}$$

It is worth noting that each classification occurs in *the same* feature space, as the same kernel function is used with identical parameters for the whole set of SVMs. Thus, all SVMs are embedded into the same Hilbert space and the obtained positions are identical across all classifications; only their labels y_i are subject to change from a classification to another, according to the relative position of the point with respect to the region. Hence, distances from different

hyperplanes are comparable.

Computation of the weights with either method is however quite sensitive to the presence of noise or outliers in the training set, which may have a non negligible impact in a practical scenario.

In the present work, we also analyzed the effect of the choice of kernels on the overall robustness, with particular regard to the ones proposed by [21]. The *first-tier* kernel proposed therein is a manipulation of the connectivity information matrix $\mathbf{R}$, in order to guarantee the Mercer condition [33], i.e. the requirement for κ (and hence $\mathbf{R}$) to be semidefinite positive. The *second-tier* kernel is a linear one, simply represented by the inner product between feature vectors. Finally, the *third-tier* kernel is a classical Gaussian kernel. In SVMloc^{gen}, we propose a regularized version of kernels [22], which operate on feature vectors normalized according to the 2-norm and with their mass center in the origin of the feature space. Given a "standard" kernel function $\kappa(\mathbf{x}, \mathbf{z})$, its regularized version $\kappa'(\mathbf{x}, \mathbf{z})$ between two feature vectors $\mathbf{x}$ and $\mathbf{z}$ is defined as:

$$\kappa'(\mathbf{x}, \mathbf{z}) = \frac{\widehat{\kappa}(\mathbf{x}, \mathbf{z})}{\sqrt{\widehat{\kappa}(\mathbf{x}, \mathbf{x})\widehat{\kappa}(\mathbf{z}, \mathbf{z})}}, \tag{5}$$

with:

$$\begin{aligned}\widehat{\kappa}(\mathbf{x}, \mathbf{z}) = \kappa(\mathbf{x}, \mathbf{z}) - \frac{1}{n}\sum_{i=1}^{n}\kappa(\mathbf{x}, \mathbf{r}_i) - \\ \frac{1}{n}\sum_{i=1}^{n}\kappa(\mathbf{z}, \mathbf{r}_i) - \frac{1}{n^2}\sum_{i=1}^{n}\sum_{l=l\,i}^{n}\kappa(\mathbf{r}_i, \mathbf{r}_l)\end{aligned} \tag{6}$$

where $\mathbf{r}_i$ are the rows of the kernel matrix (in this context, the rows of $\mathbf{R}$).

Eq. (5) defines a normalization, while (6) is the inner product of feature vectors, referred to the center of mass of the training set [33]. After these manipulations, the feature vectors lay inside a hypersphere centered in the origin of the feature space; if the training set is in fact representative of the data, these operations make parameter selection simpler and less dependent from the particular instance of the problem.

3.2. An Automatic Method for Parameter Selection

Besides choosing the appropriate kernel, it is important to carefully adapt it to the specific scenario, by setting its parameters properly; this is a main concern in SVMs, as their classification performance is highly dependent on the choice for parameters; moreover, an empirical choice is not often easy. In our scenario, in particular, parameters are strictly tied to the environmental features, since they are function of the connectivity information (e.g. RSSI, or LQI), which is clearly highly dependent on the specific characteristics of the monitored area. Several factors influence signal transmission, which are quite hard to be precisely modeled [34]; however, they can be coded into the tuning parameters of the SVMs, hence reliability can be enhanced by inferring

an optimal set of parameters, even from just a limited amount of sensor data, such as those obtainable by our hybrid simulator's probes.

Two main parameters are involved in optimization of the SVM: C, a normalization parameter, and σ, which characterizes the Gaussian kernel function. A common approach to parameter selection is to perform a grid search, by testing the performance of the system with a set of (C, σ) pairs. In our context, this approach does not appear feasible, as it would require performing a grid search for all the areas we intend to monitor, which clashes with our assumption that just few sensor nodes are available in probe environments; we therefore opted to enrich SVMloc^{gen} with a fully automated tuning procedure. The choice of the C parameter influences the generalization error of the SVM [33], that is crucial in our approach. Following [35], we compute C as function of the square of the radius R of the hypersphere enclosing the training data. This radius is estimated by using Support Vector Data Descriptions (SVDDs), and its value is computed as $C = R^{-2}$.

The selection of σ influences the nonlinear transformation from input space to feature space, when a Gaussian kernel is used. In particular, a Gaussian kernel is defined as:

$$\kappa(\mathbf{r}, \mathbf{r}_i) = \exp\left(-\frac{\|\mathbf{r} - \mathbf{r}_i\|^2}{2\sigma^2}\right). \tag{7}$$

Overfitting is a well-known drawback of this kernel, which is unavoidably mirrored in the selection of the σ parameter, that has been shown to intervene in the generalization ability of the SVM.

In this work, we propose an algorithm for tuning σ that takes advantage of a heuristic adaptive to the specific collected data, which is based on the properties of the kernel matrix. We chose to adopt the solution proposed in [36], that does not imply additional cost for the training phase, unlike computational and time-intensive techniques like cross validation; this method was originally applied to One-class SVM, and we adapted it to the general case. The main idea is to select the value of σ that maximizes the spread of values in the $[0; 1]$ range (the co-domain of the Gaussian kernel function). Overestimating σ flattens the kernel function, hindering its classification ability, while the opposite case leads to overfitting. In our context, the value of σ is obtained by solving the following optimization problem:

$$\max_{\sigma} \frac{\varsigma^2}{\overline{\mu} + \epsilon}, \tag{8}$$

where ς^2 and $\overline{\mu}$ are the variance and the mean of the triangular upper part of the kernel matrix $\mathbf{K} = (k_{il})_{i,l=1}^{n} \in \mathbb{R}^{n \times n}$, where $k_{il} = \kappa(r_i, r_l)$, and ϵ is a real positive small value, acting as stabilization parameter to strengthen numerical stability.

Problem (8) is solved by hill climbing optimization, which corresponds to

minimizing the term:

$$\sum_{i,l=1}^{n} y_i y_l \alpha_i \alpha_l \kappa(\mathbf{r}_i, \mathbf{r}_l) \tag{9}$$

which may be rewritten as $\boldsymbol{\alpha}^T \mathbf{Q} \boldsymbol{\alpha}$, with $\mathbf{Q} = \mathbf{YKY}$, where $\mathbf{Y} \in \mathbb{R}^{n \times n}$. So, a very sparse $\boldsymbol{\alpha}$ is desirable, because a reduced number of support vector is an indicator of a good generalization ability. If the matrix $\mathbf{K}$ is very sparse, and so is $\mathbf{Q}$, minimizing (9) will not provide meaningful support vectors, because $\boldsymbol{\alpha}$ will be composed by a large number of null elements. Therefore, maximizing the variance of the kernel matrix guarantees a better solution to SVM optimization problem, adapted to the data and less influenced by the minimization algorithm.

4. Experimental Assessment

In order to assess the applicability of our method, we specifically consider its performance both with respect to the achievable accuracy, and its practical relevance in a real-life context.

In the first set of experiments, we evaluate the performance of the proposed approach with respect to the original one. In particular, we test our system with the same setting as in [21].

In the second group of experiments, a set of real data is used to assess our approach to parameter tuning in the context of a simulator for WSNs. The simulator is featured with models ported from template environments, that are used to predict the localization accuracy in a new one, guiding the designer in choosing the best topology and density for the network, given a desired accuracy.

4.1. Simulated environment

Initially we aim at assessing the impact of our modifications to [21]; we used the same settings indicated therein and conducted a set of tests in a simulated environment.

Namely, the simulation area is a 10m $\times$ 10m square, where beacon nodes are positioned along a grid, with an added Gaussian noise in every direction, in order to simulate random deployment, while still guaranteeing adequate coverage of the area. The total number of beacon nodes depends on the granularity of the grid; in our tests we consider grids whose densities range from 3×3 to 10×10. Target nodes are placed at the vertices of a 20×20 grid, superimposed to the monitored area.

We used two different signal models to provide connectivity information. The first model is represented by a Gaussian signal, with varying standard deviation σ_s, corrupted by a white Gaussian noise with standard deviation $\tau = 0.2$, so as to simulate qualitatively different signals. This is the same signal model used to test SVMloc by its authors, so it enables us for a side-by-side performance comparison with respect to SVMloc^{gen}. Moreover, in order to assess the behavior of our approach in a more realistic setting, we also explored the influence of a more refined simulated signal; to this purpose, we used the model

Table 2: Parameters of the signal propagation model for the four selected scenarios.

Scenario	$\mathbf{n_p}$	$\boldsymbol{\sigma^2}$	$\mathbf{P_0(d_0)}$	$\mathbf{d_0}$
Engineering 1	1.9	5.7	[-50.5; -39]	1
Apartment hallway	2	8	[-38.2; -35]	1
One-sided corridor	1.9	8	[-44.2; -33.5]	1
One-sided patio	3.2	3.7	[-39; -34.2]	1

proposed by [37] and showed in Eq. (10), specifically designed for low power wireless links. In particular, authors derive an expression for the packet reception rate as a function of distance (d), that incorporates important channel and radio parameters such as the reference distance (d_0), path loss exponent (n_p), the shadowing variance of the channel (ς), and the modulation and encoding of the radio:

$$P_R(d)_{[dBm]} = P_0(d_0)_{[dBm]} - 10n_p \log_{10} \frac{d}{d_0} + X_\varsigma. \tag{10}$$

We parameterized the model according to the suggestions reported in [38], that experimentally computed the parameters from real data, collected in several actual scenarios. We selected the four scenarios resembling our setting the most, namely: *Engineering 1*, *Apartment hallway*, *One-sided corridor* and *One-sided patio*, as summarized in Table 2.

Accuracy of localization was measured by computing the mean localization error for the set of target nodes, varying the number of beacons nodes, the radius and the number of regions used in the coarse-grained phase and each test was averaged across 10 runs.

In Figure 2, a sample of localization accuracy results is shown for all the mentioned scenarios, demonstrating that accuracy is heavily influenced by density of the beacon network; however, over a certain value, the increased number of beacons does not significantly improve the accuracy.

Tables 3, 4, 5 report the results of the comparisons between our scheme (SVMloc^{gen}) and the original one (SVMloc) for the different kernels, using 64 beacons; 225 regions were used, each with a 5m radius. SVMloc^{gen} consistently outperforms the original approach across the different environments, exhibiting considerably lower mean error, thus proving the effectiveness of our modified fine-grained localization, and of automatic parameter tuning, since classical (non regularized) kernels were used in both cases.

The adaptiveness of our approach also depends of the choice for weights $w_j^{(i)}$ in Eq. (3); Table 6 shows the obtained mean and variance of the localization error when those are computed according to a probabilistic belief, or simply depending on the distance from the SVM hyperplane. The outcomes are in this case comparable, so the extra cost deriving from computing the probability output does not appear to be justified.

Table 3: First Tier kernel: mean (variance) of the localization error in meters; comparing our proposal (SVMloc^{gen}) against the original one.

	FIRST TIER	
	SVMloc^{gen}	**SVMloc**
$\sigma_s = 1$, $\tau = 0.2$	0.2914 (0.001)	0.4146 (0.004)
$\sigma_s = 2.5$, $\tau = 0.2$	0.1989 (0.001)	0.3744 (0.001)
$\sigma_s = 5$, $\tau = 0.2$	0.1733 (0.001)	0.3451 (0.001)
Engineering 1	1.0140 (0.060)	2.0493 (0.251)
Apartment Hallway	1.1340 (0.009)	1.8433 (0.004)
One-sided Corridor	1.2042 (0.037)	1.8335 (0.006)
One-sided Patio	0.6102 (0.001)	0.6573 (0.001)

Table 4: Second Tier kernel: mean (variance) of the localization error in meters; comparing our proposal (SVMloc^{gen}) against the original one.

	SECOND TIER	
	SVMloc^{gen}	**SVMloc**
$\sigma_s = 1$, $\tau = 0.2$	0.3225 (0.003)	0.4195 (0.001)
$\sigma_s = 2.5$, $\tau = 0.2$	0.2366 (0.001)	0.3629 (0.001)
$\sigma_s = 5$, $\tau = 0.2$	0.2184 (0.005)	0.3023 (0.001)
Engineering 1	1.1505 (0.002)	1.2736 (0.003)
Apartment Hallway	1.5358 (0.011)	1.6519 (0.005)
One-sided Corridor	1.5984 (0.007)	1.5234 (0.004)
One-sided Patio	0.6122 (0.006)	0.5846 (0.001)

Table 5: Third Tier kernel: mean (variance) of the localization error in meters; comparing our proposal (SVMloc^{gen}) against the original one.

	THIRD TIER	
	SVMloc^{gen}	**SVMloc**
$\sigma_s = 1$, $\tau = 0.2$	0.3545 (0.004)	0.4758 (0.002)
$\sigma_s = 2.5$, $\tau = 0.2$	0.2816 (0.001)	0.4402 (0.001)
$\sigma_s = 5$, $\tau = 0.2$	0.2428 (0.007)	0.4234 (0.001)
Engineering 1	1.0200 (0.001)	1.1021 (0.003)
Apartment Hallway	1.2170 (0.004)	1.1224 (0.003)
One-sided Corridor	1.2514 (0.007)	1.4456 (0.003)
One-sided Patio	0.4798 (0.001)	0.5354 (0.002)

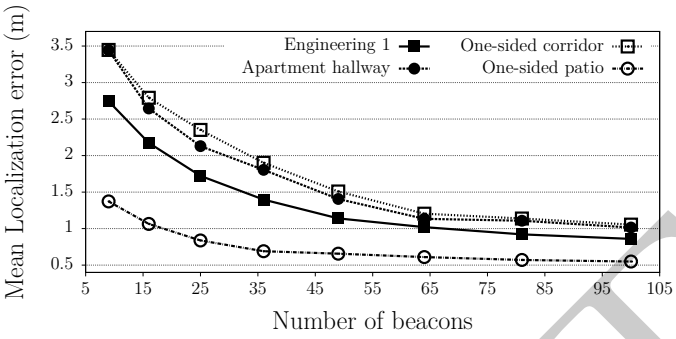

Figure 2: Accuracy of localization, varying the number of beacons for all the four scenarios.

Table 6: Comparing mean (variance) of the localization error in meters for the fine-grained phase.

	Probability	**Distance**
Engineering 1	1.0561 (0.002)	1.1553 (0.003)
Apartment Hallway	1.3418 (0.007)	1.2170 (0.004)
One-sided Corridor	1.4312 (0.008)	1.4080 (0.005)
One-sided Patio	0.5498 (0.089)	0.4809 (0.001)

4.2. Testing the Performances with Real Data

A second set of experiments was also devised to assess the portability of the parameters computed for our localization module; for this purpose, we fed data collected from an actual deployment of nodes, into a hybrid simulator for WSNs. The use of such simulator gives the designer the freedom to port models obtained from a densely deployed network into a new environment, where only a minimal set of probe nodes are available; in other words, we want to avoid calibrating and tuning the localization algorithm over and over for every new environment. As far as link quality estimators are concerned, we assume that different environments can be clustered according to their properties; for instance, some areas will reasonably share similar signal propagation models (e.g. most hallways in a building), whereas other ones will be clearly distinguishable (e.g. office rooms versus lecture halls), and this will be reflected in the signal propagation parameters, as outlined in [38].

For our tests, we used empirical data collected in a real environment, provided from the SIGNET group at the University of Padova [39]. Two datasets were collected using different sensor boards and in different environments: a laboratory (*Signet*), very similar to our simulated area, and a lecture hall (*Lecture Hall*).

The Signet location is a room of 10m×10m, monitored by 48 EyesIFX nodes, positioned at the vertices of a 7×7 grid, suspended at approximately 75 centimeters from the ceiling. The radio channel was disturbed by the presence

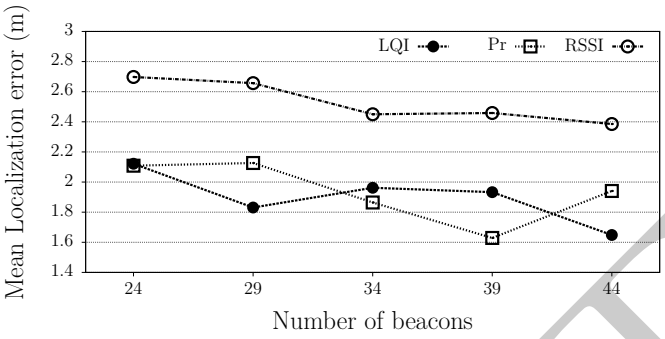

Figure 3: Localization error with 3 connectivity indicators.

of furniture and people in the room, that worsen the multipath interference. IFX-Eyes sensor nodes were used to collect data; each node is equipped with the Infineon TDA5250 radio module, a single chip FSK/ASK transceiver able of half duplex low datarate communication (19,2 kbps) in the 868-870MHz band.

Measurements of the *Lecture Hall* scenario were collected in the Lecture Hall "Antonio Lepschy" of the University of Padova, using Tmote SKY sensor platforms. The Tmote SKY module is equipped with a Chipcon CC2420 radio module for wireless communication, compliant with IEEE 802.15.4 and providing 250 kbps. Moreover, it is featured with a digital RSSI and a LQI value is provided for each received packet. A set of 25 of these modules were deployed on the chairs of the Lecture Hall. Every node tried to send packets to all the other nodes over all the 16 available RF channels.

Two different pieces of information are available for both datasets: the Pr and the LQI; just for the *Signet* dataset, the RSSI is also provided.

We begin by assessing each of the mentioned connectivity estimators for the localization problem with respect to their reliability. To this aim, we apply our localization method to the two datasets; as an example, the corresponding results for the *Signet* scenario are shown in Figure 3, but they are representative also for the second dataset. Notably, the best results are conveyed by LQI, which will therefore be used as an estimator in the following.

In order to assess the portability of the parameters computed for our localization module in a given environment to a new one, we manipulated the available datasets so as to separate a monitored area into two zones: a larger one, representing a densely monitored area, and a smaller one, representing the region of interest where probe sensors are located. We obtained target regions for both datasets including about 40% of the whole node set, and used the regularized version of the third-tier kernel with 64 regions with radius set to 3.

The effectiveness of porting the model built on the larger region over the target one is measured by means of the localization error, computed via a leave-one-out approach on the nodes belonging to the region of interest.

Results of the described tests are shown in Table 7. The first column reports the mean localization error and variance for nodes in the target area, computed

Table 7: Mean (variance) of the localization error (in meters), for default/ported parameters.

	Default	**Ported**
Signet	2.8484 (1.132)	2.1095 (0.882)
Lecture Hall	3.8946 (1.435)	3.1362 (1.182)

using default parameters for the localization module (identical to the ones reported in [21]: $C = 10$, and $\sigma = 0.5$), whereas the second column shows the performance obtained using the parameters computed on the larger area to port the model to the target one. It can be pointed out that performance does improve by porting the parameters from a known environment to a similar target one, showing the feasibility of the proposed approach.

5. Conclusion

We presented an algorithm for localizing users in a field monitored by a wireless sensor network. In particular, the proposed method allows for tuning the parameters of an SVM in a fully automated manner, so as to obtain models easily adaptable to different scenarios, without requiring in-depth knowledge about low-level issues. A key point of the proposal is its flexibility, as connectivity information may be expressed by different metrics (such as RSSI, LQI); the proposal is also generalizable for use on previously unseen environments, by re-using models built over template areas. The basic idea is that when a new field needs to be monitored, a few probe sensors will be deployed so that sample data may be collected in order to infer the connectivity information characterizing the target environment. The provided experiments on simulated data show the soundness of our proposal; moreover, the results of the tests run on empirical data confirm our intuition about the portability of such models (i.e. localization parameters) across different environments.

Possible future developments in this line of research include investigating some improvements to our system. In particular, it would be possible to substitute current beacon node with more powerful ones, in order to distribute the computation of our model across all the sensor network, as suggested in [40]. Moreover, it would be interesting to explicitly introduce the concept of *path*, using past positions in order to predict the next one. Finally, in order to increase the scalability of our approach, we are studying a mechanism to part the monitored area into zones, delegating each of them to a different base station, with a method inspired to [23].

Acknowledgment

Work partially supported by the Italian Ministry of Education, University and Research on the "StartUp" call funding the "BIGGER DATA" project, ref. PAC02L1_0086 – CUP: B78F13000700008.

Acronyms

AoA	Angle of Arrival
AmI	Ambient Intelligence
BN	Beacon Node
BS	Base Station
CI	Connectivity Information
HS	Harmony Search
LBS	Location-Based Service
LQI	Link Quality Indicator
k-NN	k-Nearest Neighbor
ML	Maximum Likelihood
Pr	Received Power
RF	Radio Frequency
RSSI	Received Signal Strength Indicator
SVDD	Support Vector Data Description
SVM	Support Vector Machine
TDoA	Time Difference of Arrival
WSN	Wireless Sensor Network

References

[1] G. Lo Re, M. Morana, M. Ortolani, Improving user experience via motion sensors in an ambient intelligence scenario., in: Pervasive and Embedded Computing and Communication Systems (PECCS), 2013, 2013, pp. 29–34.

[2] D. Marsh, R. Tynan, D. O'Kane, G. M. P. O'Hare, Autonomic wireless sensor networks, Engineering Applications of Artificial Intelligence 17 (7) (2004) 741 – 748.

[3] K. Langendoen, N. Reijers, Distributed localization in wireless sensor networks: a quantitative comparison, Computer Networks 43 (2003) 499–518.

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

[5] J. Wang, R. Ghosh, S. Das, A survey on sensor localization, Journal of Control Theory and Applications 8 (1) (2010) 2–11.

[6] Q. Song, Z. Ning, S. Wang, A. Jamalipour, Link stability estimation based on link connectivity changes in mobile ad-hoc networks, Journal of Network and Computer Applications 35 (6) (2012) 2051 – 2058.

[7] N. Aitsaadi, N. Achir, K. Boussetta, G. Pujolle, Artificial potential field approach in WSN deployment: Cost, QoM, connectivity, and lifetime constraints, Computer Networks 55 (1) (2011) 84–105.

[8] N. M. K. Chowdhury, R. Boutaba, A survey of network virtualization, Computer Networks 54 (5) (2010) 862 – 876.

[9] A. Lalomia, G. Lo Re, M. Ortolani, A hybrid framework for soft real-time wsn simulation, in: Proc. of the 13th IEEE/ACM International Symposium on Distributed Simulation and Real Time Applications, IEEE, 2009, pp. 201–207.

[10] A. De Paola, G. Lo Re, F. Milazzo, M. Ortolani, Adaptable data models for scalable ambient intelligence scenarios, in: Proc. of the IEEE International Conference on Information Networking (ICOIN), 2011, pp. 80–85.

[11] A. De Paola, S. Gaglio, G. Lo Re, M. Ortolani, Multi-sensor fusion through adaptive bayesian networks, in: AI*IA 2011: Artificial Intelligence Around Man and Beyond, Vol. 6934 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2011, pp. 360–371.

[12] J. Hightower, G. Borriello, Location systems for ubiquitous computing, IEEE Computer 34 (8) (2001) 57–66.

[13] P. Bahl, V. Padmanabhan, RADAR: an in-building RF-based user location and tracking system, in: Proceedings of IEEE INFOCOM 2000, the 19^{th} Annual Joint Conference of the IEEE Computer and Communications Societies, Vol. 2, 2000, pp. 775–784.

[14] T. He, C. Huang, B. M. Blum, J. A. Stankovic, T. Abdelzaher, Range-free localization schemes for large scale sensor networks, in: Proceedings of the 9^{th} Annual International Conference on Mobile Computing and Networking, MobiCom '03, ACM, New York, NY, USA, 2003, pp. 81–95.

[15] L. Doherty, K. Pister, L. El Ghaoui, Convex position estimation in wireless sensor networks, in: Proceedings of IEEE INFOCOM 2001, the 20^{th} Annual Joint Conference of the IEEE Computer and Communications Societies., Vol. 3, 2001, pp. 1655–1663 vol.3.

[16] S. Li, F. Qin, A dynamic neural network approach for solving nonlinear inequalities defined on a graph and its application to distributed, routing-free, range-free localization of WSNs, Neurocomputing 117 (2013) 72 – 80.

[17] A. Simonetto, G. Leus, Distributed maximum likelihood sensor network localization, IEEE Transactions on Signal Processing 62 (6) (2014) 1424–1437.

[18] A. Kuh, C. Zhu, D. Mandic, Sensor network localization using least squares kernel regression, in: Knowledge-Based Intelligent Information and Engineering Systems, Vol. 4253 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2006, pp. 1280–1287.

[19] F. Vanheel, J. Verhaevert, E. Laermans, I. Moerman, P. Demeester, A linear regression based cost function for wsn localization, in: Proc. of 19th Intl Conf on Software, Telecom. and Computer Networks (SoftCOM), IEEE, 2011, pp. 1–5.

[20] D. Manjarres, J. Del Ser, S. Gil-Lopez, M. Vecchio, I. Landa-Torres, S. Salcedo-Sanz, R. Lopez-Valcarce, On the design of a novel two-objective harmony search approach for distance- and connectivity-based localization in wireless sensor networks, Engineering Applications of Artificial Intelligence 26 (2) (2013) 669–676.

[21] X. Nguyen, M. I. Jordan, B. Sinopoli, A kernel-based learning approach to ad-hoc sensor network localization, ACM Transactions on Sensor Networks (2005) 134–152.

[22] B. Scholkopf, A. J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, Cambridge, MA, USA, 2001.

[23] D. A. Tran, T. Nguyen, Localization in wireless sensor networks based on support vector machines, IEEE Trans. on Parallel and Distributed Systems 19 (7) (2008) 981–994.

[24] S. Afzal, H. Beigy, A localization algorithm for large scale mobile wireless sensor networks: a learning approach, The Journal of Supercomputing 69 (1) (2014) 98–120.

[25] H. Wu, J. Chen, C. Wang, Y. Sun, A kernel-based localization approach in wireless sensor networks, in: Proc. of the 2^{nd} Intl. Conf. on Future Generation Communication and Networking, 2008, pp. 31–34.

[26] I. T. Haque, A sensor based indoor localization through fingerprinting, Journal of Network and Computer Applications 44 (2014) 220 – 229.

[27] D. J. Suroso, P. Cherntanomwong, P. Sooraksa, J. Takada, Fingerprint-based technique for indoor localization in wireless sensor networks using fuzzy c-means clustering algorithm, in: Proc. of the Intl Symposium on Intelligent Signal Processing and Communications Systems (ISPACS), 2011, pp. 1–5.

[28] C.-Y. Cheng, T.-H. Lan, C.-H. Chan, An improved localization algorithm with wireless heartbeat monitoring system for patient safety in psychiatric wards, Engineering Applications of Artificial Intelligence 26 (2) (2013) 905 – 912.

[29] S. Li, Y. Guo, Distributed consensus filter on directed switching graphs, International Journal of Robust and Nonlinear Control 25 (13) (2015) 2019–2040.

[30] S. Li, Y. Guo, Distributed source seeking by cooperative robots: All-to-all and limited communications, in: Robotics and Automation (ICRA), 2012 IEEE International Conference on, 2012, pp. 1107–1112.

[31] N. Atanasov, R. Tron, V. Preciado, G. Pappas, Joint estimation and localization in sensor networks, in: 53rd IEEE Annual Conference on Decision and Control (CDC), 2014, pp. 6875–6882.

[32] H.-T. Lin, C.-J. Lin, R. C. Weng, A note on Platt's probabilistic outputs for support vector machines, Machine Learning 68 (2007) 267–276.

[33] J. Shawe-Taylor, N. Cristianini, Kernel Methods for Pattern Analysis, Cambridge University Press, 2004.

[34] A. Boudries, M. Aliouat, P. Siarry, Detection and replacement of a failing node in the wireless sensors networks, Computers and Electrical Engineering 40 (2) (2014) 421 – 432.

[35] D. M. J. Tax, R. P. W. Duin, Support Vector Data Description, Machine Learning 54 (1) (2004) 45–66.

[36] P. F. Evangelista, M. J. Embrechts, B. K. Szymanski, Some properties of the gaussian kernel for one class learning, in: Proceedings of the 17th International Conference on Artificial Neural Networks, ICANN'07, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 269–278.

[37] M. Zuniga, B. Krishnamachari, Analyzing the transitional region in low power wireless links, in: Proceeding of First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004, pp. 517–526.

[38] K. Sohrabi, B. Manriquez, G. J. Pottie, Near ground wideband channel measurement in 800–1000 mhz, in: Proceeding of IEEE 49th Conference on Vehicular Technology, Vol. 1, 1999, pp. 571–574.

[39] G. Zanca, F. Zorzi, A. Zanella, M. Zorzi, Experimental comparison of RSSI-based localization algorithms for indoor wireless sensor networks, in: Proc. of the Workshop on Real-world Wireless Sensor Networks, ACM, 2008, pp. 1–5.

[40] P. A. Forero, A. Cano, G. B. Giannakis, Consensus-based distributed support vector machines, Journal of Machine Learning Research 11 (2010) 1663–1707.