Sensors **2015**, *15*, 12539-12559; doi:10.3390/s150612539

OPEN ACCESS
sensors
ISSN 1424-8220
www.mdpi.com/journal/sensors

Article

Target Localization in Wireless Sensor Networks Using Online Semi-Supervised Support Vector Regression

Jaehyun Yoo and H. Jin Kim *

Department of Mechanical and Aerospace Engineering, Seoul National University, 599 Gwanangno, Gwanak-gu, Seoul KS013, Korea; E-Mail: yjh5455@snu.ac.kr

* Author to whom correspondence should be addressed; E-Mail: hjinkim@snu.ac.kr; Tel.: +82-2-880-1907.

Academic Editor: Leonhard M. Reindl

Received: 13 February 2015 / Accepted: 18 May 2015 / Published: 27 May 2015

Abstract: Machine learning has been successfully used for target localization in wireless sensor networks (WSNs) due to its accurate and robust estimation against highly nonlinear and noisy sensor measurement. For efficient and adaptive learning, this paper introduces online semi-supervised support vector regression (OSS-SVR). The first advantage of the proposed algorithm is that, based on semi-supervised learning framework, it can reduce the requirement on the amount of the labeled training data, maintaining accurate estimation. Second, with an extension to online learning, the proposed OSS-SVR automatically tracks changes of the system to be learned, such as varied noise characteristics. We compare the proposed algorithm with semi-supervised manifold learning, an online Gaussian process and online semi-supervised colocalization. The algorithms are evaluated for estimating the unknown location of a mobile robot in a WSN. The experimental results show that the proposed algorithm is more accurate under the smaller amount of labeled training data and is robust to varying noise. Moreover, the suggested algorithm performs fast computation, maintaining the best localization performance in comparison with the other methods.

Keywords: semi-supervised learning; online support vector regression; wireless sensor network

1. Introduction

Localization is one of the most important issues for wireless sensor networks (WSNs), because many applications need to know where the data have been obtained. For example, in a target tracking application, the measurement data are meaningless without location information about where the data were obtained.

With recent advances in wireless communications and electronics, localization using RSSI (received signal strength indicator) has attracted interest in many works in the literature [1,2]. Dealing with highly nonlinear and noisy RSSI, machine learning methods, such as neural network [3], Q-learning [4] and supervised learning [5], achieve good estimation for target localization. Because all of these methods require a large amount of the labeled training data for high accuracy, however, significant effort, such as cost, time and human skill, is needed. For example, in the indoor localization where GPS (Global Positioning System) is not available, the labeled training data points have to be collected by the human operator.

Against the need for a large set of labeled training data, semi-supervised learning has been recently developed. For example, in localization or tracking using WSN, an acquisition of labeled data may involve repeatedly placing a target and measuring corresponding RSSI from the sensor nodes in the known locations. On the other hand, the unlabeled data can be easily collected by recording RSSI without the position information.

A popular method to exploit unlabeled data is a manifold regularization (or graph-based method) that captures an intrinsic geometric structure of the training data and eventually makes the estimator smooth along the manifold [6]. The existing graph-based semi-supervised learning algorithms are mainly focused on classification problems (e.g., [7–13]), while only a few regression problems, such as localization, can be found in [14–16].

An advantage of the proposed method is its extension to online learning. Batch learning algorithms [16,17] cannot adjust to environmental changes, because they initially learn data only once. For example, in localization or tracking using WSN, the batch learning algorithms are not robust to changes in noise characteristics. To improve the robustness, these batch algorithms must be retrained from scratch, which has an expensive computational cost. To solve this problem, online learning methods, such as [18], are developed in order to update the learning model sequentially with the arrival of a new data point. They automatically track changes of the system model.

Motivated by the need for online learning, we extend the semi-supervised support vector regression (SVR) to the online learning framework that takes advantage of both semi-supervised learning and online learning. Online semi-supervised SVR (OSS-SVR) is accurate in that its solution is equivalent to the batch semi-supervised SVR. Contrary to other online algorithms [18,19], moreover, OSS-SVR can provide information on which data points are more important using support values [20]. These values are utilized when removing data in order to manage data storage, maintaining the good estimation performance.

We evaluate the developed algorithm (OSS-SVR) to estimate an unknown location of a mobile robot in a WSN, by comparing with two existing semi-supervised learning algorithms (*i.e.*, semi-supervised colocalization [14], semi-supervised manifold learning [15]) and one online learning algorithm using a

Gaussian process [19]. In comparison with these state-of-the-art methods, we confirm that our algorithm estimates the location most accurately with fewer labeled data by efficiently exploiting unlabeled data. Furthermore, OSS-SVR converges most rapidly to the best estimated trajectory when the environment is changed. The computation of the proposed algorithm is very fast, maintaining the best accuracy.

This paper is organized as follows. In Section 2, we review some existing work for localization in a wireless sensor network. Section 3 overviews the technical structure for our localization problem. Section 4 presents the semi-supervised support vector machine algorithm. Section 5 describes an extension to the online version from the batch algorithm in Section 4. Section 6 reports empirical results with the description of the comparative algorithms, the parameter settings and the performance indices. Finally, concluding remarks are given in Section 7.

2. Related Work

Machine learning for localization aims to learn a model that defines the mapping function between sensor measurements and the location of a target of interest. However, due to the nonlinear and noisy characteristics of sensors, such as RSSI, sufficient labeled training data are needed for a good estimation. Furthermore, the algorithm should be able to update a learning model upon arrival of a new data point, in order to adjust to dynamic environments. We review some existing work based on whether they adopt the semi-supervised learning or online learning for the localization as follows.

Yang *et al.* [15] develop a semi-supervised manifold learning that solves the optimization problem based on a least squares loss function and graph Laplacian. Because it solves simplified linear optimization, it is fast, but inaccurate. Moreover, an online version of this algorithm has not been developed.

There is an online method based on a Gaussian process (GP) [19]. Since the online GP of [19] uses only current sensor measurements as training data, it is fast and naturally online. However, a strong assumption of known sensor locations is needed.

An online and semi-supervised learning method named colocalization by Pan *et al.* [14] estimates target location, as well as the locations of sensor nodes. They merge singular value decomposition, graph Laplacian and the loss function into one optimization problem. However, many tuning parameters that affect the optimal solution can be a burden. Furthermore, when the algorithm is applied to online learning, it updates the model by approximation using harmonic functions [13]. Because the approximated model is different from the solution using batch optimization, it can lose accuracy.

We introduce an SVR-based learning algorithm, which takes advantage of both online learning and semi-supervised learning methods. It does not need the assumption of known sensor locations and provides a unique solution and sparseness [20], contrary to an artificial neural network and reinforcement learning [21]. It performs accurate and efficient online learning based on incremental and decremental algorithms. The incremental algorithm learns the model given a new data point, providing the same solution to the batch optimization. The decremental algorithm safely removes worthless or less important data in order to limit the data size.

3. Overview

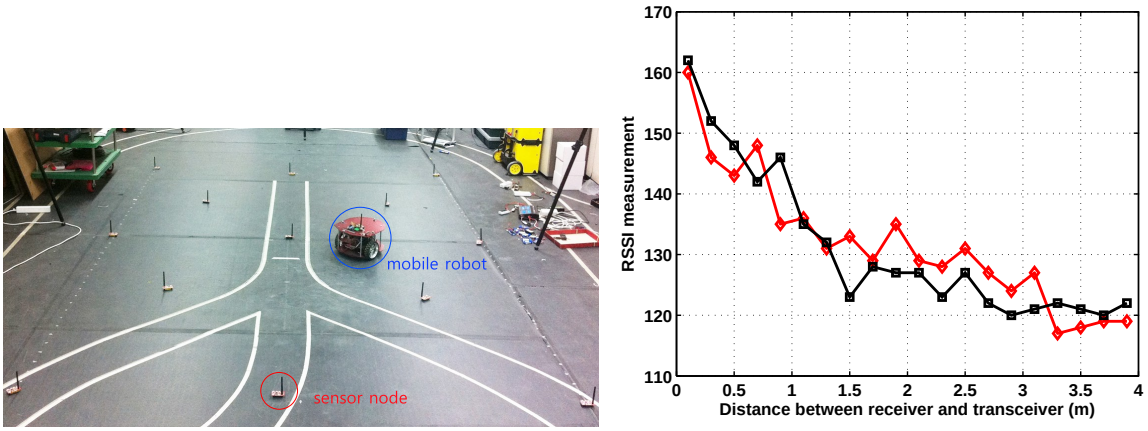
This section overviews the problem formulation, including the experimental setup, the data collection, the localization structure and the extended localization structure for a mobile target. In advance, we summarize the notations of functions and variables that are used in this paper:

$i \in \{1, \ldots, l\}$	:	an index in a labeled training dataset when a target is located at l different locations
$y_{\mathrm{X}i} \in \mathcal{R}$	:	the i-th target location in X coordinate
$y_{\mathrm{Y}i} \in \mathcal{R}$	:	the i-th target location in Y coordinate
$z_{ij} \in \mathcal{R}$	:	an RSSI measurement of j-th sensor node corresponding to the i-th target location $y_{\mathrm{X}i}, y_{\mathrm{Y}i}$
$r_i = \{z_{i1}, \ldots, z_{in}\} \in \mathcal{R}^n$	:	an RSSI observation set collected from all n sensor nodes
$\{r_i, y_{\mathrm{X}i}\}_{i=1}^{l}$	:	a labeled dataset for X coordinate
$\{r_i, y_{\mathrm{Y}i}\}_{i=1}^{l}$	:	a labeled dataset for Y coordinate
$\{r_i\}_{j=1}^{u}$	:	an unlabeled dataset with the number of u data points
$f_{\mathrm{X}}\left(r \rightarrow y_{\mathrm{X}}\right)$	:	the function mapping an RSSI measurement set r to X coordinate position
$f_{\mathrm{Y}}\left(r \rightarrow y_{\mathrm{Y}}\right)$	:	the function mapping an RSSI measurement set r to Y coordinate position.

3.1. Experimental Setup

Figure 1a shows the localization setup where 13 static sensor nodes are fixed in the 3 m $\times$ 4 m workspace. There is another base node (or base station) that is wirelessly connected to all deployed nodes. The role of the base station is to collect and record RSSI from the deployed nodes and to estimate a target position. A mobile robot equipped with one node broadcasts an RF signal whose strength (RSSI) is recorded by the sensor nodes. For accuracy analysis, the true location of the mobile robot is measured by the Vicon motion system that tracks reflective markers attached to the mobile robot.

The radio signal is easily influenced by the environment. Furthermore, its measured value affects the performance of the localization. Here, we examine the characteristics of RSSI measurement. In Figure 1b, we record RSSI measurements of one moving node (receiver), emitted by one fixed sensor node (transceiver). The distance between the transceiver and the receiver varies from 0.2 m to 4 m. The results of two trials show nonlinear and noisy RSSI measurements, which supports the need for learning to obtain accurate position estimates.



(a) **(b)**

Figure 1. (**a**) Experimental setup with one mobile robot and thirteen sensor nodes. The nodes use a commercial CC2420 radio chip, which provides an IEEE 802.15.4 communication, and a received signal strength indicator (RSSI) for each received packet; (**b**) The relationship between RSSI measurement and distance between one fixed sensor node and one mobile node. The two trials are marked by a black-squared line and a red-diamond line, respectively.

3.2. Data Acquisition

Semi-supervised learning uses both labeled and unlabeled datasets. For obtaining the labeled training data, *i.e.*, two sets $\{r_i, y_{\mathrm{X}i}\}_{i=1}^{l}$, $\{r_i, y_{\mathrm{Y}i}\}_{i=1}^{l}$, we collect RSSI measurements of the mobile robot placed at the different l locations, repeatedly. This collection process for the labeled dataset is called fingerprinting. On the other hand, the unlabeled dataset, *i.e.*, $\{r_j\}_{j=1}^{u}$, is obtained as we let the mobile robot move autonomously and collect only the RSSI measurements. Because the unlabeled dataset does not include the labels, *i.e.*, y_{X} and y_{Y}, it is easy to collect a large amount of unlabeled data points. We note that the training dataset does not include the positions of the sensor nodes. Therefore, our algorithm does not need the positions of the sensor nodes.

3.3. Localization Structure

Our algorithm for the localization consists of three steps, *i.e.*, offline learning phase, test phase and online learning phase.

3.3.1. Offline Learning Phase

Given the labeled and unlabeled training dataset, the major output of the training phase is two mapping functions f_{X} and f_{Y}, which represent a relationship between RSSI observation set and the 2D position of a target. The details to obtain those functions will be described in Section 4.

3.3.2. Test Phase

Test data $r^* \in \mathcal{R}^n$ are defined as an RSSI measurement set obtained from all n sensor nodes. Then, a base station estimates the location of the target using the test data r^* by using the learned models f_{X} and f_{Y}.

3.3.3. Online Learning Phase

This paper considers a situation where new labeled data points are available after the offline learning phase. For example, landmark-based applications using RFID [22] or vision sensors [23] can obtain the online labeled dataset. The purpose of the online learning phase is to update the model f_{X} and f_{Y} when a new labeled data point comes. The detailed description is shown in Section 5.

3.4. Combination with Kalman Filter

In most target localization, targets move. However, the training datasets in the fingerprinting method do not include the velocity information of the target, because the training data are collected when the robot stops. One way to consider a moving target without the exact velocity measurement and without modifying the fingerprinting method is using a target's dynamic model in the framework of the Kalman filter. By defining the observation model as the estimated location obtained from the fingerprinting method, Kalman filter-based fingerprinting localization can be formulated. In Section 6.5, we compare the experimental results of the Kalman filter-based fingerprinting methods.

4. Semi-Supervised SVR

This section describes semi-supervised support vector regression. The models f_{X} and f_{Y} defined in the previous section are learned independently. For simplification, from this section, we omit the subscripts of f_{X}, f_{Y} and y_{X}, y_{Y}.

4.1. Semi-Supervised Learning Framework

Given a set of l labeled samples $\{(r_i, y_i)\}_{i=1}^{l}$ and a set of u unlabeled samples $\{r_i\}_{i=l+1}^{l+u}$, semi-supervised learning aims to establish a mapping f by the following regularized minimization functional:

$$f^* = \arg\min_{f \in \mathcal{H}_k} \frac{1}{l} \sum_{i=1}^{l} V(r_i, y_i, f) + \gamma_A \|f\|_A^2 + \gamma_I \|f\|_I^2 \tag{1}$$

where V is a loss function that will be defined in Equation (7), $\|f\|_A^2$ in Equation (3) is the norm of the function in the reproducing kernel Hilbert space (RKHS), $\|f\|_I^2$ in Equation (5) is the norm of the function in the low dimensional manifold and γ_A, γ_I are the regularization weight parameters.

By the representer theorem [24], the solution of Equation (1) can be defined as an expansion of kernel function over the labeled and the unlabeled data, given by:

$$f(x) = \sum_{i=1}^{l+u} \alpha_i K(x_i, x) + b \tag{2}$$

with the bias term b and the kernel function $K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$, where $\phi(\cdot)$ is a nonlinear mapping to the RKHS.

The regularization term $\|f\|_A^2$, which is associated with the RKHS, is defined as:

$$\|f\|_A^2 = (\Phi\alpha)^T(\Phi\alpha) = \alpha^T K \alpha \tag{3}$$

where $\Phi = [\phi(x_1), \ldots, \phi(x_{l+u})]$, $\alpha = [\alpha_1, \ldots, \alpha_{l+u}]^T$ and K is the $(l+u) \times (l+u)$ kernel matrix whose element is K_{ij}. We adopt the Gaussian kernel given by:

$$K_{ij} = K(r_i, r_j) = \exp\left(-\|r_i - r_j\|^2 / \sigma_k^2\right) \tag{4}$$

where σ_k^2 is the kernel width parameter.

According to the manifold regularization, data points are samples from a low-dimensional manifold embedded in a high-dimensional space. This is represented by the graph Laplacian and the kernel function [6]:

$$\begin{aligned} \|f\|_I^2 &= \frac{1}{(l+u)^2} \sum_{i=1}^{l+u} \sum_{j=1}^{l+u} W_{ij} \left(f(r_i) - f(r_j)\right)^2 \\ &= \frac{1}{(l+u)^2} \mathbf{f}^T L \mathbf{f} \end{aligned} \tag{5}$$

where L is the graph Laplacian given by $L = D - W$, $\mathbf{f} = [f(r_1), \ldots, f(r_{l+u})]^T$, W is the adjacency matrix of the data graph and D is the diagonal matrix given by $D_{ii} = \sum_{j=1}^{l+u} W_{ij}$. In general, the edge weights W_{ij} are defined as a Gaussian function of the Euclidean distance, given by:

$$W_{ij} = \exp\left(-\|r_i - r_j\|^2 / \sigma_w^2\right) \tag{6}$$

where σ_w^2 is the kernel width parameter.

Minimizing $\|f\|_I^2$ is equivalent to penalizing the rapid changes of the regression function evaluated between two data points. Therefore, $\gamma_I \|f\|_I^2$ in Equation (1) controls the smoothness of the data geometric structure.

We employ the following ϵ-insensitive loss function:

$$V(r_i, y_i, f) = \begin{cases} 0 & \text{if } |f(r_i) - y_i| < \epsilon \\ |f(r_i) - y_i| - \epsilon & \text{otherwise} \end{cases} \tag{7}$$

Minimizing the loss function Equation (7) aims to bound the estimated error $|f(r_i) - y_i|$ within the margin ϵ.

4.2. Complete Formulation

In this subsection, the complete formulation of the semi-supervised support vector regression is given based on the ingredients from Equations (2), (3), (5) and (7) to Equation (1). We additionally introduce slack variables ξ_i, ξ_i^* along with the ϵ-insensitive loss function in order to cope with infeasible constraints of the optimization problem that uses only the ϵ constraint. Substituting Equations (2) to (7) into Equation (1) with the slack variables ξ_i, ξ_i^*, the primal problem is defined as:

$$\begin{aligned} \min_{\alpha \in \mathcal{R}^{l+u}, \xi \in \mathcal{R}^l, \xi^* \in \mathcal{R}^l} & \frac{1}{l} \sum_{i=1}^{l} (\xi_i + \xi_i^*) + \gamma_A \alpha^T K \alpha \\ & + \frac{1}{(l+u)^2} \gamma_I \alpha^T K L K \alpha \\ \text{subject to: } & y_i - \sum_{j=1}^{l+u} \alpha_j K_{ij} - b \leq \epsilon + \xi_i \\ & \sum_{j=1}^{l+u} \alpha_j K_{ij} + b - y_i \leq \epsilon + \xi_i^* \\ & \xi_i, \xi_i^* \geq 0, \quad i = 1, \ldots, l \end{aligned} \tag{8}$$

After the introduction of four sets of l multipliers β, β^*, η and η^*, the Lagrangian H associated with the problem is:

$$\begin{aligned} & H(\alpha, b, \xi, \xi^*, \beta, \beta^*, \eta, \eta^*) = \\ & \frac{1}{2} \alpha^T \left(2\gamma_A K + \frac{2}{(l+u)^2} \gamma_I K L K \right) \alpha + \frac{1}{l} \sum_{i=1}^{l} (\xi_i + \xi_i^*) \\ & - \sum_{i=1}^{l} \beta_i \left(\epsilon + \xi_i - y_i + \sum_{j=1}^{l+u} \alpha_j K_{ij} + b \right) \\ & - \sum_{i=1}^{l} \beta_i^* \left(\epsilon + \xi_i^* + y_i - \sum_{j=1}^{l+u} \alpha_j K_{ij} - b \right) \\ & - \frac{1}{l} \sum_{i=1}^{l} (\eta_i \xi_i + \eta_i^* \xi_i^*) \end{aligned} \tag{9}$$

In order to convert the primal problem to the dual representation, we take derivatives:

$$\frac{\partial H}{\partial b} = \sum_{i=1}^{l} (\beta_i - \beta_i^*) = 0 \tag{10}$$

$$\frac{\partial H}{\partial \xi_i^{(*)}} = \frac{1}{l} - \beta_i^{(*)} - \frac{1}{l} \eta_i^{(*)} = 0 \tag{11}$$

where $\beta_i^{(*)}$ denotes both β_i and β_i^*.

Using Equations (10) and (11), we can rewrite the Lagrangian as a function of only α, β and β^* from Equation (9), given by:

$$\begin{aligned} & H(\alpha, \beta, \beta^*) = \\ & \frac{1}{2}\alpha^T \left(2\gamma_A K + \frac{2}{(l+u)^2}\gamma_I K L K\right)\alpha - \alpha^T K J_L^T \mathcal{B} \\ & \qquad -\epsilon \sum_{i=1}^{l}(\beta_i + \beta_i^*) + \sum_{i=1}^{l} y_i(\beta_i - \beta_i^*) \\ & \text{subject to :} \quad \sum_{i=1}^{l}(\beta_i - \beta_i^*) = 0 \\ & \qquad 0 \le \beta_i, \beta_i^* \le \frac{1}{l}, \quad i = 1, \ldots, l \end{aligned} \tag{12}$$

here, $J_L = [I_{l\times l}\ \vec{0}_{l\times u}]$ is an $l \times (l+u)$ matrix, where $I_{l\times l}$ is the $l \times l$ identity matrix and $\vec{0}_{l\times u}$ is the $l \times u$ zero matrix, and $\mathcal{B} = [\beta_1 - \beta_1^*, \ldots, \beta_l - \beta_l^*]^T$.

Taking derivatives with respect to α, we obtain:

$$\frac{\partial H}{\partial \alpha} = \left(2\gamma_A K + \frac{2}{(l+u)^2}\gamma_I K L K\right)\alpha - K J_L^T \mathcal{B} = 0 \tag{13}$$

From Equation (13), a direct relationship between α and $\mathcal{B}$ is obtained as follows:

$$\alpha = \left(2\gamma_A I + 2\frac{1}{(l+u)^2}\gamma_I L K\right)^{-1} J_L^T \mathcal{B} \tag{14}$$

where I is the $(l+u) \times (l+u)$ identity matrix. Substituting Equation (14) back into the Lagrangian Equation (12), we arrive at the convex optimization problem:

$$\begin{aligned} \max_{\beta\in\mathcal{R}^l, \beta^*\in\mathcal{R}^l} \ & -\frac{1}{2}\mathcal{B}^T J_L K \left(2\gamma_A I + \frac{2}{(l+u)^2}\gamma_I L K\right)^{-1} \\ & J_L^T \mathcal{B} - \epsilon \sum_{i=1}^{l}(\beta_i + \beta_i^*) + \sum_{i=1}^{l} y_i(\beta_i - \beta_i^*) \\ & \text{subject to :} \quad \sum_{i=1}^{l}(\beta_i - \beta_i^*) = 0 \\ & \qquad 0 \le \beta_i, \beta_i^* \le \frac{1}{l}, \quad i = 1, \ldots, l \end{aligned} \tag{15}$$

The optimal solution $\mathcal{B}^*$ of the above quadratic program is linked to the optimal α^* in Equation (14). Then, the optimal regression function $f^*(x)$ can be obtained by substituting $\alpha^* = [\alpha_1^*, \ldots, \alpha_{l+u}^*]^T$ into Equation (2). In summary, for the batch algorithm with given labeled and unlabeled datasets, the basic steps for obtaining the weights α^* are: (I) construct edge weights W_{ij} and the graph Laplacian $L = D - W$; (II) build a kernel function K; (III) choose regularization parameters γ_A and γ_I; and (IV) solve the quadratic programming Equation (15).

5. Online Semi-Supervised SVR

This section extends the semi-supervised batch SVR described in the previous section to online semi-supervised SVR. We use the idea from [25] for online extension, *i.e.*, adding or removing new data, maintaining the satisfaction of the Karush–Kuhn–Tucker (KKT) conditions. The resulting solution has the equivalent form to the batch version; thus, it does not learn the entire model again, which allows a much faster model update.

5.1. Karush–Kuhn–Tucker Conditions of Semi-Supervised SVR

We define a margin function $h(r_i)$ for the i-th data (r_i, y_i) as:

$$h(r_i) = f(r_i) - y_i = \sum_{j=1}^{l} P_{ij}\mathcal{B}_j - y_i + b \tag{16}$$

where $P = KQ$ and $Q = \left(2\gamma_A I + \frac{2}{(l+u)^2}\gamma_I LK\right)^{-1}$ Furthermore, the (i,j)-th element of the matrix P is defined as:

$$P(r_i, r_j) = P_{ij} = \sum_{k=1}^{l+u} K_{ik}Q_{kj} \tag{17}$$

where $K_{ik} = K(r_i, r_k)$ and $Q_{kj} = Q(r_k, r_j)$.

The Lagrange formulation of Equation (15) can be represented as:

$$\begin{aligned} L_D &= \frac{1}{2}\sum_{i=1}^{l}\sum_{j=1}^{l} P_{ij}(\beta_i - \beta_i^*)(\beta_j - \beta_j^*) + \epsilon\sum_{i=1}^{l}(\beta_i + \beta_i^*) \\ &- \sum_{i=1}^{l} y_i(\beta_i - \beta_i^*) - \sum_{i=1}^{l}(\delta_i\beta_i + \delta_i^*\beta_i^*) \\ &+ \sum_{i=1}^{l}\mu_i(\beta_i - \frac{1}{l}) + \sum_{i=1}^{l}\mu_i^*(\beta_i^* - \frac{1}{l}) \\ &+ b\sum_{i=1}^{l} y_i(\beta_i - \beta_i^*) \end{aligned} \tag{18}$$

Computing partial derivatives of the Lagrangian L_D leads to the KKT conditions:

$$\frac{\partial L_D}{\partial \beta_i} = \sum_{j=1}^{l} P_{ij}(\beta_j - \beta_j^*) + \epsilon - y_i + b - \delta_i + \mu_i = 0 \tag{19}$$

$$\frac{\partial L_D}{\partial \beta_i^*} = \sum_{j=1}^{l} P_{ij}(\beta_j - \beta_j^*) + \epsilon + y_i - b - \delta_i^* + \mu_i^* = 0 \tag{20}$$

$$\frac{\partial L_D}{\partial b} = \sum_{j=1}^{l}(\beta_j - \beta_j^*) = 0 \tag{21}$$

$$\delta_i^{(*)} \geq 0, \quad \delta_i^{(*)}\beta_i^{(*)} = 0 \tag{22}$$

$$\mu_i^{(*)} \geq 0, \quad \mu_i^{(*)} \left(\beta_i^{(*)} - \frac{1}{l} \right) = 0 \tag{23}$$

$$0 \leq \beta_i^{(*)} \leq \frac{1}{l} \tag{24}$$

we recall $\mathcal{B}_i = \beta_i - \beta_i^*$ in Equation (12) and define a margin function $h(x_i)$ for the i-th data (x_i, y_i) as:

$$h(r_i) = f(r_i) - y_i = \sum_{j=1}^{l} P_{ij}\mathcal{B}_j - y_i + b \tag{25}$$

By combining all of the KKT conditions from Equations (19) to (24), we can obtain the following:

$$\begin{cases} h(r_i) > \epsilon, & \mathcal{B}_i = -\frac{1}{l} \\ h(r_i) = \epsilon, & -\frac{1}{l} < \mathcal{B}_i < 0 \\ -\epsilon < h(x_i) < \epsilon, & \mathcal{B}_i = 0 \\ h(r_i) = -\epsilon, & 0 < \mathcal{B}_i < \frac{1}{l} \\ h(r_i) < -\epsilon, & \mathcal{B}_i = \frac{1}{l} \end{cases} \tag{26}$$

According to Karush–Kuhn–Tucker (KKT) conditions, we can separate labeled training samples into three subsets:

$$\begin{array}{ll} \text{Support set} & S = \{(r_i, y_i) \mid 0 < |\mathcal{B}_i| < \frac{1}{l},\ |h(r_i)| = \epsilon\} \\ \text{Error set} & E = \{(r_i, y_i) \mid |\mathcal{B}_i| = \frac{1}{l},\ |h(r_i)| > \epsilon\} \\ \text{Remaining set} & R = \{(r_i, y_i) \mid |\mathcal{B}_i| = 0,\ |h(r_i)| < \epsilon\} \end{array} \tag{27}$$

5.2. Adding a New Sample

Let us denote a new sample and corresponding coefficient by (r_c, y_c), $\mathcal{B}_c$ whose initial value is set to zero. When the new sample is added, $\mathcal{B}_i$ (for $i = 1, \cdots, l$), b and $\mathcal{B}_c$ are updated. The variation of the margin is given by:

$$\Delta h(r_i) = \sum_{j=1}^{l} P_{ij}\Delta\mathcal{B}_j + P_{ic}\Delta\mathcal{B}_c + \Delta b \tag{28}$$

The sum of all of the coefficients should remain zero according to Equation (10), which, in turn, can be written as:

$$\Delta\mathcal{B}_c + \sum_{i=1}^{l} \Delta\mathcal{B}_i = 0 \tag{29}$$

By the KKT conditions in Equation (27), only the support set samples can change $\mathcal{B}_i$. Furthermore, for the support set samples, the margin function is always ϵ, so the variation of the margin function is zero. Thus, Equations (28) and (29) can be represented to discover the values of the variations of $\mathcal{B}$ and b, in the following:

$$\begin{bmatrix} \Delta b & \Delta\mathcal{B}_{s_1} & \cdots & \Delta\mathcal{B}_{s_{l_s}} \end{bmatrix}^T = \kappa\Delta\mathcal{B}_c \tag{30}$$

where:

$$\kappa = \begin{bmatrix} \kappa_b & \kappa_{s_1} & \cdots & \kappa_{s_{l_s}} \end{bmatrix}^T \tag{31}$$

$$= -M \begin{bmatrix} 1 & P_{s_1 c} & \cdots & P_{s_{l_s} c} \end{bmatrix}^T \tag{32}$$

and:

$$M = \begin{bmatrix} 0 & 1 & \cdots & 1 \\ 1 & P_{s_1 s_1} & \cdots & P_{s_1 s_{l_s}} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & P_{s_{l_s} s_1} & \cdots & P_{s_{l_s} s_{l_s}} \end{bmatrix}^{-1} \tag{33}$$

This states that we can update $\mathcal{B}_i$ for $(r_i, y_i) \in S$ and b when $\mathcal{B}_c$ is given. Moreover, $h(r_i)$ for $(r_i, y_i) \in S$ are consistent according to Equation (27).

5.3. Recursive Update of Inverse Matrix

Calculating the inverse matrix Equation (33) can be inefficient. Fortunately, Ma *et al.* [25] and Martin [26] suggest a recursive algorithm to update M Equation (33) without explicitly computing the inverse. This method also can be used for our algorithm. When a sample (r_i, y_i) is added to the set S, the new matrix M can be updated in the following:

$$M^{new} = \left[\begin{array}{c|c} M & \vec{0} \\ \hline \vec{0}^T & 0 \end{array}\right] + \frac{1}{\nu_i} \begin{bmatrix} \kappa_M \\ 1 \end{bmatrix} \begin{bmatrix} \kappa_M^T & 1 \end{bmatrix} \tag{34}$$

where:

$$\kappa_M = -M \begin{bmatrix} 1 & P_{is_1} & \cdots & P_{is_{l_s}} \end{bmatrix}^T \tag{35}$$

$$\nu_i = P_{ii} + \begin{bmatrix} 1 & P_{is_1} & \cdots & P_{is_{l_s}} \end{bmatrix} \kappa_M \tag{36}$$

and $\vec{0}$ is the $(l_s + 1) \times 1$ zero vector.

When the k-th sample (r_k, y_k) in the set S is removed, M can be updated by:

$$M_{ij}^{new} = M_{ij} - M_{ik} M_{kj} / M_{kk} \tag{37}$$

for $i, j \in [1, \cdots, k, k+2, \cdots, l_s + 1]$.

5.4. Incremental and Decremental Algorithms

Since the standard online SVR was developed in [25], many other researches, such as [27–29], had followed its incremental and decremental algorithms that update the variation $\Delta\mathcal{B}_c$ caused by a new sample or a removed sample, for the quick termination of the online learning. The only difference in comparison with the existing online SVR is making the kernel matrix K and the graph Laplacian L before starting the incremental and decremental algorithms. Therefore, the same incremental and decremental algorithms can be used once we build K and L. We skip the details (see Section 3.2 and Appendix in [25]) to avoid overlap.

We close this section with remarks on the usages of the incremental and decremental algorithms. The incremental algorithm is used whenever a new data point comes in. However, maintaining all data whose volume increases over time is inefficient for calculation speed and memory. As a solution to this problem, the decremental algorithm is used to limit the data size by forgetting useless or less important data. The concept of support vectors offers a natural criterion for this. The useless dataset is defined as the training

data in the set R. Furthermore, less important data are defined as the training data within $S \cup E$, which have small coefficients $\mathcal{B}_i$ that slightly affect the margin function in Equation (25). The relationship between localization performance and calculation time, when adopting the decremental algorithm, will be analyzed in Section 6.3.

6. Experiments

For evaluation of the online learning algorithms, we assume that the target location corresponding RSSI measurements at any time are available as labeled training data.

This section describes the comparative algorithms in Section 6.1 and the parameter setting in Section 6.2. In Section 6.3, localization results are shown when we vary the initial training data amount and intentionally change an environment. In Section 6.4, the results are shown when we apply decremental algorithm. In Section 6.5, Kalman filter-based fingerprinting methods are compared.

6.1. Comparative Algorithms

The proposed algorithm is compared with other recently-developed learning algorithms, summarized in Table 1.

Table 1. Comparative algorithms. SVR, support vector regression.

Learning Method	**Semi-Supervised**	**Online**
Semi-Supervised Manifold Learning [15]	◯	×
Gaussian Process [19]	×	◯
Semi-Supervised Colocalization [14]	◯	◯
Online Semi-Supervised SVR (developed)	◯	◯

6.1.1. Semi-Supervised Colocalization

Similarly to our algorithm, this builds an optimization problem with a loss function and graph Laplacian. As a training set, semi-supervised colocalization (SSC) uses target location, locations of sensor nodes and RSSI measurements. Given RSSI measurements as test data, SSC estimates the location of the target and also the locations of the sensor nodes in order to recover unknown locations of the sensor nodes in the training set. For better target tracking performance, many labeled (known) locations of sensor nodes are needed.

6.1.2. Gaussian Process

For RSSI-based localization, a Gaussian process makes a probabilistic distribution that represents where the target is located. Similar to SSC, the online GP of [19] also needs an assumption that the locations of all sensor nodes are known, while the SVR-based method does not need to know the sensor

locations. We give SSC and GP advantageous information of the locations of the sensor nodes for all experiments. Localization using GP in a wireless sensor network is studied in our previous work.

6.1.3. Semi-Supervised Manifold Learning

Semi-supervised manifold learning (SSML) is extended from Laplacian regularized least squares (LapRLS [30]) by modifying a classification problem to a regression problem. Because this method uses fast linear optimization, it can be suitable for real-time applications. However, this is inaccurate, and only the batch version has been reported. In order to compare online tracking performance dealing with new incoming data, we perform SSML whenever a new data point comes, enduring a long calculation time.

6.2. Parameter Setting

All tuning parameters are optimized by 10-fold cross-validation [31] using 40 labeled data and 30 unlabeled data. We set optimal kernel parameters minimizing the training error. Figure 2 shows the training error curves of the used algorithms with respect to the tuning parameters. The training error is defined as CV (cross-validation) error, given by:

$$\text{CV error} = \frac{1}{K}\sum_{k=1}^{K} E_k(\lambda_1, \lambda_2, \ldots)$$

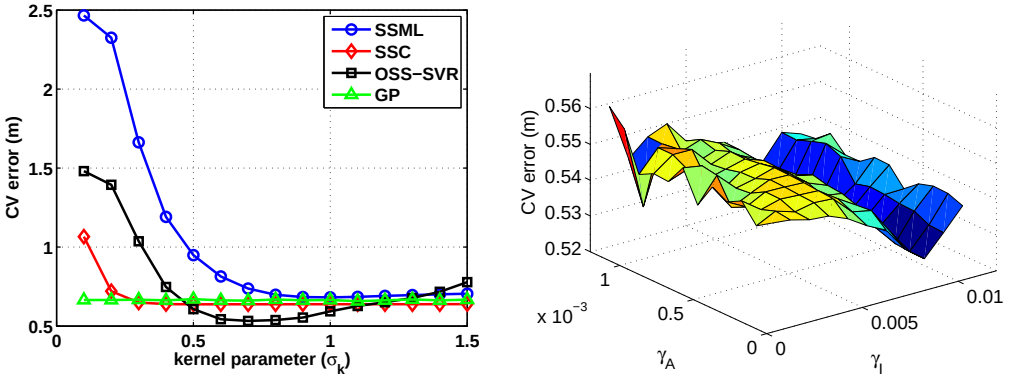
$$E_k(\lambda_1, \lambda_2, \ldots) = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \|\hat{y}_i - y_i\|_2^2}$$

where $K = 10$ is the number of the fold, E_k is the RMSE (root mean squared error) of one fold, N is the number of data in one fold, $\hat{y}_i$ is the estimated location, y_i is the true location and $\|\cdot\|_2$ denotes the vector two-norm. Variables of CV error are defined as $\lambda_1 = \sigma_k$ in Figure 2a and $\lambda_1 = \gamma_I$, $\lambda_2 = \gamma_A$ in Figure 2b.

All of the compared learning algorithms use common kernel function Equation (4). The kernel parameter σ_k in Equation (4) is selected to be smaller than 1.5, avoiding extremely small values, which yield large RMSE in all of the methods, as shown in Figure 2a. The parameter ϵ in Equation (7) is set to 0.02, after the similar analysis in the range [0.01, 1]. Small changes in the tuning parameters do not affect training error much.

The parameters γ_I and γ_A are additional parameters for OSS-SVR, as described in Section 4. As γ_I increases, the influence of unlabeled data increases by Equation (1). Therefore, γ_I determines the impact of unlabeled data. As can be seen in Figure 2b, γ_I is more sensitive than γ_A. Therefore, we find a proper range of γ_A first and, then, select good values for γ_I, γ_A.

In general, the graph Laplacian Equation (6) is built with k-nearest neighbor data points. When the graph Laplacian with k-nearest neighbors is updated online with new data, however, calculation of the new graph Laplacian is quite slow, because it has to find new neighbors and make new kernel matrices. This has to be repeated while $\Delta\mathcal{B}_i$, Δb, $\Delta\mathcal{B}_c$ and $\Delta h(x_i)$ are being updated using Equation (30). To avoid this complex process, we build the graph Laplacian with all data points instead of the k-nearest data points. Therefore, we can quickly update the graph Laplacian with the additional advantage of no need for finding optimal parameter k.



(a) **(b)**

Figure 2. CV (cross-validation) error over the validation set as (**a**) a function of the kernel parameter for all compared methods and (**b**) a function of regularization parameters γ_I, γ_A for online semi-supervised (OSS)-SVR.

6.3. Localization Results for a Circular Trajectory

In this section, we test localization algorithms where the target moves along a circular path. First, we show the online tracking results of each algorithm with respect to the repeated target motion. Second, we compare performance varying the amount of initial training data. Third, we examine localization performance when the system model is disturbed by bias noise.

First, Figure 3 shows the online tracking result when no initial labeled data and 30 initial unlabeled data points are used for each algorithm. Moreover, a new labeled data point is learned per time step. The number on the title of each figure indicates how many laps the target has moved. Overall, localization performances, except GP, tend to improve over time, because they learn the repeated relationship between RSSI measurements and target positions. In this experiment, we compare the capability of the algorithms for accuracy and speed of convergence to the true trajectory. In Figure 3, although 10 repetitions passed, SSC, SSML and GP do not approach the true trajectory. On the other hand, our algorithm gets close to the true trajectory in only four repetitions. In comparison with all of the others, the proposed online semi-supervised SVR (OSS-SVR) gives the best accurate localization with the fastest convergence.

Next, we vary the initial number of labeled training data and show localization error during two laps of the target motion. Localization error is defined as the root mean squared error (RMSE), given by:

$$\text{RMSE} = \sqrt{1/T \sum_{t=1}^{T} \|\hat{y}_t - y_t\|_2^2} \tag{38}$$

where $\|\cdot\|_2$ denotes the vector two-norm, $\hat{y}_t$ is the estimated location, y_t is the true location at time step t and T is the duration of two laps of the target moving. For all of the methods, the same unlabeled data (total 30 points) are used along with the randomly-picked labeled data. Figure 4a shows the mean-standard deviation over 10 repetitions in order to reduce the statistical variability of the randomly-picked labeled data. We can observe that our method gives the smallest error and deviation over the variation of labeled data.

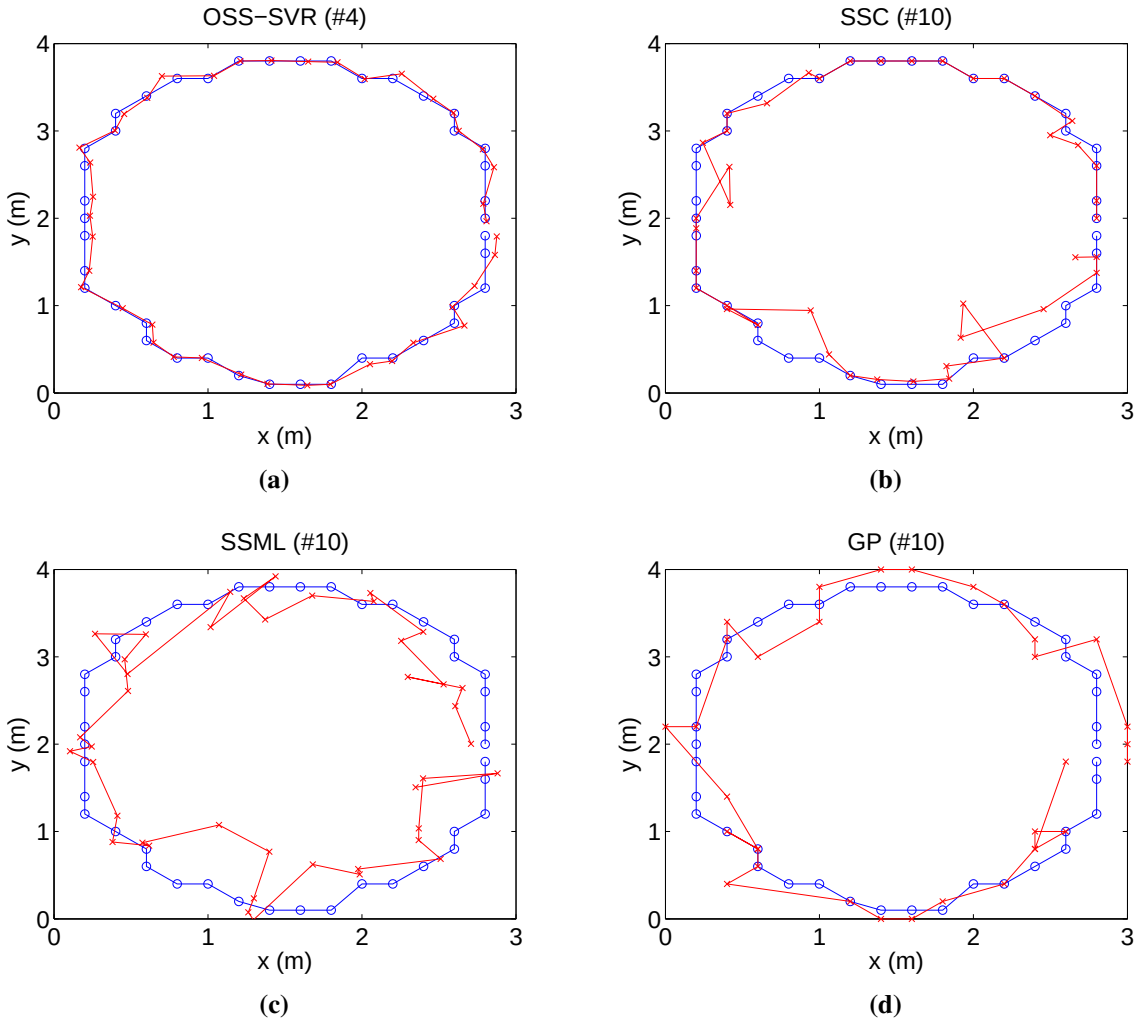


Figure 3. Localization results of a circular trajectory for four different methods. Target circles ground repeatedly, whose true trajectory is marked by the blue-circled line, and the estimated path is shown in red line. All of the figures show online tracking results where the title number of each figure indicates the number of laps the target has made.

Finally, when the environment is changed, we examine the robustness or fast recovery to an accurate solution of each online learning. Figure 4b shows how fast and accurate they learn the changed model under the same experimental setup used in Figure 3. At this time, localization error is defined by RMSE $= \sqrt{1/20 \sum_{t=i-19}^{i} \|\hat{y}_t - y_t\|_2^2}$, with current time step i. We intentionally force constant bias noise, whose magnitude is 10% of the maximum RSSI value, into the measurements of one sensor node. The bias noise is added (*i.e.*, at 10 seconds in Figure 4b) just after the learning models of each algorithm have converged. Our method quickly builds a corrected learning model leading to the best estimation, while SSC shows large error to the bias noise and slow recovery. We note that SSML and GP are not much affected by bias noise, because the online GP of [19] uses only current sensor measurement and SSML trains all of the old and current data per every time step. Therefore, the bias noise does not have meaning for the two methods. However in the long run, they cannot approach accurate estimation in comparison with SSC and OSS-SVR.

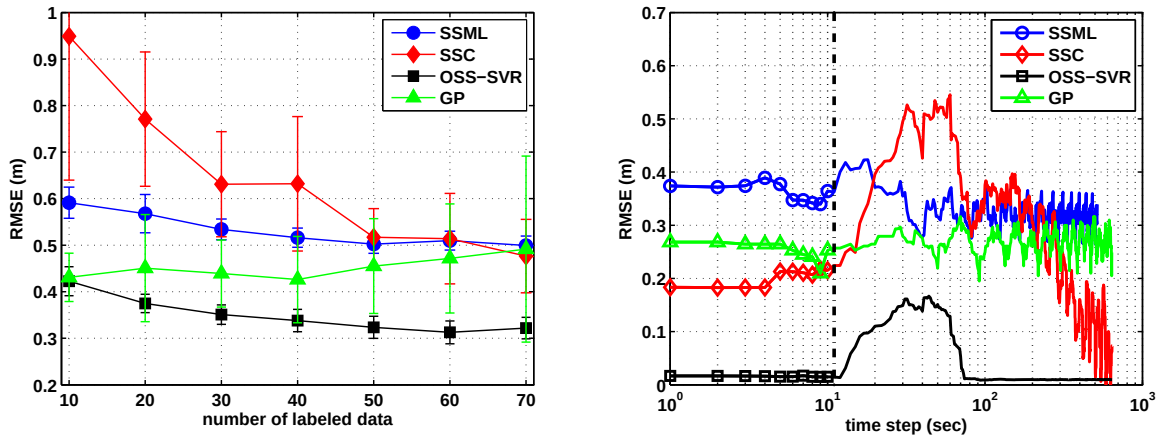


Figure 4. Localization performance of circular trajectory according to (**a**) the number of labeled data and (**b**) the change of the model interrupted by bias noise.

6.4. Localization Results for a Sinusoidal Trajectory

In the experimental setting of the prior section, the target moved in a circular trajectory in order to show visual localization results. In this section, a more complex target path is made, as in Figure 5a, to search the whole area. In the case of the online learning, many training data points stack up. Therefore, we limit the amount of training data using a decremental algorithm. For OSS-SVR, a data point having the lowest support value is removed. For the other methods, the oldest data point is removed sequentially. Those data points are removed when the RMSE is smaller than a pre-defined threshold value. We set 0.3 for OSS-SVR and SSC and 1.0 for SSML as the threshold values, because each method has different values for the final error. Figure 5 shows the online tracking results when no initial labeled data and 30 initial unlabeled data points are used. As shown in Figure 5b, OSS-SVR yields the best accuracy with the fastest convergence. During the early time steps, OSS-SVR may take a slightly longer time than the others in Figure 5c. As time passes, however, the computation time remains bounded to being small, because the well-learned model does not spend much time for learning the new training data.

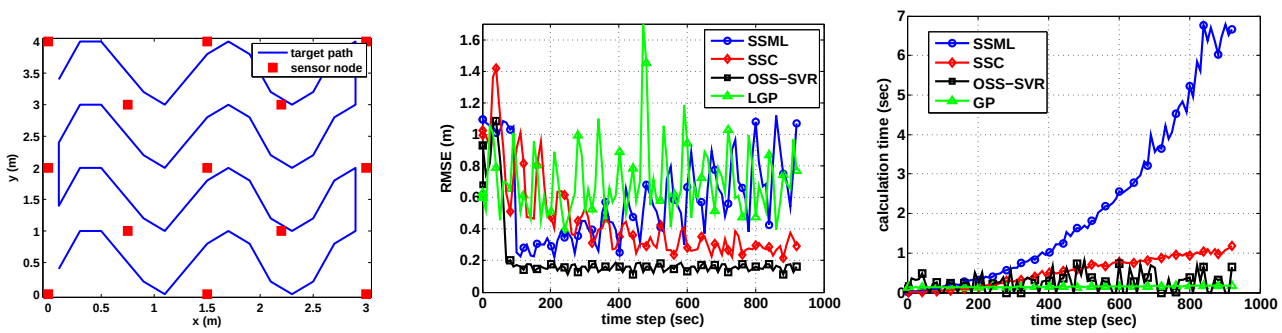


Figure 5. Localization performance of a sinusoidal trajectory, when we apply a decremental algorithm to each method in order to limit the amount of training data. (**a**) Target path; (**b**) localization error; and (**c**) calculation time.

6.5. Localization Results of Kalman Filter-Based Fingerprinting

Fingerprinting methods can be further improved for tracking a moving target. This section shows an extended localization algorithm where the fingerprinting methods are combined with the Kalman filter. The target state takes the form $T_t = [y_{\mathrm{X}t}, y_{\mathrm{Y}t}, v_{\mathrm{X}t}, v_{\mathrm{Y}t}]^T$ where $y_{\mathrm{X}t}, y_{\mathrm{Y}t}$ are 2D positions and $v_{\mathrm{X}t}, v_{\mathrm{Y}t}$ are 2D velocities. The state dynamics are given by:

$$T_t = FT_{t-1} + \Gamma_\sigma \tag{39}$$

where four-by-four matrix F is given by $[1\ 0\ \Delta t\ 0; 0\ 1\ 0\ \Delta t; 0\ 0\ 1\ 0; 0\ 0\ 0\ 1]$ and Γ_σ is a Gaussian noise, whose mean is zero and variance is Σ_σ. The variance matrix Σ_σ is a diagonal matrix whose diagonal elements are σ_{X}^2, σ_{Y}^2, $\sigma_{v_{\mathrm{X}}}^2$ and $\sigma_{v_{\mathrm{Y}}}^2$, respectively.

We use the estimated position obtained from the fingerprinting methods as the observation of the Kalman filter, *i.e.*, $Z_t = [\hat{y}_{\mathrm{X}t}, \hat{y}_{\mathrm{Y}t}]^T$, where $\hat{y}_{\mathrm{X}t}$, $\hat{y}_{\mathrm{Y}t}$ are the estimated position from the fingerprinting learning method, such as OSS-SVR, SSC, SSML and GP. Therefore, the measurements are given by:

$$Z_t = HT_t + \Gamma_\epsilon \tag{40}$$

where two-by-four matrix H is given by $[1\ 0\ 0\ 0; 0\ 1\ 0\ 0]$ and Γ_ϵ is a Gaussian noise, whose mean is zero and variance is Σ_ϵ. The variance matrix Σ_ϵ is a diagonal matrix whose diagonal elements are ϵ^2. Then, iterative updates of dynamic and observation model are implemented.

We compare the results of the Kalman filter-based localization of SSML, GP, SSC and our algorithm. The experimental trajectory is same as Figure 5a. In this scenario, we use 100 initial labeled training data points and 50 initial unlabeled data points. As shown in Figure 6, all of the Kalman filter-based localization results provide better accuracy than the basic fingerprinting methods in Figure 5b. Furthermore, in Figure 6, the suggested algorithm gives the greatest accuracy when the Kalman filter is combined.

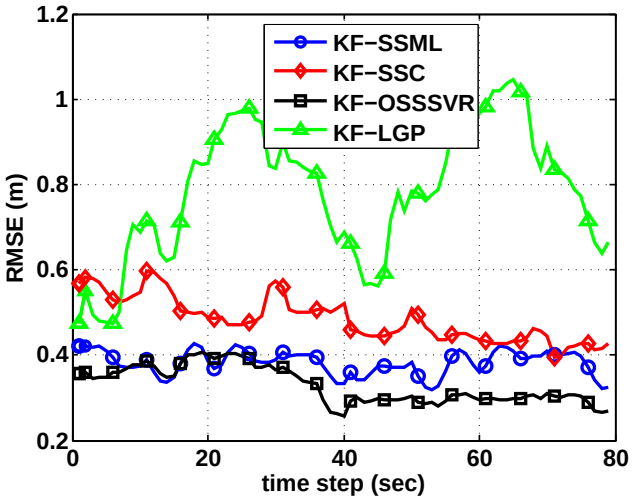


Figure 6. Comparison of Kalman filter-based localization of SSML, GP and our algorithm.

7. Conclusions

This paper proposes an online semi-supervised regression algorithm by combining the core concepts of the manifold regularization framework and the supervised online SVR. By taking advantage of both semi-supervised learning and online learning, our algorithm achieves good accuracy using only a small number of labeled training data and automatically tracks the change of the system to be learned. Furthermore, support vectors are used to decide the importance of a data point in a straightforward manner, allowing minimal memory usage. In comparison with the three state-of-the-art learning methods for target localization using WSN, the proposed algorithm yields the most precise performance of online estimation and rapid recovery to accurate estimation after bias noise is added. Moreover, computation of the suggested algorithm is fast, while maintaining the best accuracy in comparison with the other methods. Furthermore, we formulate a Kalman filter-based fingerprinting localization in order to track a moving target more smoothly and accurately.

Acknowledgments

This work was conducted at High-Speed Vehicle Research Center of KAIST with the support of Defense Acquisition Program Administration (DAPA) and Agency for Defense Development (ADD).

Author Contributions

Jaehyun Yoo and Hyoun Jim Kim designed the learning algorithm; Jaehyun Yoo performed the experiments; Jaehyun Yoo and Hyoun Jim Kim wrote the paper.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Sugano, M.; Kawazoe, T.; Ohta, Y.; Murata, M. Indoor localization system using RSSI measurement of wireless sensor network based on ZigBee standard. In Proceedings of the 2006 IASTED International Conference on Wireless Sensor Networks (WSN), Banff, AL, Canada, 3–4 July 2006; pp. 1–6.
2. Yoo, J.; Kim, H.J. Target Tracking and Classification from Labeled and Unlabeled Data in Wireless Sensor Networks. *Sensors* **2014**, *14*, 23871–23884.
3. Ahmad, U.; Gavrilov, A.; Nasir, U.; Iqbal, M.; Cho, S.J.; Lee, S. In-building localization using neural networks. In Proceedings of the 2006 IEEE International Conference on Engineering of Intelligent Systems, Islamabad, Pakistan, 22–23 April 2006; pp. 1–6.
4. Reddy, P.P.; Veloso, M.M. RSSI-based physical layout classification and target tethering in mobile ad-hoc networks. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), San Francisco, CA, USA, 25–30 September 2011; pp. 2327–2332.
5. Wang, Y.; Martonosi, M.; Peh, L.S. Predicting link quality using supervised learning in wireless sensor networks. *ACM SIGMOBILE Mob. Comput. Commun. Rev.* **2007**, *11*, 71–83.

6. Belkin, M.; Niyogi, P. Semi-supervised learning on Riemannian manifolds. *Mach. Learn.* **2004**, *56*, 209–239.
7. Käll, L.; Canterbury, J.D.; Weston, J.; Noble, W.S.; MacCoss, M.J. Semi-supervised learning for peptide identification from shotgun proteomics datasets. *Nat. Methods* **2007**, *4*, 923–925.
8. Guillaumin, M.; Verbeek, J.; Schmid, C. Multimodal semi-supervised learning for image classification. In Proceedings of the 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), San Francisco, CA, USA, 13–18 June 2010; pp. 902–909.
9. Zhang, Z.; Schuller, B. Semi-supervised learning helps in sound event classification. In Proceedings of the 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Kyoto, Japan, 25–30 March 2012; pp. 333–336.
10. Adankon, M.M.; Cheriet, M.; Biem, A. Semisupervised least squares support vector machine. *IEEE Trans. Neural Netw.* **2009**, *20*, 1858–1870.
11. Kumar Mallapragada, P.; Jin, R.; Jain, A.K.; Liu, Y. Semiboost: Boosting for semi-supervised learning. *IEEE Trans. Pattern Anal. Mach. Intell.* **2009**, *31*, 2000–2014.
12. Lee, D.; Lee, J. Equilibrium-based support vector machine for semisupervised classification. *IEEE Trans. Neural Netw.* **2007**, *18*, 578–583.
13. Zhu, X.; Ghahramani, Z.; Lafferty, J. Semi-supervised learning using gaussian fields and harmonic functions. *ICML* **2003**, *3*, 912–919.
14. Pan, J.J.; Pan, S.J.; Yin, J.; Ni, L.M.; Yang, Q. Tracking mobile users in wireless networks via semi-supervised colocalization. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 587–600.
15. Yang, B.; Xu, J.; Yang, J.; Li, M. Localization algorithm in wireless sensor networks based on semi-supervised manifold learning and its application. *Clust. Comput.* **2010**, *13*, 435–446.
16. Chen, L.; Tsang, I.W.; Xu, D. Laplacian embedded regression for scalable manifold regularization. *IEEE Trans. Neural Netw. Learn. Syst.* **2012**, *23*, 902–915.
17. Tran, D.A.; Nguyen, T. Localization in wireless sensor networks based on support vector machines. *IEEE Trans. Parallel Distrib. Syst.* **2008**, *19*, 981–994.
18. Park, S.; Mustafa, S.K.; Shimada, K. Learning-based robot control with localized sparse online Gaussian process. In Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Tokyo, Japan, 3–7 November 2013; pp. 1202–1207.
19. Yoo, J.H.; Kim, W.; Kim, H.J. Event-driven Gaussian process for object localization in wireless sensor networks. In Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), San Francisco, CA, USA, 25–30 September 2011; pp. 2790–2795.
20. Suykens, J.A.; De Brabanter, J.; Lukas, L.; Vandewalle, J. Weighted least squares support vector machines: robustness and sparse approximation. *Neurocomputing* **2002**, *48*, 85–105.
21. Bishop, C.M.; Nasrabadi, N.M. *Pattern Recognition and Machine Learning*; Springer: New York, NY, USA, 2006; Volume 1.
22. Bekkali, A.; Sanson, H.; Matsumoto, M. RFID indoor positioning based on probabilistic RFID map and Kalman filtering. In Proceedings of the Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, White Plains, NY, USA, 8–10 October 2007; pp. 21–21.

23. Pizarro, D.; Mazo, M.; Santiso, E.; Marron, M.; Jimenez, D.; Cobreces, S.; Losada, C. Localization of mobile robots using odometry and an external vision sensor. *Sensors* **2010**, *10*, 3655–3680.
24. Schölkopf, B.; Herbrich, R.; Smola, A.J. A generalized representer theorem. In *Computational Learning Theory*; Springer: Amsterdam, Netherlands, 2001; pp. 416–426.
25. Ma, J.; Theiler, J.; Perkins, S. Accurate on-line support vector regression. *Neural Comput.* **2003**, *15*, 2683–2703.
26. Martin, M. On-line support vector machine regression. In *Machine Learning: ECML 2002*; Springer: Helsinki, Finland, 2002; pp. 282–294.
27. Wang, H.; Pi, D.; Sun, Y. Online SVM regression algorithm-based adaptive inverse control. *Neurocomputing* **2007**, *70*, 952–959.
28. Parrella, F. Online Support Vector Regression. Master's Thesis, Department of Information Science, University of Genoa, Genoa, Italy, 2007.
29. Laskov, P.; Gehl, C.; Krüger, S.; Müller, K.R. Incremental support vector learning: Analysis, implementation and applications. *J. Mach. Learn. Res.* **2006**, *7*, 1909–1936.
30. Belkin, M.; Niyogi, P.; Sindhwani, V. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *J. Mach. Learn. Res.* **2006**, *7*, 2399–2434.
31. Golub, G.H.; Heath, M.; Wahba, G. Generalized cross-validation as a method for choosing a good ridge parameter. *Technometrics* **1979**, *21*, 215–223.