Novel Range-Free Localization Based on Multidimensional Support Vector Regression Trained in the Primal Space

Jaehun Lee, Baehoon Choi, and Euntai Kim

***Abstract*— A novel range-free localization algorithm based on the multidimensional support vector regression (MSVR) is proposed in this paper. The range-free localization problem is formulated as a multidimensional regression problem, and a new MSVR training method is proposed to solve the regression problem. Unlike standard support vector regression, the proposed MSVR allows multiple outputs and localizes the sensors without resorting to multilateration. The training of the MSVR is formulated directly in primal space and it can be solved in two ways. First, it is formulated as a second-order cone programming and trained by convex optimization. Second, its own training method is developed based on the Newton–Raphson method. A simulation is conducted for both isotropic and anisotropic networks, and the proposed method exhibits excellent and robust performance in both isotropic and anisotropic networks.**

***Index Terms*— Convex optimization, range-free localization, support vector regression (SVR), wireless sensor networks (WSNs).**

I. Introduction

TYPICALLY, wireless sensor networks (WSNs) consist of a large number of sensor nodes that are randomly distributed in a given field. In general, most of the sensor nodes are tiny sensors with small memories and are unaware of their own locations. The sensor node location information, however, is of great importance in many WSN applications such as target tracking [1], environmental monitoring [2], and search and rescue [3], among other things. In many applications, hundreds or thousands of sensor nodes are used, but only the locations of a small number of sensor nodes are known *a priori* by either a global positioning system (GPS) [4] or by manual input. The sensors for the locations that are known *a priori* are called anchor nodes.

Recently, much research has been conducted to develop solutions to the localization problem in WSNs [5]–[11], [21]–[31], including range-based algorithms and range-free algorithms. Range-based algorithms estimate the location of unknown sensor nodes based on measurements such as the distance or angle between sensor nodes. These measurements can be estimated by the received signal strength (RSS), time of arrival, or angle of arrival [5]. Range-based algorithms might provide relatively accurate localization results, but they require some additional hardware for distance or angle measurements.

In contrast, the range-free algorithms use only the connectivity information among the sensor nodes [6]. Range-free algorithms are less expensive and simpler than the range-based methods since they do not require any additional hardware. Thus, the range-free method might be more appropriate than the range-based method in large-scale WSNs, which consist of hundreds or even thousands of sensor nodes.

As a pioneering work of the range-free approach, Niculescu and Nath [7] developed the DV-hop approach for the localization problem. The DV-hop method works well in an isotropic network since the sensor and anchor nodes are placed uniformly over the entire area. However, this method results in large errors in anisotropic networks since the nodes are not uniformly distributed and the relationship between the hop counts and geographic distances is very weak. Bulusu *et al.* [8] proposed a centroid localization algorithm. In the centroid method, the anchor nodes broadcast their positions and each sensor node computes its position as a center of the connected anchor nodes. The centroid algorithm is simple and economical, but requires many anchor nodes since all the sensor nodes must be connected to the anchor node to achieve good localization results. Furthermore, He *et al.* [9] proposed the APIT localization algorithm. In the APIT method, location estimation is conducted by dividing the environment into triangular regions between the anchor nodes and narrowing down the likely area based on each sensor node's presence inside or outside of those triangles. The limitation of the APIT localization algorithm is that the sensor nodes should be connected to several anchor nodes. Also, Ma *et al.* [10] proposed the HCQ approach. In this approach, the hop count measure is represented by real number to improve the localization performance. Interestingly, Lim and Hou [11] developed a proximity-distance map (PDM) algorithm to tackle the anisotropic network. In their work, the authors considered localization as an embedding problem that mapped the geographic distance into the proximity measurement and solved this problem by defining the linear mapping matrix. The localization was formulated as a regression problem and recast into the least squares problem. The solution of the problem is a PDM that is sent to all sensor nodes. Each sensor node estimates its distance from all of the

Manuscript received February 16, 2012; revised January 16, 2013; accepted February 10, 2013. Date of publication March 27, 2013; date of current version May 14, 2013. This work was supported by the Basic Science Research Program through the National Research Foundation of Korea Funded by the Ministry of Education, Science and Technology under Grant 2012-0003006.

The authors are with the School of Electrical and Electronic Engineering, Yonsei University, Seoul 120-749, Korea (e-mail: aznable17@yonsei.ac.kr; choibae@yonsei.ac.kr; etkim@yonsei.ac.kr).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2013.2250996

anchor nodes based on the PDM, and finally localizes using the multilateration technique described in [7].

On the other hand, machine learning (ML) has received much attention within the artificial intelligence community and has been applied to a variety fields. The two most representative algorithms in ML are support vector machine/regression (SVM/SVR) [12]–[15] and extreme learning machine (ELM) [16]–[20]. SVM and SVR are based on the idea that not all the data points are important but a limited number of data points called support vectors are critical. Therefore, only support vectors are required to send to sensor nodes, so it reduces the communications overhead. Instead, ELM is based on the idea that it can use a number of randomly generated hidden nodes. The ML methods have been applied to the WSN localization problem. For example, [21]–[26] are the range-based approaches, while [27]–[31] are the range-free approaches.

In this paper, a new range-free localization algorithm is proposed based on the ML technique. The proposed method employs multidimensional SVR. Motivated by [27], the proposed method views the localization problem as a regression problem and solves it using SVR, which is one of the most popular machine learning methods. The multidimensional support vector regression (MSVR) is not new, and it was reported in [32]. In this paper, however, a new training method is proposed, and it is applied to a range-free localization problem. The training of the MSVR is formulated directly as an second-order cone programming (SOCP) in the primal space, and it is solved by either the interior point method [33] or the Newton–Raphson method [34]. The formulation in the primal space is motivated by [35], but the subsequent development is completely new.

The rest of this paper is organized as follows: In Section II, the localization problem is formulated into a regression problem. In Section III, the proposed method based on the MSVR model is presented. Simulations are described in Section IV, and the results of the proposed method are compared with those of previous methods. Finally, conclusions are given in the last section.

II. Problem Formulation

Let us consider a sensor network $SN = \{s_1, s_2, \ldots, s_{M+N}\}$ with M anchor nodes and N sensor nodes where $N \gg M$, and let us denote the location of each node by

$$\mathbf{l}_i = (x_i, y_i)^T \quad \text{for } i = 1, \ldots, M+N. \tag{1}$$

For the sake of simplicity, suppose that the space of interest is 2-D, but the proposed method can be extended to a 3-D space in a straightforward way. It is assumed that the locations of M anchor nodes $s_i \in \mathrm{A}$ are known, but the locations of the other N sensor nodes $s_j \in \Sigma$ are unknown, where $\mathrm{A} = \{s_1, \ldots, s_M\}$ and $\Sigma = \{s_{M+1}, \ldots, s_{M+N}\}$ denote the sets of anchor and sensor nodes, respectively. The only available measurement is the proximity or connectivity information represented by

$$p_{i,j} \in \mathbf{Z} = \{0, 1, 2, 3, \ldots\} \tag{2}$$

where $p_{i,j}$ denotes the number of hops between the ith node s_i and the jth node s_j. For example, if s_i and s_j are directly connected, then the proximity value $p_{i,j}$ is set to be 1. Note that $p_{i,i} = 0$ for all nodes $i = 1, 2, \ldots, M+N$. The localization problem can then be defined as follows:

$$\begin{aligned}&\text{Estimate } \mathbf{l}_j\\ &\text{given } \mathbf{l}_i \text{ and } p_{k,l}\end{aligned} \tag{3}$$

where $i \in \{1, \ldots, M\}$, $j \in \{M+1, \ldots, M+N\}$, and $k, l \in \{1, \ldots, M+N\}$.

In [33], the above range-free localization problem was formulated as a linear regression problem or linear optimization problem using quadratic programming. Here, the above problem is formulated as a multidimensional nonlinear regression problem. This formulation is based on the fact that the sensor nodes that are close to each other will have similar connections to the anchor nodes. Thus, when a sensor node is given, its connection pattern to the anchor nodes gives us good information about its location. That is, when the proximity information $\mathbf{p}_i$ between nodes is given, and the locations of the anchor nodes $s_i \in \mathrm{A}$ are known, the goal is to build a regression function

$$\mathbf{l}_j = \mathbf{f}\left(\mathbf{p}_j\right) = \mathbf{W}\boldsymbol{\varphi}\left(\mathbf{p}_j\right) + \mathbf{b} \in \mathbf{R}^2 \tag{4}$$

such that the locations $\mathbf{l}_j$ of the unknown sensor nodes $s_j \in \Sigma$ are estimated as accurately as possible, where W is the weight matrix, $\boldsymbol{\varphi}(\cdot)$ is a nonlinear function acting on the proximity vector, $\mathbf{b}$ is the bias vector, and $\mathbf{p}_j$ is the proximity vector between the jth node and all anchor nodes

$$\mathbf{p}_j = \left[\, p_{j,1} \cdots p_{j,M} \,\right]^T \in \mathbf{Z}^M. \tag{5}$$

The nonlinear function $\varphi(\cdot)$ is called a feature vector [37], and will be specified later. Here, it is worth noting that the regression model (4) used in the WSN localization has two outputs (x, y) and therefore the standard SVR cannot be applied. In the next section, a new multidimensional support vector regression (MSVR) is proposed and a new training method is developed.

III. Localization Based on MSVR Optimization

In this section, a novel range-free localization algorithm based on the MSVR model is proposed. The whole localization procedure of the proposed method is as follows.

1) Each anchor node $s_i \in \mathrm{A}$ broadcasts a message to all the other nodes, including both the sensor and anchor nodes. Each anchor node s_i then transmits its location information to the sink node s_k, which is one of the anchor nodes, i.e., $s_k \in \mathrm{A}$ (Measurement step).
2) The sink node s_k estimates the optimal MSVR model and broadcasts the resulting parameters of the MSVR model to all sensor nodes (MSVR step). The details about the parameters are given in Section III-B.
3) Each sensor node $s_j \in \Sigma$ estimates its location using the MSVR model and the parameters given by the sink node s_k (Localization step).

In the above algorithm, any anchor node can be selected as the sink node. To facilitate the subsequent training of MSVR, however, the anchor node with highest computational power is selected as the sink node. The communication costs for the entire process are $O(MN)$ messages. The following subsections describe the details of the proposed method.

A. Measurement Step

In this step, all sensor and anchor nodes communicate with each other when collecting the connectivity measurements. The actual communication procedure is similar to that in [27]. First, each anchor node $s_i \in \mathrm{A}$ broadcasts a HELLO message to all of the other nodes, including both the sensor and anchor nodes, so that each anchor node $s_i \in \mathrm{A}$ and sensor node $s_j \in \Sigma$ know their corresponding proximity vectors $\mathbf{p}_i$ and $\mathbf{p}_j$, respectively, which denote the number of hop counts between itself and all anchor nodes. Each anchor node s_i then sends an INFO message consisting of its proximity vector $\mathbf{p}_i$ and location information $\mathbf{l}_i$ to a sink node s_k. The total communication cost of the measurement step is composed of $M(M+N-1)$ HELLO messages and $M-1$ INFO messages. Since the number of anchor nodes is generally much smaller than the number of sensor nodes ($M \ll N$), the total communication costs are $O(MN)$.

B. Multidimensional Support Vector Regression Model

Here, the MSVR model is proposed and applied to the given localization problem. Our goal is to build a regression model such that it outputs the estimate of location $\mathbf{l}_i$ of the sensor node $s_i \in \Sigma$ based on the proximity information $\mathbf{p}_i$. After the measurement step, M training data pairs $(\mathbf{p}_i, \mathbf{l}_i) \in \mathbf{Z}^M \times \mathbf{R}^2$ $(i = 1, 2, \ldots, M)$ are given to a sink node s_k. When this training dataset is presented, a subsequent multidimensional regression function

$$f(\mathbf{p}_i) = \mathbf{W}\boldsymbol{\varphi}(\mathbf{p}_i) + \mathbf{b} = \begin{bmatrix} \mathbf{w}_x^T \\ \mathbf{w}_y^T \end{bmatrix} \boldsymbol{\varphi}(\mathbf{p}_i) + \begin{bmatrix} b_x \\ b_y \end{bmatrix} \tag{6}$$

is trained, where $\mathbf{W} = \begin{bmatrix} \mathbf{w}_x & \mathbf{w}_y \end{bmatrix}^T$ is the weight matrix and $b = \begin{bmatrix} b_x & b_y \end{bmatrix}^T$ is the bias vector for 2-D regression. The training of function (6) is formulated as the following optimization problem

$$\begin{aligned} &\text{minimize } \left(\|\mathbf{w}_x\|^2 + \|\mathbf{w}_y\|^2\right) + C\sum_{i=1}^{M} \xi_i^2 \\ &\text{s.t. } \|\mathbf{l}_i - \mathbf{W}\boldsymbol{\varphi}(\mathbf{p}_i) - \mathbf{b}\| \le \varepsilon + \xi_i \\ &\quad \xi_i \ge 0 \end{aligned} \tag{7}$$

where ξ_i denotes the slack variables, and C denotes the soft margin parameter. Equation (7) is the generalization of the SVR [37], with an L_2 penalty from increasing the number of outputs. To train the MSVR, the above optimization problem should be formulated in the dual space, but this is not directly possible here since the first inequality constraint is not affine. To solve this problem, a new training method for the MSVR is proposed in this paper. The new training is formulated directly in the primal space without resorting to the dual formulation. This idea is motivated by [35]. By defining the ε-insensitive quadratic loss function $L_\varepsilon : \mathbf{R}^+ \to \mathbf{R}^+$ as

$$L_\varepsilon(z) = \begin{cases} 0, & \text{if } z \le \varepsilon \\ (z-\varepsilon)^2, & \text{otherwise} \end{cases} \tag{8}$$

where $\mathbf{R}^+$ denotes a nonnegative real set, the training of the MSVR can be equivalently rewritten as an unconstrained optimization problem

$$\text{minimize } \left(\|\mathbf{w}_x\|^2 + \|\mathbf{w}_y\|^2\right) + C\sum_{i=1}^{M} L_\varepsilon\left(\|\mathbf{l}_i - \mathbf{W}\boldsymbol{\varphi}(\mathbf{p}_i) - \mathbf{b}\|\right). \tag{9}$$

It is worth noting here that the two optimization problems (7) and (9) are equivalent. Since the objective function (9) is convex in W and b, the optimal solution is achieved by zeroing its derivative

$$\begin{aligned} 2\mathbf{w}_x^* + C\sum_{i=1}^{M} \frac{\partial}{\partial \mathbf{w}_x} L_\varepsilon\left(\|\mathbf{l}_i - \mathbf{W}\boldsymbol{\varphi}(\mathbf{p}_i) - \mathbf{b}\|\right)\boldsymbol{\varphi}(\mathbf{p}_i) = 0 \\ 2\mathbf{w}_y^* + C\sum_{i=1}^{M} \frac{\partial}{\partial \mathbf{w}_y} L_\varepsilon\left(\|\mathbf{l}_i - \mathbf{W}\boldsymbol{\varphi}(\mathbf{p}_i) - \mathbf{b}\|\right)\boldsymbol{\varphi}(\mathbf{p}_i) = 0 \end{aligned} \tag{10}$$

where $\mathbf{w}_x^*$ and $\mathbf{w}_y^*$ represent the optimal solution of $\mathbf{w}_x$ and $\mathbf{w}_y$, respectively.

Defining

$$\begin{aligned} \beta_{i,x} &= -\frac{C}{2}\frac{\partial}{\partial \mathbf{w}_x} L_\varepsilon\left(\|\mathbf{l}_i - \mathbf{W}\boldsymbol{\varphi}(\mathbf{p}_i) - \mathbf{b}\|\right) \\ \beta_{i,y} &= -\frac{C}{2}\frac{\partial}{\partial \mathbf{w}_y} L_\varepsilon\left(\|\mathbf{l}_i - \mathbf{W}\boldsymbol{\varphi}(\mathbf{p}_i) - \mathbf{b}\|\right) \end{aligned} \tag{11}$$

in (10) yields

$$\begin{aligned} \mathbf{w}_x &= \sum_{i=1}^{M} \boldsymbol{\varphi}(\mathbf{p}_i)\beta_{i,x} \\ \mathbf{w}_y &= \sum_{i=1}^{M} \boldsymbol{\varphi}(\mathbf{p}_i)\beta_{i,y}. \end{aligned} \tag{12}$$

It is worth noting here that the optimal solution of the above optimization problem (9) is represented as a linear combination of feature functions evaluated at the training samples. This result is also known as the representer theorem in machine learning [35], [38]. Substituting (12) into (7) and (9) yields the following two equivalent optimization problems:

Constrained optimization problem

$$\begin{aligned} \underset{\beta,\xi}{\text{minimize}}\ \lambda &\left(\sum_{i,j=1}^{M} \beta_{i,x}\beta_{j,x} k(\mathbf{p}_i, \mathbf{p}_j) + \sum_{i,j=1}^{M} \beta_{i,y}\beta_{j,y} k(\mathbf{p}_i, \mathbf{p}_j)\right) + \sum_{i=1}^{M} \xi_i^2 \\ \text{s.t. } &\left\|\mathbf{l}_i - \sum_{j=1}^{M} \beta_j k(\mathbf{p}_j, \mathbf{p}_i) - \mathbf{b}\right\| \le \varepsilon + \xi_i \\ &\xi_i \ge 0. \end{aligned} \tag{13}$$

Unconstrained optimization problem

$$\text{minimize}\ \lambda\left(\sum_{i,j=1}^{M}\beta_{i,x}\beta_{j,x}k\left(\mathbf{p}_i,\mathbf{p}_j\right)+\sum_{i,j=1}^{M}\beta_{i,y}\beta_{j,y}k\left(\mathbf{p}_i,\mathbf{p}_j\right)\right)+\sum_{i=1}^{M}L_\varepsilon\left(\left\|\mathbf{l}_i-\sum_{j=1}^{M}k\left(\mathbf{p}_i,\mathbf{p}_j\right)\beta_j-b\right\|\right)\quad(14)$$

where $\lambda = 1/C$ is the regularization parameter, $k(\mathbf{p}_i,\mathbf{p}_j) = \boldsymbol{\varphi}^T(\mathbf{p}_i)\boldsymbol{\varphi}(\mathbf{p}_j)$ denotes the kernel function, and $\boldsymbol{\beta}_j = [\,\beta_{j,x}\ \beta_{j,y}\,]^T$. The training can be solved in either the constrained or unconstrained formulations. In the case of the constrained formulation, the problem (13) belongs to SOCP [39] and can be solved by the interior method [33] or by using off-the-shelf commercial programs such as CVX [40]. Algorithm 1 shows the cvx code of the problem (13) in MALTAB implementation.

Algorithm 1 MATLAB CVX Code for (13)

```
cvx_begin
    variable beta(M,2), Xi(M),b(dim);
    K = kernelTransform(TrainX,TrainX);
    K_ext = zeros(M,M);
    expression beta_vec(M*2);
    for i = 1 : 2
       beta_vec(M*(i-1)+1:M*i)=beta(:,i);
       K_ext(M*(i-1)+1:M*i,M*(i-1)+1:M*i)=K;
    end
    expression norm_const(M);
    for i = 1 : M
       norm_const(i)=norm(l_i - beta'*K(:,i)-b,2);
    end
    minimize
       (ramda*beta_vec'*K_ext*beta_vec + Xi'*Xi)
    subject to
       Xi >= 0;
       norm_const-Xi <= epsilon
cvx_end
```

In the case of the unconstrained formulation, (14) can be solved by iterating the Newton–Raphson method [34]. The objective function in (14) can be rewritten in matrix and vector form as

$$\lambda\left(\boldsymbol{\beta}_x^T\mathbf{K}\boldsymbol{\beta}_x+\boldsymbol{\beta}_y^T\mathbf{K}\boldsymbol{\beta}_y\right)+\sum_{i=1}^{M}L_\varepsilon\left(\left\|\mathbf{l}_i-\boldsymbol{\beta}^T\mathbf{k}_i-\mathbf{b}\right\|\right)\quad(15)$$

where $\boldsymbol{\beta}_x = [\,\beta_{1,x}\cdots\beta_{M,x}\,]^T$, $\boldsymbol{\beta}_y = [\,\beta_{1,y}\cdots\beta_{M,y}\,]^T$, $\boldsymbol{\beta} = [\,\boldsymbol{\beta}_x\ \boldsymbol{\beta}_y\,]$, and $\mathbf{K}$ is the kernel matrix with $K_{i,j} = k(\mathbf{p}_i,\mathbf{p}_j)$ and $\mathbf{k}_i = [\,k(\mathbf{p}_i,\mathbf{p}_1)\cdots k(\mathbf{p}_i,\mathbf{p}_M)\,]^T$. By augmenting $\boldsymbol{\beta}_x$, $\boldsymbol{\beta}_y$, and $\mathbf{b}$ as a single vector $\boldsymbol{\beta}_{\text{aug}}$, (15) can be rewritten as

$$\lambda\boldsymbol{\beta}_{\text{aug}}^T\mathbf{K}_{\text{aug}}\boldsymbol{\beta}_{\text{aug}}+\sum_{i=1}^{M}L_\varepsilon\left(\left\|\mathbf{l}_i-\hat{\mathbf{K}}_i^T\boldsymbol{\beta}_{\text{aug}}\right\|\right)\quad(16)$$

where

$$\boldsymbol{\beta}_{\text{aug}}=\begin{bmatrix}\boldsymbol{\beta}_x\\ \boldsymbol{\beta}_y\\ \mathbf{b}\end{bmatrix}_{(2M+2)\times 1},\quad \mathbf{K}_{\text{aug}}=\begin{bmatrix}\mathbf{K}&\mathbf{0}&\mathbf{0}\\ \mathbf{0}&\mathbf{K}&\mathbf{0}\\ \mathbf{0}&\mathbf{0}&\mathbf{0}\end{bmatrix}_{(2M+2)\times(2M+2)},$$

$$\hat{\mathbf{K}}_i=\begin{bmatrix}\mathbf{k}_i&\mathbf{0}\\ \mathbf{0}&\mathbf{k}_i\\ 1&0\\ 0&1\end{bmatrix}_{(2M+2)\times 2}.$$

For a given value of vector $\boldsymbol{\beta}_{\text{aug}}$, let the ith data point $\mathbf{p}_i$ be a support vector if the loss defined by (8) is not zero, i.e., $\|\mathbf{l}_i-\hat{\mathbf{K}}_i^T\boldsymbol{\beta}_{\text{aug}}\| > \varepsilon$. Let us reorder the training datasets such that the first M_{sv} points are support vectors. The objective function (16) can then be represented as

$$L\left(\boldsymbol{\beta}_{\text{aug}}\right)=\lambda\boldsymbol{\beta}_{\text{aug}}^T\mathbf{K}_{\text{aug}}\boldsymbol{\beta}_{\text{aug}}+\sum_{i=1}^{M_{sv}}\left(\left\|\mathbf{l}_i-\hat{\mathbf{K}}_i^T\boldsymbol{\beta}_{\text{aug}}\right\|-\varepsilon\right)^2.\quad(17)$$

When applying the Newton–Raphson method to the minimization of (17), the following property is required.

Property: The first and second derivatives of scalar value $\|\mathbf{Ax}+\mathbf{b}\|$ with respect to vector x are represented as follows:

$$\frac{\partial}{\partial\mathbf{x}}\|\mathbf{Ax}+\mathbf{b}\|=\frac{1}{\|\mathbf{Ax}+\mathbf{b}\|}\mathbf{A}^T\left(\mathbf{Ax}+\mathbf{b}\right)\quad(18)$$

$$\frac{\partial^2}{\partial\mathbf{x}^2}\|\mathbf{Ax}+\mathbf{b}\|=\frac{\mathbf{A}^T\mathbf{A}}{\|\mathbf{Ax}+\mathbf{b}\|}-\frac{\mathbf{A}^T\left(\mathbf{Ax}+\mathbf{b}\right)\left(\mathbf{Ax}+\mathbf{b}\right)^T A}{\|\mathbf{Ax}+\mathbf{b}\|^3}.\quad(19)$$

Proof: The details of the proof are given in the Appendix.

Using the above property, the gradient of (17) with respect to $\boldsymbol{\beta}_{\text{aug}}$ is computed by

$$\begin{aligned}\nabla &= \frac{\partial}{\partial\boldsymbol{\beta}_{\text{aug}}}\left(\lambda\boldsymbol{\beta}_{\text{aug}}^T\mathbf{K}_{\text{aug}}\boldsymbol{\beta}_{\text{aug}}+\sum_{i=1}^{M_{sv}}\left(\left\|\mathbf{l}_i-\hat{\mathbf{K}}_i^T\boldsymbol{\beta}_{\text{aug}}\right\|-\varepsilon\right)^2\right)\\ &= \frac{\partial}{\partial\boldsymbol{\beta}_{\text{aug}}}\left(\lambda\boldsymbol{\beta}_{\text{aug}}^T\mathbf{K}_{\text{aug}}\boldsymbol{\beta}_{\text{aug}}\right)\\ &\quad+\frac{\partial}{\partial\boldsymbol{\beta}_{\text{aug}}}\left(\sum_{i=1}^{M_{sv}}\left(\left\|\mathbf{l}_i-\hat{\mathbf{K}}_i^T\boldsymbol{\beta}_{\text{aug}}\right\|-\varepsilon\right)^2\right)\\ &= 2\lambda\mathbf{K}_{\text{aug}}\boldsymbol{\beta}_{\text{aug}}+2\sum_{i=1}^{M_{sv}}\left(\left\|\mathbf{l}_i-\hat{\mathbf{K}}_i^T\boldsymbol{\beta}_{\text{aug}}\right\|-\varepsilon\right)\\ &\quad\times\frac{\partial}{\partial\boldsymbol{\beta}_{\text{aug}}}\left(\left\|\mathbf{l}_i-\hat{\mathbf{K}}_i^T\boldsymbol{\beta}_{\text{aug}}\right\|-\varepsilon\right)\\ &= 2\lambda\mathbf{K}_{\text{aug}}\boldsymbol{\beta}_{\text{aug}}+2\sum_{i=1}^{M_{sv}}\left(\left\|\mathbf{l}_i-\hat{\mathbf{K}}_i^T\boldsymbol{\beta}_{\text{aug}}\right\|-\varepsilon\right)\\ &\quad\times\frac{\hat{\mathbf{K}}_i\left(\hat{\mathbf{K}}_i^T\boldsymbol{\beta}_{\text{aug}}-\mathbf{l}_i\right)}{\left\|\mathbf{l}_i-\hat{\mathbf{K}}_i^T\boldsymbol{\beta}_{\text{aug}}\right\|}\\ &= 2\lambda\mathbf{K}_{\text{aug}}\boldsymbol{\beta}_{\text{aug}}+2\sum_{i=1}^{M_{sv}}\hat{\mathbf{K}}_i\left(\hat{\mathbf{K}}_i^T\boldsymbol{\beta}_{\text{aug}}-\mathbf{l}_i\right)\\ &\quad-2\sum_{i=1}^{M_{sv}}\varepsilon\frac{\hat{\mathbf{K}}_i\left(\hat{\mathbf{K}}_i^T\boldsymbol{\beta}_{\text{aug}}-\mathbf{l}_i\right)}{\left\|\mathbf{l}_i-\hat{\mathbf{K}}_i^T\boldsymbol{\beta}_{\text{aug}}\right\|}\end{aligned}\quad(20)$$

and the Hessian is given by

$$\begin{aligned}\mathbf{H} &= \frac{\partial}{\partial \boldsymbol{\beta}_{\text{aug}}}\left(2\lambda \mathbf{K}_{\text{aug}}\boldsymbol{\beta}_{\text{aug}} + 2\sum_{i=1}^{M_{sv}} \hat{\mathbf{K}}_i \left(\hat{\mathbf{K}}_i^T \boldsymbol{\beta}_{\text{aug}} - \mathbf{l}_i\right)\right. \\ &\qquad \left. -2\sum_{i=1}^{M_{sv}} \varepsilon \frac{\hat{\mathbf{K}}_i \left(\hat{\mathbf{K}}_i^T \boldsymbol{\beta}_{\text{aug}} - \mathbf{l}_i\right)}{\left\|\mathbf{l}_i - \hat{\mathbf{K}}_i^T \boldsymbol{\beta}_{\text{aug}}\right\|}\right) \\ &= \frac{\partial}{\partial \boldsymbol{\beta}_{\text{aug}}}\left(2\lambda \mathbf{K}_{\text{aug}}\boldsymbol{\beta}_{\text{aug}}\right) + \frac{\partial}{\partial \boldsymbol{\beta}_{\text{aug}}}\left(2\sum_{i=1}^{M_{sv}} \hat{\mathbf{K}}_i \left(\hat{\mathbf{K}}_i^T \boldsymbol{\beta}_{\text{aug}} - \mathbf{l}_i\right)\right) \\ &\qquad -\frac{\partial}{\partial \boldsymbol{\beta}_{\text{aug}}}\left(2\sum_{i=1}^{M_{sv}} \varepsilon \frac{\hat{\mathbf{K}}_i \left(\hat{\mathbf{K}}_i^T \boldsymbol{\beta}_{\text{aug}} - \mathbf{l}_i\right)}{\left\|\mathbf{l}_i - \hat{\mathbf{K}}_i^T \boldsymbol{\beta}_{\text{aug}}\right\|}\right) \\ &= 2\lambda \mathbf{K}_{\text{aug}} + 2\sum_{i=1}^{M_{sv}} \hat{\mathbf{K}}_i \hat{\mathbf{K}}_i^T \\ &\qquad -\frac{\partial}{\partial \boldsymbol{\beta}_{\text{aug}}}\left(2\sum_{i=1}^{M_{sv}} \varepsilon \frac{\hat{\mathbf{K}}_i \left(\hat{\mathbf{K}}_i^T \boldsymbol{\beta}_{\text{aug}} - \mathbf{l}_i\right)}{\left\|\mathbf{l}_i - \hat{\mathbf{K}}_i^T \boldsymbol{\beta}_{\text{aug}}\right\|}\right) \\ &= 2\lambda \mathbf{K}_{\text{aug}} + 2\sum_{i=1}^{M_{sv}} \hat{\mathbf{K}}_i \hat{\mathbf{K}}_i^T \\ &\qquad -\frac{\partial}{\partial \boldsymbol{\beta}_{\text{aug}}}\left(2\varepsilon\sum_{i=1}^{M_{sv}} \frac{\left(-\hat{\mathbf{K}}_i\right)\left(\mathbf{l}_i - \hat{\mathbf{K}}_i^T \boldsymbol{\beta}_{\text{aug}}\right)}{\left\|\mathbf{l}_i - \hat{\mathbf{K}}_i^T \boldsymbol{\beta}_{\text{aug}}\right\|}\right) \\ &= 2\lambda \mathbf{K}_{\text{aug}} + 2\sum_{i=1}^{M_{sv}} \hat{\mathbf{K}}_i \hat{\mathbf{K}}_i^T - 2\varepsilon\sum_{i=1}^{M_{sv}} \frac{\hat{\mathbf{K}}_i \hat{\mathbf{K}}_i^T}{\left\|\mathbf{l}_i - \hat{\mathbf{K}}_i^T \boldsymbol{\beta}_{\text{aug}}\right\|} + 2\varepsilon\sum_{i=1}^{M_{sv}} \\ &\qquad \times \frac{\left(-\hat{\mathbf{K}}_i\right)\left(\mathbf{l}_i - \hat{\mathbf{K}}_i^T \boldsymbol{\beta}_{\text{aug}}\right)\left(\mathbf{l}_i - \hat{\mathbf{K}}_i^T \boldsymbol{\beta}_{\text{aug}}\right)^T\left(-\hat{\mathbf{K}}_i^T\right)}{\left\|\mathbf{l}_i - \hat{\mathbf{K}}_i^T \boldsymbol{\beta}_{\text{aug}}\right\|^3} \\ &= 2\lambda \mathbf{K}_{\text{aug}} + 2\sum_{i=1}^{M_{sv}} \hat{\mathbf{K}}_i \hat{\mathbf{K}}_i^T - 2\varepsilon\sum_{i=1}^{M_{sv}} \frac{\hat{\mathbf{K}}_i \hat{\mathbf{K}}_i^T}{\left\|\mathbf{l}_i - \hat{\mathbf{K}}_i^T \boldsymbol{\beta}_{\text{aug}}\right\|} \\ &\qquad +2\varepsilon\sum_{i=1}^{M_{sv}} \frac{\hat{\mathbf{K}}_i \left(\hat{\mathbf{K}}_i^T \boldsymbol{\beta}_{\text{aug}} - \mathbf{l}_i\right)\left(\hat{\mathbf{K}}_i^T \boldsymbol{\beta}_{\text{aug}} - \mathbf{l}_i\right)^T \hat{\mathbf{K}}_i^T}{\left\|\mathbf{l}_i - \hat{\mathbf{K}}_i^T \boldsymbol{\beta}_{\text{aug}}\right\|^3} \end{aligned} \tag{21}$$

where (18) is used in the third and fourth line in (20), and (19) is used in the fourth and fifth lines in (21). Finally, the optimal solution β_{aug} of the MSVR is obtained by iterating the Newton–Raphson update

$$\boldsymbol{\beta}_{\text{aug}}^{\text{new}} = \boldsymbol{\beta}_{\text{aug}}^{\text{old}} - \alpha \mathbf{H}^{-1}\nabla \tag{22}$$

where α is the step size parameter. After the optimal MSVR model is found through the learning process, the sink node s_k broadcasts a message, including the optimal MSVR parameter vector $\bar{\boldsymbol{\beta}}_{\text{aug}} = [\,\bar{\boldsymbol{\beta}}_x^T\ \bar{\boldsymbol{\beta}}_y^T\ \bar{\boldsymbol{b}}^T\,]^T$ and the training points $\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_M\}$, to all sensor nodes. The communication cost for this step is $O(N)$. The overall algorithm is summarized in Algorithm 2.

Algorithm 2 Training of the MSVR Model With the Newton–Raphson Method

```
Procedure Training MSVR
    calculate $\mathbf{K}$ with $\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_M\}$ and $k(\cdot,\cdot)$;
    for i = 1 : Max_iteration
        calculate $\nabla$ with (20);
        calculate $\mathbf{H}$ with (21);
        $\boldsymbol{\beta}_{aug} = \boldsymbol{\beta}_{aug} - \alpha \mathbf{H}^{-1}\nabla$;
    end
    return $\boldsymbol{\beta}_{aug}$
```

TABLE I
PARAMETERS USED IN SIMULATION

Parameter	Value
Kernel functions	Gaussian (ELM: sigmoidal)
σ^2 of kernel function	10
C	30
ϵ	0.2

C. Localization Based on Optimal MSVR

Each sensor node $s_j \in \Sigma$ starts localization after receiving the parameter vector of the optimal MSVR model $\bar{\boldsymbol{\beta}}_{\text{aug}} = [\,\bar{\boldsymbol{\beta}}_x^T\ \bar{\boldsymbol{\beta}}_y^T\ \bar{\boldsymbol{b}}^T\,]^T$ and the training points $\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_M\}$ from the sink node s_k. Each sensor node s_j estimates its own location $\tilde{l}_j = (\tilde{x}_j, \tilde{y}_j)^T$ with (6) and (12)

$$\begin{aligned}(\tilde{x}_j, \tilde{y}_j)^T &= f(\mathbf{p}_j) + \bar{\boldsymbol{b}} \\ &= \begin{bmatrix} \sum_{i=1}^{M_{sv}} \bar{\beta}_{i,x} k\left(\mathbf{p}_i, \mathbf{p}_j\right) \\ \sum_{i=1}^{M_{sv}} \bar{\beta}_{i,y} k\left(\mathbf{p}_i, \mathbf{p}_j\right) \end{bmatrix} + \begin{bmatrix} \bar{b}_x \\ \bar{b}_y \end{bmatrix} \\ &= \bar{\boldsymbol{\beta}}^T \mathbf{k}_{\mathbf{p}_j} + \bar{\boldsymbol{b}} \end{aligned} \tag{23}$$

where $\bar{\boldsymbol{\beta}}_x = [\,\bar{\boldsymbol{\beta}}_{1,x} \cdots \bar{\boldsymbol{\beta}}_{M_{sv},x}\,]^T$, $\bar{\boldsymbol{\beta}}_y = [\,\bar{\boldsymbol{\beta}}_{1,y} \cdots \bar{\boldsymbol{\beta}}_{M_{sv},y}\,]^T$, $\bar{\boldsymbol{\beta}} = [\,\bar{\boldsymbol{\beta}}_x\ \bar{\boldsymbol{\beta}}_y\,]$, $\bar{\boldsymbol{b}} = [\,\bar{\boldsymbol{b}}_x\ \bar{\boldsymbol{b}}_y\,]^T$, and $\mathbf{k}_{\mathbf{p}_j}$ is the vector of kernel values evaluated at $\mathbf{p}_j$ and training points $\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_{M_{sv}}\}$, i.e., $\mathbf{k}_{\mathbf{p}_j} = [\,k(\mathbf{p}_1, \mathbf{p}_j) \cdots k(\mathbf{p}_{M_{sv}}, \mathbf{p}_j)\,]^T$.

Remark

1) Since the problem considered herein is convex, Newton–Raphson method given in Algorithm 2 converges to an optimal point regardless of the choice of initial point.
2) The proposed method is similar to [21] in that
    a) each node collects measurement vector from all other nodes, and
    b) the anchor nodes send their vector back to a central sink node and the SVR is trained there.

    But the proposed method differs from [21] in that
    a) the proposed method uses proximity, while [21] uses RSS as a measurement;
    b) the proposed method sends the trained model back to the sensor nodes and the sensors node use it to estimate their own location. Instead, [21] sends the RSS vectors of all nodes to the sink, in which the

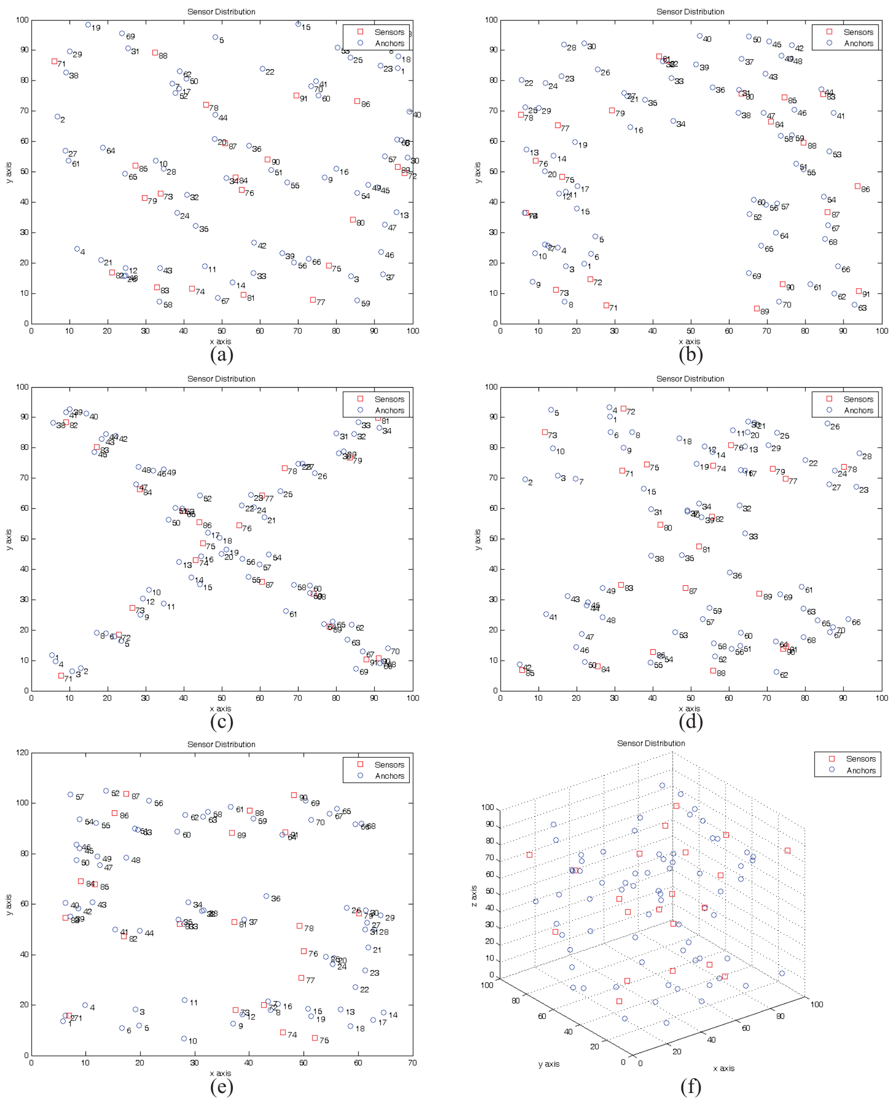


Fig. 1. Example of networks topology used in the simulation. (a) Isotropic network topology. (b)–(f) Anisotropic network topologies. (b) U-shape. (c) X-shape. (d) I-shape. (e) S-shape. (f) 3-D anisotropic network topology.

TABLE II
COMPARISON OF THE LOCATION ERROR FOR THE ISOTROPIC NETWORK

Location Error (M)		DV-hop	PDM	CSVR	LSVM	ELM	LMSVR
Case A	Avg.	**5.5478**	16.988	8.5292	21.3895	7.9247	7.4740
	Best	**4.5132**	7.8460	5.7699	17.5960	6.2984	5.1166
	Worst	**6.6069**	34.5138	15.8583	27.0423	11.0068	12.0622
Case B	Avg.	**4.7274**	20.3541	6.2224	20.4864	6.5185	5.8652
	Best	4.5313	7.7389	4.2275	16.7932	5.3256	**3.8494**
	Worst	**5.0320**	60.0526	8.5654	23.9145	8.5377	8.1487
Case C	Avg.	4.7408	19.8681	2.1875	15.4271	3.1032	**2.0090**
	Best	4.6426	6.5476	1.9571	14.0130	2.7135	**1.7303**
	Worst	4.7853	48.0706	2.4139	17.5547	3.6664	**2.2770**

SVR model is applied to estimate the locations of the sensors;

c) the proposed method uses MSVR, while [21] uses C (complex-valued) SVR. In case of a 3-D localization problem, MSVR should be extended to the 3-D version and CSVR should employ its quaternionic form [41].

3) It is worth noting that the use of the ε-insensitive loss function L_ε sparsifies the training of the SVR and reduces the number of support vectors. The sparsity in the WSN localization is more important than in other applications since the sparsity reduces not only the computational burden but also the communication overhead.

TABLE III
COMPARISON OF THE LOCATION ERROR FOR THE ANISOTROPIC NETWORK (U-SHAPE)

Location	Error (M)	DV-hop	PDM	CSVR	LSVM	ELM	LMSVR
Case A	Avg.	9.3674	13.0430	7.8681	19.5689	**6.3321**	6.8949
	Best	6.9835	5.8842	5.4642	16.6592	**5.1796**	5.4215
	Worst	12.2835	30.2201	11.4708	21.8604	**7.3806**	8.8258
Case B	Avg.	8.1751	16.3485	6.9550	19.1348	6.9807	**6.5906**
	Best	7.0327	6.1106	5.3124	17.3581	5.5944	**4.6840**
	Worst	9.2871	47.3090	9.5191	21.6518	**8.3611**	8.4881
Case C	Avg.	7.9063	14.5222	2.7390	16.1984	2.8619	**2.7260**
	Best	7.3852	8.1336	2.5138	15.1089	2.4497	**2.3950**
	Worst	8.2875	29.8465	**3.0762**	16.9718	3.1438	3.3779

TABLE IV
COMPARISON OF THE LOCATION ERROR FOR THE ANISOTROPIC NETWORK (X-SHAPE)

Location	Error (M)	DV-hop	PDM	CSVR	LSVM	ELM	LMSVR
Case A	Avg.	6.0378	10.9699	5.1516	16.2331	4.5749	**4.4368**
	Best	5.7159	3.8162	**3.6469**	15.2988	4.3253	3.9233
	Worst	6.6239	46.9052	9.7104	17.6150	5.1640	**5.0329**
Case B	Avg.	6.2371	11.4320	4.7059	16.0991	4.7693	**4.4467**
	Best	5.8868	4.4477	**3.4898**	15.4857	4.2330	3.8132
	Worst	6.6245	28.3311	6.7826	16.7788	**5.0535**	5.5190
Case C	Avg.	6.1332	14.0497	2.7865	14.8297	3.0112	**2.6758**
	Best	6.0213	5.6275	2.4947	14.4681	2.6819	**2.3986**
	Worst	6.4069	40.3207	4.1826	15.1009	3.2893	**2.9702**

TABLE V
COMPARISON OF THE LOCATION ERROR FOR THE ANISOTROPIC NETWORK (I-SHAPE)

Location	Error (M)	DV-hop	PDM	CSVR	LSVM	ELM	LMSVR
Case A	Avg.	**6.3501**	14.9895	7.5883	20.0519	6.8752	6.6032
	Best	5.3686	**5.3271**	5.4999	17.5197	5.8861	5.6403
	Worst	8.6492	43.8777	13.1763	22.3899	**7.5714**	8.1589
Case B	Avg.	**5.4883**	13.1054	6.4208	20.5719	6.5468	6.2638
	Best	5.0616	7.5885	**4.9516**	17.8675	5.6818	5.1523
	Worst	**6.1428**	33.1285	9.2037	23.0265	7.2081	7.9753
Case C	Avg.	5.4939	15.5360	2.4971	17.2709	2.8504	**2.4970**
	Best	5.3462	15.5360	2.1956	15.6182	2.6477	**2.1455**
	Worst	5.6666	7.4019	**3.0705**	19.0455	3.1516	3.1360

IV. EXPERIMENTAL RESULTS

Simulations conducted to compare the performance of the proposed method with the performance of previous methods are described in this section.

A. Simulation Setting

In the proposed method, we assumed two conditions: 1) only proximity information is used for the measurement, and 2) the communication range of each node is set to be the same value of R for both the anchor and sensor nodes. For a fair comparison, DV-hop [7], PDM [11], CSVR [21], LSVM [27], and ELM [16], [17] were used as the previous methods since they also satisfy the above two conditions. The proposed method is denoted as LMSVR, which represents localization based on MSVR, for convenience. All simulations were performed on a PC with an Intel 2.4-GHz CPU and 2 GB of memory, and all of the competing methods (DV-hop, PDM, CSVR, LSVM, ELM, and LMSVR) were implemented in MATHLAB version 7.4. ELM source code was downloaded from http://www.ntu.edu.sg/home/egbhuang/ and the other methods were implemented by us. In the simulation, both 2-D and 3-D environments are considered. The 2-D environment is 100×100 m^2 large and sensor and anchor nodes were distributed according to five different distributions as shown in Fig. 1(a)–(e). The first one implies an isotropic network, while the others are the examples of anisotropic networks. In the figures, the blue circles (○) denotes the sensor nodes while the rectangles (□) denote the anchor nodes. To check the extendibility of the competing methods, they

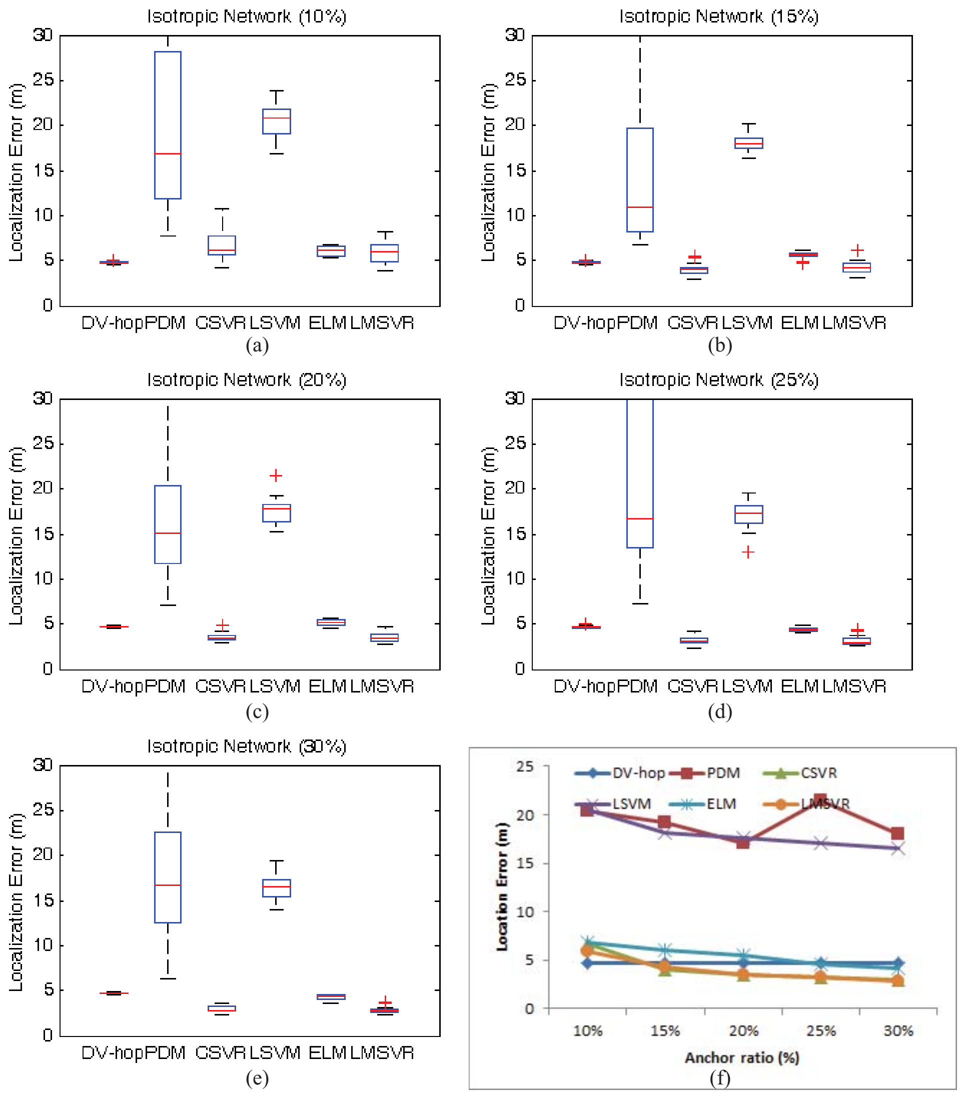


Fig. 2. Comparison of the average location estimation error for the isotropic network under various anchor populations. (a) 10% anchor nodes. (b) 15% anchor nodes. (c) 20% anchor nodes. (d) 25% anchor nodes. (e) 30% anchor nodes. (f) Overall summarization.

are applied to the localization of 3-D environment. The 3-D environment is $100 \times 100 \times 100$ m^3 large and sensor and anchor nodes were distributed uniformly over the whole space as shown in Fig. 1(f). We considered three different cases for the simulation.

1) In the first simulation, three sensor networks with different node densities were considered: Case A, a sparse network with 21 anchor nodes and 70 sensor nodes; Case B, an intermediate network with 21 anchor nodes and 210 sensor nodes; and Case C, a dense network with 100 anchor nodes and 400 sensor nodes. The communication range R for all nodes was assumed to be the same constant value.
2) In the second simulation, the network size was fixed to 210, and the anchor node ratio was varied from 10% to 30%, and the variation in performance was observed. The communication range was assumed to be the same constant value.
3) In the third simulation, more realistic environments were considered. The communication range R for all nodes was set to a modest value, and an irregular radio propagation model was used. The network size was fixed to 210 and the performance was assessed while the anchor node ratio was varied from 10% to 30%. In all of the simulations, the LMSVR could be trained by either the interior point method or the Newton–Raphson method since they produced the same result.

The parameters that we used in simulation are given in Table I. But the number of hidden node for ELM is empirically selected. In all of the simulations, the LMSVR could be trained by either the interior point method or the Newton–Raphson method since they produced the same result. The simulation dataset are available in [42].

B. Performance

In the first simulation, the communication range R for all nodes was set to 20. For each topology, ten independent simulation runs were made. Tables II–VII compare the performance of the competing algorithms, and the best results

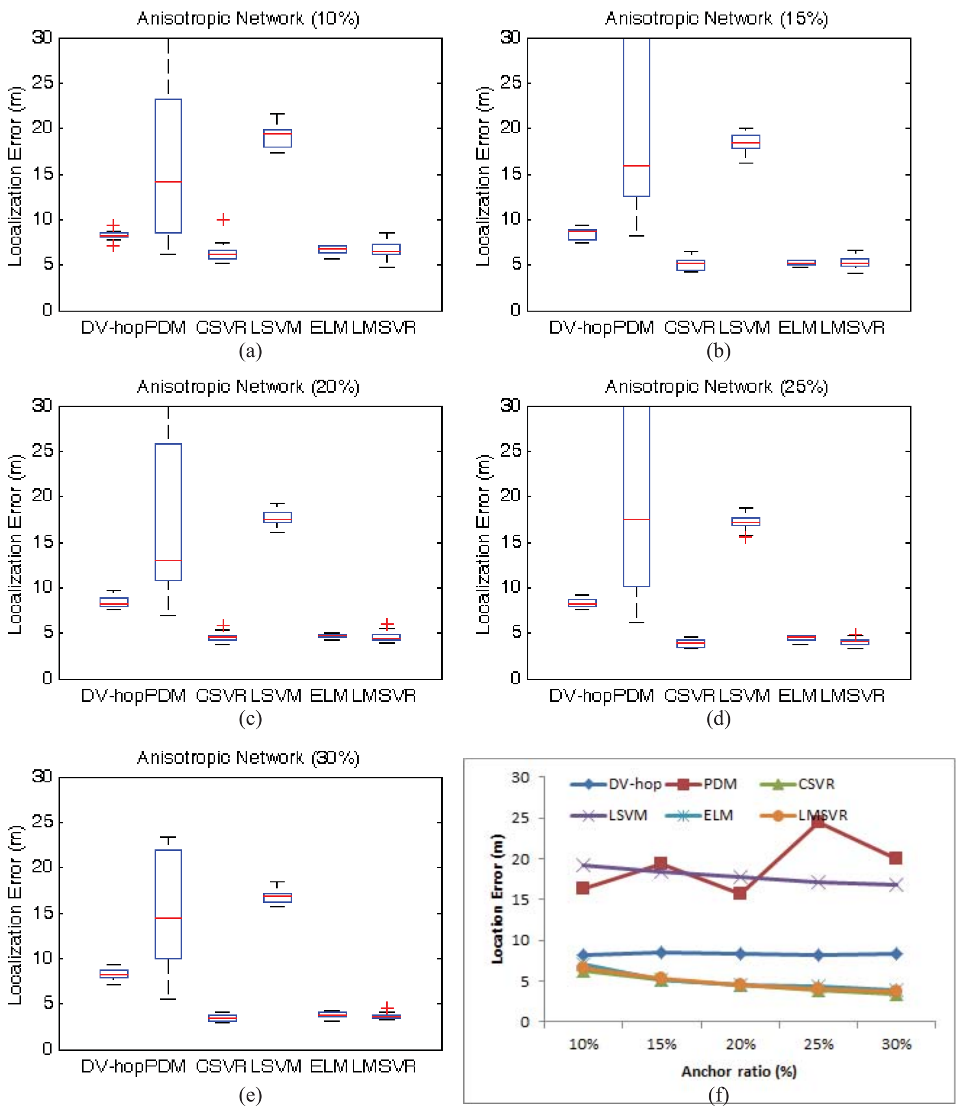


Fig. 3. Comparison of the average location estimation error for the anisotropic network under various anchor populations. (a) 10% anchor nodes. (b) 15% anchor nodes. (c) 20% anchor nodes. (d) 25% anchor nodes. (e) 30% anchor nodes. (f) Overall summarization.

are boldfaced for all the cases. For the isotropic network, the performance of most of the competing methods improved with increasing network density, except for the PDM method. For the anisotropic network, however, a substantial performance improvement was observed only for the CSVR, ELM, and LMSVR with increasing network density. The other methods did not exhibit satisfactory improvements. From the tables, it is apparent that, overall, LMSVR and ELM outperform the previous methods except for isotropic or near-isotropic (I-shaped) networks. CSVR also works well, but it is obvious that it is not as good as LMSVR and ELM. If we compare LMSVR and ELM, they sometimes outperformed and sometimes underperformed each other depending on the topology and they almost tied. CSVR was not applied here in the 3-D experiment.

The proposed method is compared with the competing methods in terms of testing time after training. For the five different topologies given in Fig. 1, 300–500 runs were made, and the average test time is listed in Table VIII. The five topologies include both isotropic and anisotropic ones. DV-hop and PDM are much faster than the proposed one, but they have high location error for anisotropic networks. CSVR and ELM show good accuracy but their execution times are longer than that of LMSVR. The proposed LMSVR demonstrated good performance both in accuracy and time complexity.

In the second, the total size of the network was fixed to 210 and the various anchor ratios were considered. The communication range R for all nodes was set to 20. Localization performances were compared using box plots for the isotropic and anisotropic networks in Figs. 2 and 3, respectively. The box plots include the minimum and maximum of the localization error, the median, and the 25th and 75th percentiles of 10 independent simulations.

As in the first simulation, the proposed LMSVR outperformed the previous methods not only on average but also in the best and worst cases. The proposed method also had a more consistent performance than the others, as indicated by the box plots. It is evident from Figs. 2

TABLE VI
COMPARISON OF THE LOCATION ERROR FOR THE ANISOTROPIC NETWORK (S-SHAPE)

Location Error (M)		DV-hop	PDM	CSVR	LSVM	ELM	LMSVR
Case A	Avg.	7.6456	12.6369	6.8614	22.0474	**5.8166**	6.4177
	Best	6.6062	6.6753	**4.9477**	18.4216	5.1210	5.3605
	Worst	8.4908	25.0879	9.8131	26.1475	**6.9410**	7.4406
Case B	Avg.	7.1940	13.3413	6.4854	22.5961	**6.1711**	6.9395
	Best	6.3828	5.7132	**4.9577**	16.8395	5.2521	5.5268
	Worst	8.0432	38.1188	8.7843	25.3201	**6.8432**	7.7991
Case C	Avg.	7.0670	20.6354	**2.7364**	20.1107	2.9667	2.7708
	Best	6.4860	6.2860	2.4596	18.4083	2.7353	**2.4475**
	Worst	7.5423	37.5925	3.1792	21.8409	3.1798	**3.1174**

TABLE VII
COMPARISON OF THE LOCATION ERROR FOR THE ANISOTROPIC NETWORK (3-D SPACE)

Location Error (M)		DV-hop	PDM	LSVM	ELM	LMSVR
Case B	Avg.	16.4895	29.2619	31.5002	13.6066	**13.1634**
	Best	15.0901	**11.0165**	29.0295	12.2294	11.5151
	Worst	18.4854	59.4486	33.2371	15.5545	**14.7813**
Case C	Avg.	14.8873	43.2283	28.0814	**7.5214**	8.0106
	Best	14.0452	22.394	27.1683	**7.1793**	7.4239
	Worst	15.8653	59.7547	28.8152	**7.8102**	9.2209

TABLE VIII
COMPARISON OF THE AVERAGE EXECUTION TIME FOR LOCALIZATION PROBLEM

Execution Time (s)	DV-hop	PDM	CSVR	LSVM	ELM	LMSVR
Case 1	0.000002	0.000844	0.046719	0.056194	1.296551	0.007146
Case 2	0.000002	0.001813	0.140600	0.166196	4.775547	0.023833
Case 3	0.000005	0.006500	1.323587	1.077331	33.372562	0.270135

and 3(f) that the proposed method yielded better localization results as the anchor ratio increased, and this might be because the number of training datasets for the MSVR model increased as the anchor ratio increased, resulting in a more accurate MSVR model for the given network. However, the superiority of the proposed method to the previous methods was always maintained regardless of the anchor ratios.

Finally, a more realistic environment was addressed in the third simulation. An irregular radio propagation model taken from [43] was employed to model the fluctuation of the communication range R due to the multipath channel in the real environment. More specifically, the communication ranges of each sensor and anchor node were not held constant. The communication range of each node was irregular, and the irregularity is represented by

$$(1 - \mathrm{DOI})R \leq \text{range} \leq R \tag{24}$$

where DOI means the degree of irregularity. Fig. 4 shows an example of the radio propagation model for each sensor and anchor node.

The number of sensor and anchor nodes was fixed at 210, and the anchor ratio was varied from 10% to 30%. The DOI value was assumed to be 0.2, and R was set at 10. The localization performances for the isotropic and anisotropic networks are compared in the box plots shown in Figs. 5 and 6, respectively. The box plots are representative of 10 independent iterations of the simulation.

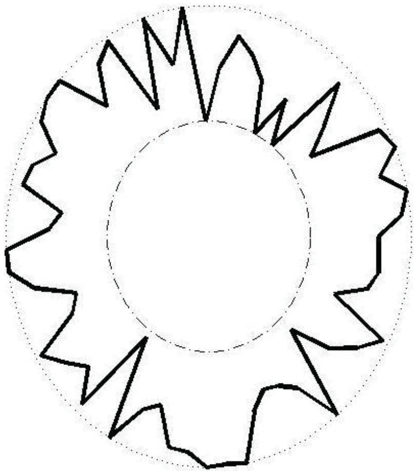


Fig. 4. Irregular radio propagation model.

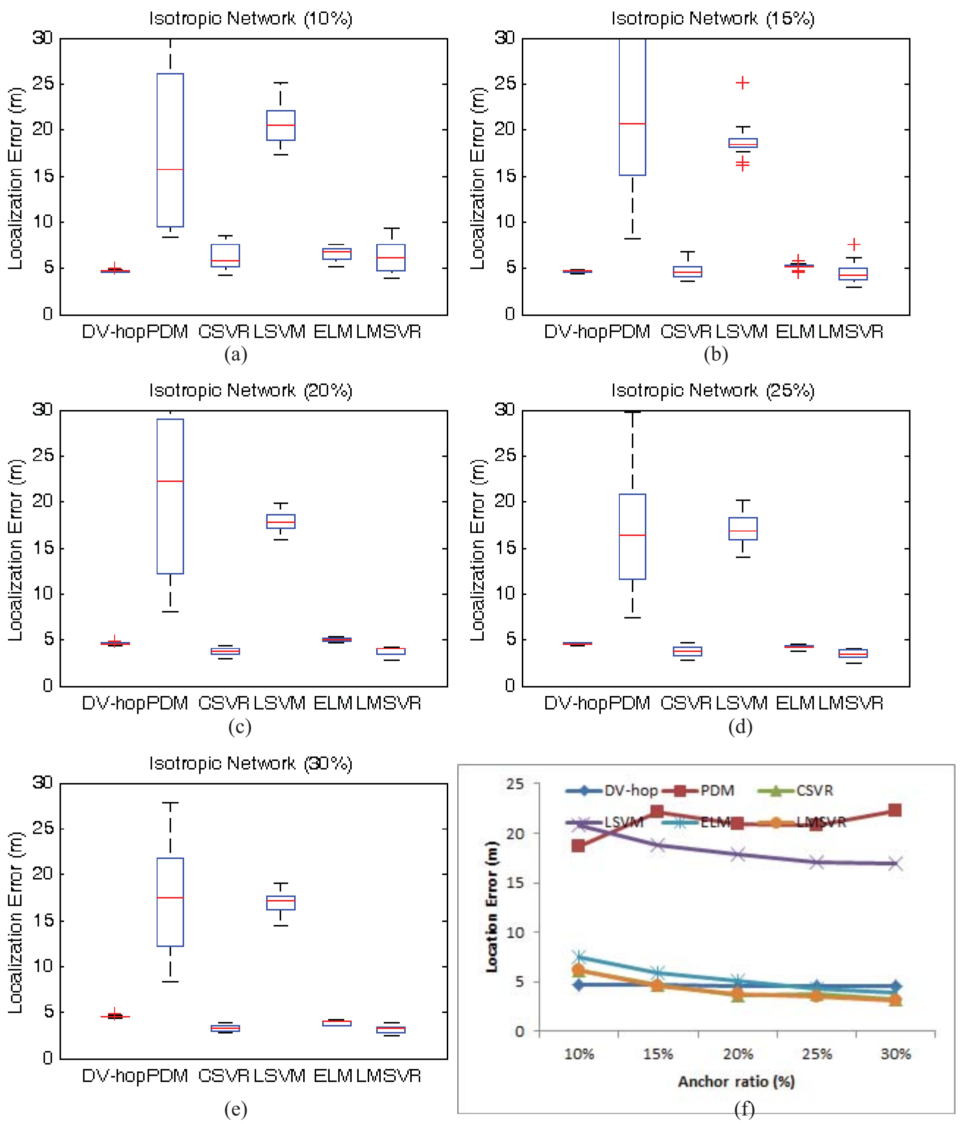


Fig. 5. Comparison of the average location estimation error for the isotropic network under various anchor populations with noisy radio propagation. (a) 10% anchor nodes. (b) 15% anchor nodes. (c) 20% anchor nodes. (d) 25% anchor nodes. (e) 30% anchor nodes. (f) Overall summarization.

TABLE IX
AVERAGE AMOUNT OF INFORMATION TO BE TRANSFERRED

Amount of Information	ELM	LMSVR
Case 1	4.86×10^4	169.50
Case 2	6.53×10^4	169.46
Case 3	16.86×10^4	789.52

As in the previous simulations, the proposed LMSVR outperformed the previous methods on average, and demonstrated a more consistent performance than the others, as indicated by the box plots. It is apparent that the localization performances were not as good as those with a perfect communication range. However, the proposed method exhibited better and more robust localization performances than the previous methods for all other conditions.

C. Other Considerations

CSVR, LMSVR, and ELM demonstrate excellent localization performance compared with other competing methods. As stated above, however, the application of CSVR to WSN localization is limited because most problems in real-world applications could be 3-D. The complex valued CSVR cannot be applied to 3-D localization problem.

ELM is also a good machine learning technique. It is very easy to use and has fast learning algorithm. For the WSN localization problem, however, ELM cannot be preferred to LMSVR because the larger amount of information should be transmitted to the sink node in ELM than in LMSVR. The transmission of large amount of information is a big problem in the WSN due the limited channel capacity, and the problem gets worse as the number of dimension increases. For reference, the mean amount of information that should be transferred is listed in Table IX for the simulation 1. The amount of information is computed by

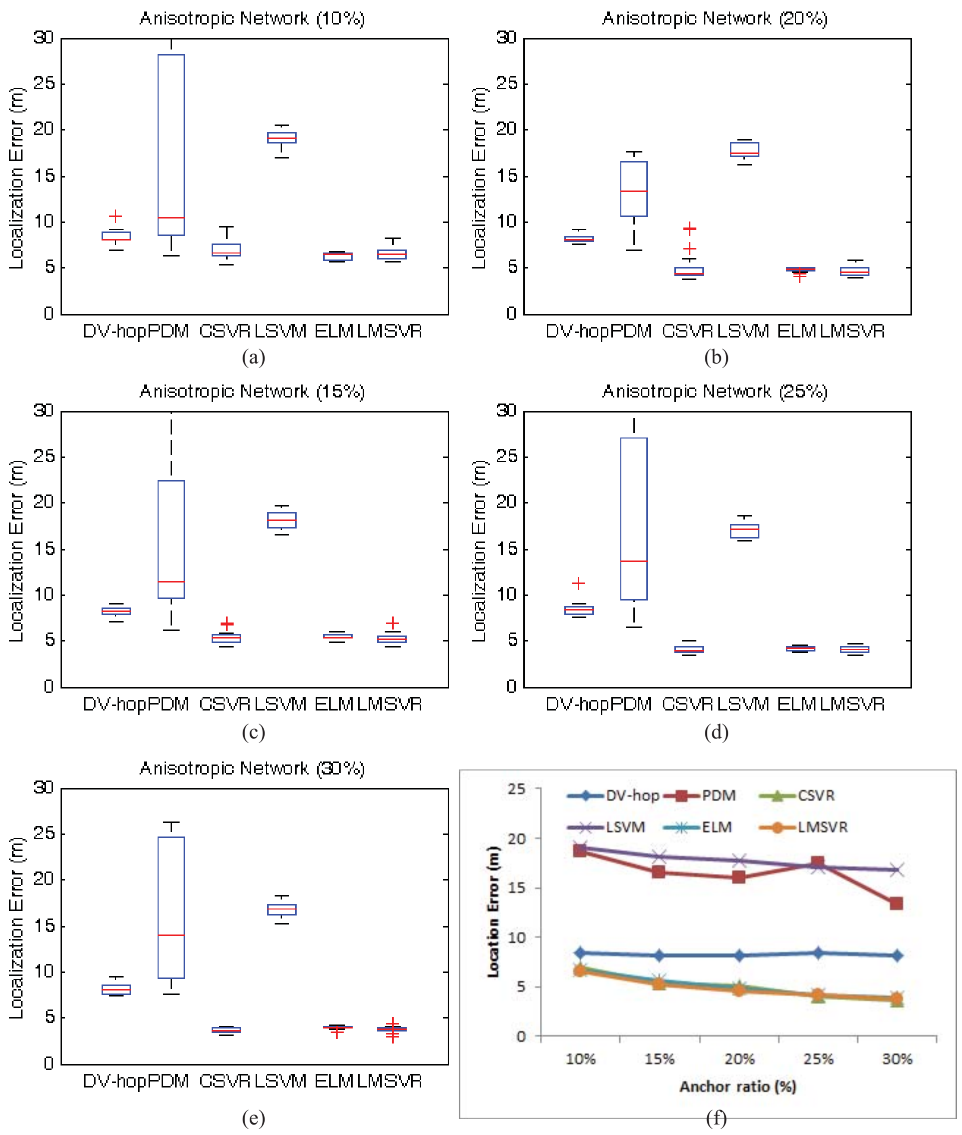


Fig. 6. Comparison of the average location estimation error for the anisotropic network under various anchor populations with noisy radio propagation. (a) 10% anchor nodes. (b) 15% anchor nodes. (c) 20% anchor nodes. (d) 25% anchor nodes. (e) 30% anchor nodes. (f) Overall summarization.

(input dimension) × (# of support vector) + (dimension of location) × (# of support vector + 1) for LMSVR and (input dimension + 1) × (# of hidden node) + (dimension of location) × (# of hidden node) for ELM. This table shows that LMSVR could be a choice for WSN localization problem.

V. Conclusion

A novel range-free localization algorithm for WSNs was proposed in this paper. The proposed method is based on MSVR learning, which is an extension of SVR, one of the most popular machine learning algorithms. In the proposed method, each sensor node estimates its location using the MSVR model trained with anchor nodes. In the MSVR model, the location of the node is predicted by information related to the proximity of the node to all of the anchor nodes. Two kinds of optimization techniques were proposed to identify the optimal MSVR model. In other words, the localization problem was formulated as a regression problem and solved using the MSVR model optimized via techniques including convex optimization. The proposed method was then applied to both isotropic and anisotropic networks. The simulation results demonstrated the superior performance of the proposed method compared to previous methods.

Appendix

First, let us define the following scalar, matrix and vectors

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{2,2} & \cdots & a_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m,1} & a_{m,2} & \cdots & a_{m,n} \end{bmatrix}_{m\times n} = \begin{bmatrix} \mathbf{a}_1^T \\ \mathbf{a}_2^T \\ \vdots \\ \mathbf{a}_m^T \end{bmatrix}_{m\times n}$$
$$= \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 & \cdots & \mathbf{A}_n \end{bmatrix}_{m\times n}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}_{n\times 1} \qquad b = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}_{m\times 1}$$

$$y = \mathbf{Ax} + \mathbf{b} = \begin{bmatrix} \mathbf{a}_1^T\mathbf{x} + b_1 \\ \mathbf{a}_2^T\mathbf{x} + b_2 \\ \vdots \\ \mathbf{a}_m^T\mathbf{x} + b_m \end{bmatrix}_{m\times 1}$$

$$\mathbf{z} = \|\mathbf{y}\| = \sqrt{\sum_{i=1}^{m} \left(\mathbf{a}_i^T\mathbf{x} + b_i\right)^2}. \tag{A.1}$$

Then, the derivative of z is represented as

$$\frac{\partial z}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial z}{\partial \mathbf{x}_1} \\ \frac{\partial z}{\partial \mathbf{x}_2} \\ \vdots \\ \frac{\partial z}{\partial \mathbf{x}_n} \end{bmatrix}_{n\times 1} = \begin{bmatrix} z^{-1}\left(\sum\limits_{i=1}^{m} a_{i,1}\left(a_i^T x + b_i\right)\right) \\ z^{-1}\left(\sum\limits_{i=1}^{m} a_{i,2}\left(a_i^T x + b_i\right)\right) \\ \vdots \\ z^{-1}\left(\sum\limits_{i=1}^{m} a_{i,n}\left(a_i^T x + b_i\right)\right) \end{bmatrix}$$

$$= \begin{bmatrix} z^{-1}\mathbf{A}_1^T y \\ z^{-1}\mathbf{A}_2^T y \\ \vdots \\ z^{-1}\mathbf{A}_n^T y \end{bmatrix}$$

$$= z^{-1}\mathbf{A}^T y = \frac{1}{\|\mathbf{Ax}+\mathbf{b}\|}\mathbf{A}^T\left(\mathbf{Ax}+\mathbf{b}\right). \tag{A.2}$$

Thus

$$\frac{\partial}{\partial \mathbf{x}}\|\mathbf{Ax}+\mathbf{b}\| = \frac{1}{\|\mathbf{Ax}+\mathbf{b}\|}\mathbf{A}^T\left(\mathbf{Ax}+\mathbf{b}\right). \tag{A.3}$$

Next, let us define two vectors

$$\mathbf{w} = \frac{1}{\|\mathbf{Ax}+\mathbf{b}\|}\mathbf{A}^T\mathbf{Ax} = \frac{1}{\|\mathbf{Ax}+\mathbf{b}\|}\begin{bmatrix} \mathbf{A}_1^T\sum\limits_{i=1}^{n}\mathbf{A}_i x_i \\ \mathbf{A}_2^T\sum\limits_{i=1}^{n}\mathbf{A}_i x_i \\ \vdots \\ \mathbf{A}_n^T\sum\limits_{i=1}^{n}\mathbf{A}_i x_i \end{bmatrix}_{n\times 1}$$

$$\frac{\partial v}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial v_1}{\partial \mathbf{x}_1} & \frac{\partial v_2}{\partial \mathbf{x}_1} & \cdots & \frac{\partial v_n}{\partial \mathbf{x}_1} \\ \frac{\partial v_1}{\partial \mathbf{x}_2} & \frac{\partial v_2}{\partial \mathbf{x}_2} & \cdots & \frac{\partial v_n}{\partial \mathbf{x}_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial v_1}{\partial \mathbf{x}_n} & \frac{\partial v_2}{\partial \mathbf{x}_n} & \cdots & \frac{\partial v_n}{\partial \mathbf{x}_n} \end{bmatrix}_{n\times n} = \begin{bmatrix} -\frac{\frac{\partial z}{\partial \mathbf{x}_1}\mathbf{A}_1^T\mathbf{b}}{z^2} & -\frac{\frac{\partial z}{\partial \mathbf{x}_1}\mathbf{A}_2^T\mathbf{b}}{z^2} & \cdots & -\frac{\frac{\partial z}{\partial \mathbf{x}_1}\mathbf{A}_n^T b}{z^2} \\ -\frac{\frac{\partial z}{\partial \mathbf{x}_2}\mathbf{A}_1^T\mathbf{b}}{z^2} & -\frac{\frac{\partial z}{\partial \mathbf{x}_2}\mathbf{A}_2^T\mathbf{b}}{z^2} & \cdots & -\frac{\frac{\partial z}{\partial \mathbf{x}_2}\mathbf{A}_n^T b}{z^2} \\ \vdots & \vdots & \ddots & \vdots \\ -\frac{\frac{\partial z}{\partial \mathbf{x}_n}\mathbf{A}_1^T\mathbf{b}}{z^2} & -\frac{\frac{\partial z}{\partial \mathbf{x}_n}\mathbf{A}_2^T\mathbf{b}}{z^2} & \cdots & -\frac{\frac{\partial z}{\partial \mathbf{x}_n}\mathbf{A}_n^T b}{z^2} \end{bmatrix}_{n\times n}$$

$$= -\frac{1}{z^3}\begin{bmatrix} \mathbf{A}_1^T(\mathbf{Ax}+\mathbf{b})\mathbf{A}_1^T\mathbf{b} & \mathbf{A}_1^T(\mathbf{Ax}+\mathbf{b})\mathbf{A}_2^T\mathbf{b} & \cdots & \mathbf{A}_1^T(\mathbf{Ax}+\mathbf{b})\mathbf{A}_n^T\mathbf{b} \\ \mathbf{A}_2^T(\mathbf{Ax}+\mathbf{b})\mathbf{A}_1^T\mathbf{b} & \mathbf{A}_2^T(\mathbf{Ax}+\mathbf{b})\mathbf{A}_2^T\mathbf{b} & \cdots & \mathbf{A}_2^T(\mathbf{Ax}+\mathbf{b})\mathbf{A}_n^T\mathbf{b} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}_n^T(\mathbf{Ax}+\mathbf{b})\mathbf{A}_1^T\mathbf{b} & \mathbf{A}_n^T(\mathbf{Ax}+\mathbf{b})\mathbf{A}_2^T\mathbf{b} & \cdots & \mathbf{A}_n^T(\mathbf{Ax}+\mathbf{b})\mathbf{A}_n^T\mathbf{b} \end{bmatrix}_{n\times n}$$

$$= -\frac{1}{z^3}\mathbf{A}^T(\mathbf{Ax}+\mathbf{b})\left(\mathbf{A}^T b\right)^T$$

$$= -\frac{\mathbf{A}^T(\mathbf{Ax}+\mathbf{b})b^T A}{z^3} \tag{A.5}$$

$$\frac{\partial w}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial w_1}{\partial \mathbf{x}_1} & \frac{\partial w_2}{\partial \mathbf{x}_1} & \cdots & \frac{\partial w_n}{\partial \mathbf{x}_1} \\ \frac{\partial w_1}{\partial \mathbf{x}_2} & \frac{\partial w_2}{\partial \mathbf{x}_2} & \cdots & \frac{\partial w_n}{\partial \mathbf{x}_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial w_1}{\partial \mathbf{x}_n} & \frac{\partial w_2}{\partial \mathbf{x}_n} & \cdots & \frac{\partial w_n}{\partial \mathbf{x}_n} \end{bmatrix}_{n\times n} = \begin{bmatrix} \frac{z\mathbf{A}_1^T\mathbf{A}_1 - \frac{\partial z}{\partial \mathbf{x}_1}\mathbf{A}_1^T\sum_{i=1}^{n}\mathbf{A}_i x_i}{z^2} & \frac{z\mathbf{A}_2^T\mathbf{A}_1 - \frac{\partial z}{\partial \mathbf{x}_1}\mathbf{A}_2^T\sum_{i=1}^{n}\mathbf{A}_i x_i}{z^2} & \cdots & \frac{z\mathbf{A}_n^T\mathbf{A}_1 - \frac{\partial z}{\partial \mathbf{x}_1}\mathbf{A}_n^T\sum_{i=1}^{n}\mathbf{A}_i x_i}{z^2} \\ \frac{z\mathbf{A}_1^T\mathbf{A}_2 - \frac{\partial z}{\partial \mathbf{x}_2}\mathbf{A}_1^T\sum_{i=1}^{n}\mathbf{A}_i x_i}{z^2} & \frac{z\mathbf{A}_2^T\mathbf{A}_2 - \frac{\partial z}{\partial \mathbf{x}_2}\mathbf{A}_2^T\sum_{i=1}^{n}\mathbf{A}_i x_i}{z^2} & \cdots & \frac{z\mathbf{A}_n^T\mathbf{A}_2 - \frac{\partial z}{\partial \mathbf{x}_2}\mathbf{A}_n^T\sum_{i=1}^{n}\mathbf{A}_i x_i}{z^2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{z\mathbf{A}_1^T\mathbf{A}_n - \frac{\partial z}{\partial \mathbf{x}_n}\mathbf{A}_1^T\sum_{i=1}^{n}\mathbf{A}_i x_i}{z^2} & \frac{z\mathbf{A}_2^T\mathbf{A}_n - \frac{\partial z}{\partial \mathbf{x}_n}\mathbf{A}_2^T\sum_{i=1}^{n}\mathbf{A}_i x_i}{z^2} & \cdots & \frac{z\mathbf{A}_n^T\mathbf{A}_n - \frac{\partial z}{\partial \mathbf{x}_n}\mathbf{A}_n^T\sum_{i=1}^{n}\mathbf{A}_i x_i}{z^2} \end{bmatrix}_{n\times n}$$

$$= \frac{\mathbf{A}^T\mathbf{A}}{z} - \begin{bmatrix} \frac{\mathbf{A}_1^T(\mathbf{Ax}+\mathbf{b})\mathbf{A}_1^T\mathbf{Ax}}{z^3} & \frac{\mathbf{A}_1^T(\mathbf{Ax}+\mathbf{b})\mathbf{A}_2^T\mathbf{Ax}}{z^3} & \cdots & \frac{\mathbf{A}_1^T(\mathbf{Ax}+\mathbf{b})\mathbf{A}_n^T\mathbf{Ax}}{z^3} \\ \frac{\mathbf{A}_2^T(\mathbf{Ax}+\mathbf{b})\mathbf{A}_1^T\mathbf{Ax}}{z^3} & \frac{\mathbf{A}_2^T(\mathbf{Ax}+\mathbf{b})\mathbf{A}_2^T\mathbf{Ax}}{z^3} & \cdots & \frac{\mathbf{A}_2^T(\mathbf{Ax}+\mathbf{b})\mathbf{A}_n^T\mathbf{Ax}}{z^3} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\mathbf{A}_n^T(\mathbf{Ax}+\mathbf{b})\mathbf{A}_1^T\mathbf{Ax}}{z^3} & \frac{\mathbf{A}_n^T(\mathbf{Ax}+\mathbf{b})\mathbf{A}_2^T\mathbf{Ax}}{z^3} & \cdots & \frac{\mathbf{A}_n^T(\mathbf{Ax}+\mathbf{b})\mathbf{A}_n^T\mathbf{Ax}}{z^3} \end{bmatrix}_{n\times n} = \frac{\mathbf{A}^T\mathbf{A}}{z} - \frac{\mathbf{A}^T(\mathbf{Ax}+\mathbf{b})\left(\mathbf{Ax}\right)^T A}{z^3} \tag{A.6}$$

$$= \begin{bmatrix} \frac{\mathbf{A}_1^T \sum_{i=1}^{n} \mathbf{A}_i x_i}{z} \\ \frac{\mathbf{A}_2^T \sum_{i=1}^{n} \mathbf{A}_i x_i}{z} \\ \vdots \\ \frac{\mathbf{A}_n^T \sum_{i=1}^{n} \mathbf{A}_i x_i}{z} \end{bmatrix}_{n\times 1} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{bmatrix}_{n\times 1}$$

$$v = \frac{1}{\|\mathbf{Ax}+\mathbf{b}\|}\mathbf{A}^T\mathbf{b} = \frac{1}{\|\mathbf{Ax}+\mathbf{b}\|}\begin{bmatrix} \mathbf{A}_1^T\mathbf{b} \\ \mathbf{A}_2^T\mathbf{b} \\ \vdots \\ \mathbf{A}_n^T\mathbf{b} \end{bmatrix}_{n\times 1}$$

$$= \begin{bmatrix} \frac{\mathbf{A}_1^T\mathbf{b}}{z} \\ \frac{\mathbf{A}_2^T\mathbf{b}}{z} \\ \vdots \\ \frac{\mathbf{A}_n^T\mathbf{b}}{z} \end{bmatrix}_{n\times 1} = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{bmatrix}_{n\times 1}. \quad \text{(A.4)}$$

Then, the derivatives of w and v are represented in (A5).

Combining (A3), (A5), and (A6), the second derivative of z is represented as

$$\frac{\partial}{\partial \mathbf{x}}\left(\frac{\partial z}{\partial \mathbf{x}}\right) = \frac{\partial}{\partial \mathbf{x}}\left(\frac{1}{\|\mathbf{Ax}+\mathbf{b}\|}\mathbf{A}^T(\mathbf{Ax}+\mathbf{b})\right)$$
$$= \frac{\mathbf{A}^T\mathbf{A}}{\|\mathbf{Ax}+\mathbf{b}\|} - \frac{\mathbf{A}^T(\mathbf{Ax}+\mathbf{b})(\mathbf{Ax}+\mathbf{b})^T A}{\|\mathbf{Ax}+\mathbf{b}\|^3}. \quad \text{(A.7)}$$

Finally, the following equation is derived

$$\frac{\partial^2}{\partial \mathbf{x}^2}\|\mathbf{Ax}+\mathbf{b}\| = \frac{\mathbf{A}^T\mathbf{A}}{\|\mathbf{Ax}+\mathbf{b}\|} - \frac{\mathbf{A}^T(\mathbf{Ax}+\mathbf{b})(\mathbf{Ax}+\mathbf{b})^T A}{\|\mathbf{Ax}+\mathbf{b}\|^3}. \quad \text{(A.8)}$$

References

[1] S. Lee, H. Cha, and R. Ha, "Energy-aware location error handling for object tracking applications in wireless sensor networks," *Comput. Commun.*, vol. 30, no. 7, pp. 1443–1450, May 2007.

[2] A. Mainwaring, J. Polastre, R. Szewczyk, and D. Culler, "Wireless sensor networks for habitat monitoring," in *Proc. ACM Int. Workshop Sensor Netw. Appl.*, 2002, pp. 88–97.

[3] G. M. Hoffmann and C. J. Tomlin, "Mobile sensor network control using mutual information methods and particle filters," *IEEE Trans. Autom. Control*, vol. 55, no. 1, pp. 32–47, Jan. 2010.

[4] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, *Global Positioning System: Theory and Practice*. New York, USA: Springer-Verlag, 1993.

[5] G. Mao, B. Fidan, and B. D. O. Anderson, "Wireless sensor network localization techniques," *Comput. Netw.*, vol. 51, no. 10, pp. 2529–2553, Jul. 2007.

[6] Y. Shang, W. Ruml, Y. Zhang, and M. Fromherz, "Localization from connectivity in sensor networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 15, no. 11, pp. 961–974, Nov. 2004.

[7] D. Niculescu and B. Nath, "DV based positioning in ad hoc networks," *J. Telecommun. Syst.*, vol. 22, nos. 1–4, pp. 267–280, Jan. 2003.

[8] N. Bulusu, J. Heidemann, and D. Estrin, "Adaptive beacon placement," in *Proc. IEEE Int. Conf. Distrib. Comput. Syst.*, Apr. 2001, pp. 489–498.

[9] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher, "Range-free localization schemes for large scale sensor networks," in *Proc. Intl. Conf. Mobile Comput. Netw.*, 2003, pp. 81–95.

[10] D. Ma, M. J. Er, B. Wang, and H. B. Lim, "Range-free wireless sensor networks localization based on hop-count quantization," *Telecomm. Syst.*, vol. 50, no. 3, pp. 199–213, Jul. 2012.

[11] H. Lim and J. Hou, "Localization for anisotropic sensor networks," in *Proc. IEEE 24th Ann. Joint Conf. Comput. Commun. Soc.*, vol. 1. Mar. 2005, pp. 138–149.

[12] C. Cortes and V. Vapnik, "Support-vector networks," *Mach. Learn.*, vol. 20, no. 3, pp. 273–297, Sep. 1995.

[13] K. D. Brabanter, J. D. Brabanter, J. A. K. Suykens, and B. D. Moor, "Approximate confidence and prediction intervals for least squares support vector regression," *IEEE Trans. Neural Netw.*, vol. 22, no. 1, pp. 110–120, Jan. 2011.

[14] J. Lopez and J. R. Dorronsoro, "Simple proof of convergence of the SMO algorithm for different SVM variants," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 7, pp. 1142–1147, Jul. 2012.

[15] F. Bellocchio, S. Ferrari, V. Piuri, and N. A. Borghese, "Hierarchical approach for multiscale support vector regression," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 9, pp. 1448–1460, Sep. 2012.

[16] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *IEEE Trans. Syst., Man, Cybern., B, Cybern.*, vol. 42, no. 2, pp. 513–529, Feb. 2012.

[17] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, nos. 1–3, pp. 489–501, Dec. 2006.

[18] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *IEEE Trans. Neural Netw.*, vol. 17, no. 4, pp. 879–892, Jul. 2006.

[19] G.-B. Huang and L. Chen, "Convex incremental extreme learning machine," *Neurocomputing*, vol. 70, nos. 16–18, pp. 3056–3062, Oct. 2007.

[20] G.-B. Huang and L. Chen, "Enhanced random search based incremental extreme learning machine," *Neurocomputing*, vol. 71, nos. 16–18, pp. 3460–3468, Oct. 2008.

[21] A. Shilton, B. Sundaram, and M. Palaniswami, "Ad-hoc wireless sensor network localization using support vector regression," in *Proc. ICT Mobile Summit*, Jun. 2008, pp. 1–8.

[22] C. Wang, J. Chen, and Y. Sun, "Sensor network localization using kernel spectral regression," *Wireless Commun. Mobile Comput.*, vol. 10, no. 8, pp. 1045–1054, Aug. 2010.

[23] S. Yun, J. Lee, W. Chung, E. Kim, and S. Kim, "A soft computing approach to localization in wireless sensor networks," *Expert Syst. Appl.*, vol. 36, no. 4, pp. 7552–7561, Oct. 2009.

[24] X. He, Y. Xiao, and Y. Wang, "A LSSVR three-dimensional WSN nodes location algorithm based on RSSI," in *Proc. Int. Conf. Electr. Control Eng.*, Sep. 2011, pp. 1889–1895.

[25] D. Gu and H. Hu, "Spatial Gaussian process regression with mobile sensor networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 8, pp. 1279–1290, Aug. 2012.

[26] J. Liang, Z. Wang, and X. Liu, "Distributed estimation for discrete-time sensor networks with randomly varyig nonlinearities and missing measurements," *IEEE Trans. Neural Netw.*, vol. 22, no. 3, pp. 486–496, Mar. 2011.

[27] D. A. Tran and T. Nguyen, "Localization in wireless sensor networks based on support vector machines," *IEEE Trans. Parallel Distrib. Syst.*, vol. 19, no. 7, pp. 981–994, Jul. 2008.

[28] R. Samadian and S. M. Noorhosseini, "Probabilistic support vector machine localization in wireless sensor networks," *ETRI J.*, vol. 33, no. 6, pp. 924–934, Dec. 2011.

[29] R. Huan, Q. Chen, K. Mao, and Y. Pan, "A three-dimensional localization algorithm for wireless sensor network nodes based on SVM," in *Proc. Int. Conf. Green Circuits Syst.*, Jun. 2010, pp. 651–654.

[30] V. Feng, and S. Y. Chang, "Determination of wireless networks parameters through parallel hierarchical support vector machines," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 3, pp. 505–512, Mar. 2012.

[31] J. Lee, W. Chung, and E. Kim, "A new kernelized approach to wireless sensor network localization," *Inf. Sci.*, 2013, to be published.

[32] M. Sánchez-Fernández, M. de Prado-Cumplido, J. Arenas-García, and F. Pérez-Cruz, "SVM multiregression for nonlinear channel estimation in multiple-input multiple-output systems," *IEEE Trans. Signal Process.*, vol. 52, no. 8, pp. 2298–2307, Aug. 2004.

[33] S. Mehrotra, "On the implementation of a primal-dual interior point method," *SIAM J. Optim.*, vol. 2, no. 4, pp. 575–601, 1992.

[34] T. J. Ypma, "Historical development of the Newton-Raphson method," *SIAM Rev.*, vol. 37, no. 4, pp. 531–551, Dec. 1995.

[35] O. Chapelle, "Training a support vector machine in the primal," *Neural Comput.*, vol. 19, no. 5, pp. 1155–1178, May 2007.

[36] J. Lee, W. Chung, and E. Kim, "A new range-free localization method using quadratic programming," *Comput. Commun.*, vol. 34, no. 8, pp. 998–1010, Jun. 2011.

[37] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," Royal Holloway College, Univ. London, London, U.K., NeuroCOLT Tech. Rep. NC2-TR-1998-030, 1998.

[38] G. S. Kimeldorf and G. Wahba, "A correspondence between Bayesian estimation on stochastic processes and smoothing by splines," *Ann. Math. Stat.*, vol. 41, no. 2, pp. 495–502, 1970.

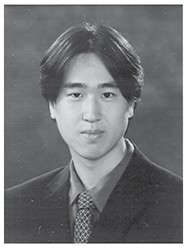
[39] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[40] CVX Research, Inc. (2009, Sep.). *CVX: Matlab Software for Disciplined Convex Programming, Version 1.2 Beta*, Austin, TX, USA [Online]. Available: http://cvxr.com/cvx/

[41] A. Shilto and D. T. H. Lai, "Quaternionic and complex-valued support vector regression for equalization and function approximation," in *Proc. Int. Joint Conf. Neural Netw.*, Orlando, FL, USA, Aug. 2007, pp. 920–925.

[42] *CILAB. Simulation Data set-JLeeTNNLS2013*. (2013, Mar.) [Online]. Available: http://cilab.yonsei.ac.kr/pubs/TNNLS13

[43] J. P. Sheu, P. C. Chen, and C. S. Hsu, "A distributed localization scheme for wireless sensor networks with improved grid-scan and vector-based refinement," *IEEE Trans. Mobile Comput.*, vol. 7, no. 9, pp. 1110–1123, Sep. 2008.



Jaehun Lee received the B.S., M.S., and Ph.D. degrees in electrical and electronic engineering from Yonsei University, Seoul, Korea, in 2005, 2007, and 2012, respectively.

He has been a Senior Research Engineer with the Convergence R&D Laboratory, LG Electronics, Seoul, since 2012. His current research interests include computational intelligence, machine learning and localization, and tracking in wireless sensor network.

Baehoon Choi received the B.S. degree from Yonsei University, Seoul, Korea, in 2010, where he is currently pursuing the Ph.D. degree, both in electrical and electronic engineering.

His current research interests include intelligence vehicle system, machine learning, and pattern recognition.

Euntai Kim was born in Seoul, Korea, in 1970. He received the B.S., M.S., and Ph.D. degrees in electronic engineering from Yonsei University, Seoul, in 1992, 1994, and 1999, respectively.

He was a full-time Lecturer with the Department of Control and Instrumentation Engineering, Hankyong National University, Kyonggi-do, Korea, from 1999 to 2002. Since 2002, he has been with the faculty of the School of Electrical and Electronic Engineering, Yonsei University, where he is currently a Professor. He was a Visiting Researcher at Berkeley Initiative in Soft Computing, UC at Berkeley, Berkeley, CA, USA. His current research interests include computational intelligence and statistical machine learning and their application to intelligent robot, vehicle, and machine vision.

Dr. Kim was the recipient of the Haedong Prize from the Institute of Institute of Electronics Engineers of Korea and Woo kwangbang's Prize from the Institute of Control, Automation, and Systems Engineers in 2003 and 2004, respectively. He is an Associate Editor of the *International Journal of Control and Systems*.