Support Vector Classification Strategies for Localization in Sensor Networks

Duc A. Tran
Department of Computer Science
University of Dayton
Dayton, OH 45469
Email: duc.tran@udayton.edu

Thinh Nguyen
Department of Electrical and Computer Engineering
Oregon State University
Corvallis, OR 97331
Email: thinhq@eecs.oregonstate.edu

***Abstract*— We consider the problem of estimating the geographic locations of nodes in a wireless sensor network where most sensors are without an effective self-positioning functionality. A solution to this localization problem is proposed, which uses Support Vector Machines (SVM) and mere connectivity information only. We investigate two versions of this solution, each employing a different multi-class SVM strategy. They are shown to perform well in various aspects such as localization error, processing efficiency, and effectiveness in addressing the border issue.**

I. Introduction

Wireless sensor networks are typically consisted of inexpensive sensing devices with limited resources. In most cases, sensors are not equipped with any GPS-like receiver, or when such an unit is installed it does not function due to environmental difficulties. On the other hand, knowing the geographic locations of the sensor nodes is critical to many tasks of a sensor network such as network management, event detection, geography-based query processing, and routing. Therefore, an important problem is to devise an accurate, efficient, and fast-converging technique for estimating the sensor locations given that the true location information is minimally or un- known.

To this problem, we explore the applicability of Support Vector Machines (SVM). SVM is a classification method with two main components: a kernel function and a set of support vectors. The support vectors are obtained via the training phase given the training data. New data is classified using a simple computation involving the kernel function and support vectors only. For sensor localization, we define a set of geographical regions in the sensor field and classify each sensor node into these regions. Then its location can be estimated inside the intersection of the containing regions. The training data is the set of beacons, and the kernel function is defined based on hop counts only.

A. Existing Work

There are many dimensions to categorize existing techniques, such as centralized vs. decentralized, beacons vs. beacon-less, and ranging vs. ranging-free.

In the centralized approach [1], [2], the information (e.g., connectivity, pair-wise distance measure) about the entire network into one place, where the collected information is processed centrally to estimate the sensors' locations. This approach is obviously impractical for large-scale sensor networks due to high computation and communication costs.

Examples of distributed techniques are the relaxation-based techniques ([3], [4]) and coordinate-system stitching techniques ([5]–[8]). Some other distributed techniques (e.g., [8]–[16]) assume the existence of nodes with known location, called the beacons, and extrapolate unknown locations from the beacon locations.

Most current techniques assume that the distance between two neighbor nodes can be measured via a ranging process. This process is subject to noise and incurs complexity/cost increasing with accuracy requirement. For a large sensor network with low-end sensors, it is often not affordable to equip them all with ranging capability. Few range-free techniques [6], [9], [16]–[19] have been proposed for this type of networks.

B. Our contribution

Our approach only assumes that beacon nodes exist and only connectivity information may be used for location estimation. It does not require a sensor to be able to hear directly from any beacon node. Compared to the existing ranging-free techniques, none (except one) satisfies all these assumptions. In [16], [19], a node must be able to hear directly from a large number (or all) of beacons; we relax this requirement. [17], [18] require specialized assisting moving subjects such as an aerial vehicle to generate light onto the sensor field or a mobile node to assist pair-wise distance measurements; we do not require any such additional devices.

The approach that shares the same requirements with our work is Diffusion [6], [9], where each node is repeatedly positioned as the centroid of its neighbors until convergence. Figure 1 illustrates two main problems of this approach: the convergence problem (i.e., many averaging loops result in long localization time and significant bandwidth consumption), and the border problem (i.e., nodes near the edge of the sensor field are poorly positioned). The latter also occurs in many existing techniques. Our technique is fast to converge and we will show in our performance study that it almost eliminates the border problem.

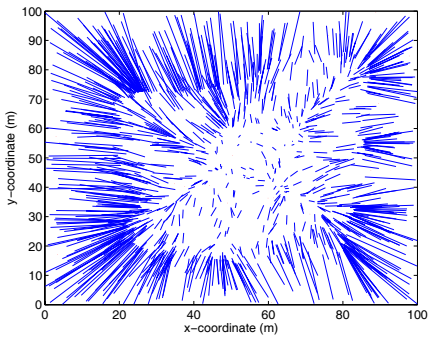


Fig. 1. Diffusion: 1000 sensors on a 100m x 100m field, with 50 random beacons. A line connects the true and estimated positions of each sensor node. Note the border problem after 10,000 averaging loops

C. Paper Organization

We provide a brief background on SVM in the next section. We describe how SVM can be applied to the sensor localization problem and propose two alternate solutions in Section III. Evaluation results are presented in Section IV. Finally, we conclude this paper with pointers to our future research in Section V.

II. Support Vector Machine Classification

Consider the problem of classifying data in a data space X into either one of two classes: G or $\neg G$ (not G). Suppose that each data point x has a feature vector $\vec{x}$ in some feature space $\vec{X} \subseteq \Re^n$. We are given k data points $x_1, x_2, ..., x_k$, called the "training points", with labels $y_1, y_2, ..., y_k$, respectively (where $y_i = 1$ if $x_i \in G$ and -1 otherwise). We need to predict whether a new data point x is in G or not.

Support Vector Machines (SVM) [20] is an efficient method to solve this problem. For the case of finite data space (e.g., location data of nodes in a sensor network), the steps typically taken in SVM are as follows:

- Define a kernel function K: $X \times X \rightarrow \Re$. This function must be symmetric and the $k \times k$ matrix $[K(x_i, x_j)]_{i,j=1}^{k}$ must be positive semi-definite (i.e., has non-negative eigenvalues)
- Maximize

$$W(\alpha) = \sum_{i=1}^{k} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{k} y_i y_j \alpha_i \alpha_j K(x_i, x_j) \quad (1)$$

  - subject to

$$\sum_{i=1}^{k} y_i \alpha_i = 0 \quad (2)$$

$$0 \leq \alpha_i \leq C, i \in [1, k] \quad (3)$$

Suppose that $\{\alpha_1^*, \alpha_2^*, ..., \alpha_k^*\}$ is the solution to this optimization problem. We choose $b = b^*$ such that $y_i h_K(x_i) = 1$ for all i with $0 < \alpha_i^* < C$. The training points corresponding to such (i, α_i^*)'s are called the *support vectors*. The decision rule to classify a data point x is: $x \in G$ iff $sign(h_K(x)) = 1$, where

$$h_K(x) = \sum_{i=1 \rightarrow k,\ x_i \ is\ a\ support\ vector} \alpha_i^* y_i K(x, x_i) + b^* \quad (4)$$

According to Mercer's theorem [20], there exists a feature space $\vec{X}$ where the kernel K defined above serves as the inner product of $\vec{X}$ (i.e., $K(x, z) = <\vec{x} \cdot \vec{z}>$ for every $x, z \in X$). The function $h_K(.)$ represents the hyperplane in $\vec{X}$ that maximally separates the training points in X (G points in the positive side of the plane, $\neg G$ points in the negative side). It is provable that SVM has bounded classification error when applied to test data. We present LSVM next.

III. LSVM: Localization based on SVM

A. Network model

We consider a large wireless sensor network of N nodes $\{S_1, S_2, ..., S_N\}$ deployed in a 2-d geographic area [0, D] $\times$ [0, D] ($D > 0$). (We assume 2 dimensions for simplicity, even though LSVM can work with any dimensionality.) Each node S_i has a communication range $r(S_i)$ which we assume is the same ($r > 0$) for every node. Two nodes can communicate with each other if no signal blocking entity exists between them and their geographic distance is less than their communication range. Two nodes are said to be "reachable" from each other if there exists a path of communication between them. We assume the existence of $k < n$ beacon nodes $\{S_i\}$ ($i = 1 \rightarrow k$) that know their own location and are reachable from each other. We need to devise a distributed algorithm each remaining node S_j ($j = k + 1 \rightarrow N$) can use to estimate its location.

Many existing localization techniques require that each node be within the communication range of some (or all) beacon nodes (e.g., [17], [19]). We relax these strict requirements by only assuming that each node is able to reach to a beacon node. Therefore, our proposed technique, LSVM, is applicable to more types of sensor networks.

B. SVM model

Let $(x(S_i), y(S_i))$ denote the true (to be found) coordinates of node S_i's location, and $h(S_i, S_j)$ the hop-count length of the shortest path between nodes S_i and S_j. Each node S_i is represented by a vector $s_i = <h(S_i, S_1), h(S_i, S_2), ..., h(S_i, S_k)>$. The training data for SVM is the set of beacons $\{S_i\}$ ($i = 1 \rightarrow k$). We define the kernel function as a Radial Basis Function because of its empirical effectiveness [21]:

$$K(S_i, S_j) = e^{-\gamma \|s_i - s_j\|_2^2} \quad (5)$$

where $\| \ . \ \|_2$ the l_2 norm, and $\gamma > 0$ a constant to be computed during the cross-validation phase of the training process.

We consider the following geographic regions:

- M vertical regions, called x-regions, $\{r_0^x, r_1^x, .., r_{M-1}^x\}$ where each region r_i^x is the rectangle containing all points with x-coordinate $x \in [iD/M, (i+1)D/M]$

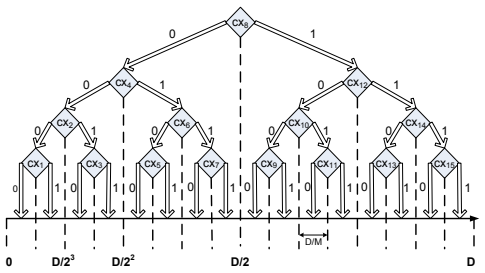


Fig. 2. Decision tree for the x dimension: label 0 coming out of a classification node cx_i means "not belong to cx_i", while label 1 means otherwise. In this example, $m = 4$.

- M horizontal regions, called y-regions, $\{r_0^y, r_1^y, .., r_{M-1}^y\}$ where each region r_i^y is the rectangle containing all points with y-coordinate $y \in [iD/M, (i+1)D/M]$

Our goal is to predict which x-region r_i^x and y-region r_i^y that each sensor S belongs to. We then simply use the center point of the cell $[iD/M, (i+1)D/M] \times [jD/M, (j+1)D/M]$ as the estimated position of node S. The parameter M (or m) controls how close we want to localize a sensor. If the above prediction is indeed correct, the location error for node S is at most $D/(M\sqrt{2})$. However, every SVM is subject to some classification error, and so we should maximize the probability that S is classified into its true cell, and, in case of misclassification, minimize the localization error.

C. Classification Strategies

A component of SVM is to define the target classes for classification. We propose two strategies for this purpose: the multi-class strategy (MCS) and the decision-tree strategy (DTS). They are presented in the following subsections.

1) Multi-class strategy (MTS): We define the following $2M$ classes:

- M classes for the x dimension $\{cx_0, cx_1, ..., cx_{M-1}\}$: Each class cx_i contains nodes with the x-coordinate $\in [iD/M, (i+1)D/M]$.
- M classes for the y dimension $\{cy_0, cy_1, ..., cy_{M-1}\}$: Each class cy_i contains nodes with the y-coordinate $\in [iD/M, (i+1)D/M]$.

Obviously, each class represents a geographic region r_i^x (or r_i^y) defined earlier. Consider a sensor node S. We formulate the problem of estimating its x-coordinate as a multi-class SVM classification, where the number of classes is M. Multi-class SVM is a generalized version of the traditional SVM, or binary SVM, where the number of classes is 2. Many algorithms have been proposed for multi-class SVM. The most popular methods are (see overview in [22]): the "one vs. one" algorithm, the "one vs. all" algorithm, and the "DAGSVM" algorithm. We can use any of these algorithms. Supposing that node S is classed into the x-class cx_i and the y-class cy_i, its estimate location is $[(i+1/2)D/M, (j+1/2)D/M]$.

2) Decision-tree strategy (DTS): The multi-class strategy typically involves $M(M-1)/2$ binary SVM classifications for each dimension. The decision-tree strategy proposed below uses only $M-1$ binary classifications in the training phase and $log_2 M$ binary classifications in the decision phase. We define the following $2M-2$ classes (where $M = 2^m$):

- $M-1$ classes for the x dimension $\{cx_1, cx_2, ..., cx_{M-1}\}$: Each class cx_i contains nodes with the x-coordinate greater or equal to iD/M.
- $M-1$ classes for the y dimension $\{cy_1, cy_2, ..., cy_{M-1}\}$: Each class cy_i contains nodes with the y-coordinate greater or equal to iD/M.

Firstly, we train on these $2M-2$ classes separately based on binary SVM classification. Then, we organize the classifiers for each dimension into a binary decision tree. Let us focus on the x-dimension, whose decision tree is illustrated in Figure 2. Each tree node is an x-class and the two outgoing links represent the outcomes (0: "not belong", 1: "belong") of classification on this class. The classes are assigned to the tree nodes such that if we traverse the tree in the order {leftchild $\rightarrow$ parrent $\rightarrow$ rightchild}, the result is the ordered list $cx_1 \rightarrow cx_2 \rightarrow ... \rightarrow cx_M$. Given this decision tree, each sensor S can estimate its x-coordinate using the following algorithm:

Algorithm 3.1 (X-dimension Localization): Estimate the x-coordinate of sensor S:

1) $i = M/2$ (start at root of the tree $cx_{M/2}$)
2) IF (SVM predicts S not in class cx_i)
   a) IF (cx_i is a leaf node) RETURN $x'(S) = (i-1/2)D/M$
   b) ELSE Move to leftchild cx_j and set $i = j$
3) ELSE
   a) IF (cx_i is a leaf node) RETURN $x'(S) = (i+1/2)D/M$
   b) ELSE Move to rightchild cx_t and set $i = t$
4) GOTO Step 2)

Similarly, a decision tree is built for the y-dimension classes and each sensor S estimates its y-coordinate $y'(S)$ based on the *y-dimension localization algorithm* (like Algorithm 3.1). The estimated location for node S, consequently, is $(x'(S), y'(S))$. Using these algorithms, localization of a node requires visiting $log_2 M$ classifiers of each decision tree, after each visit the geographic range that contains node S downsizing by a half. Table I provides a comparison between the multi-class and decision tree strategies in terms of the number of binary classifications involved in the training phase, testing phase, and the amount of SVM model information (counted for both x- and y-dimensions).

D. Network protocols

We recall that the information that a node S needs to localize itself is consisted of the following (according to Formula 4):

- The support vectors $\{S_i{:}(i, y_i, \alpha_i^*)\}$ and b^* for each class
- The hop-count distance from each beacon node to S, so that the kernel function (see Formula 5) can be computed

TABLE I

MULTI-CLASS STRATEGY VS. DECISION-TREE STRATEGY: TRAINING TIME, TESTING TIME, AND AMOUNT OF SVM MODEL INFORMATION

Strategy	Training	Testing	#SVM models
MCS/One vs. one	$M(M-1)$	$M(M-1)$	$M(M-1)$
MCS/One vs. all	$2M$	$2M$	$2M$
MCS/DAGSVM	$M(M-1)$	$2log_2 \frac{M(M+1)}{2}$	$M(M-1)$
DTS	$2M-2$	$2log_2 M$	$2M-2$

TABLE II

NETWORK CONNECTIVITY SUMMARY

Radius	Min degree	Max degree	Avg degree	Netw. Diameter
r=7m	2	27	14	25 (hops)
r=10m	8	48	28	16 (hops)

Who computes those values and how are they communicated to node S? We divide the entire process into 3 phases: training phase, advertisement phase, and localization phase.

1) Training Phase: We assume that a beacon is selected as the head beacon. The head beacon will later run the SVM training algorithm and therefore should be the most resourceful node. This is a feasible assumption because the head node can be a base station or sink node of the sensor network. The training phase is conducted among the beacon nodes. Firstly, each beacon node sends a HELLO message to every other beacon node. After this round, a beacon knows its hop-count distance from each other beacon node. Next, each beacon node sends an INFO message to the head beacon, containing the location of the sending node and its pairwise hop-count distances from the other beacon nodes. After this round, the head beacon knows the location of every beacon and hop-count distance between every two beacons. The head beacon then runs the SVM training procedure using either the multi-class strategy or the decision-tree strategy. As a result, we obtain the *SVM model information* that will be disseminated to every node in the network.

2) Advertisement Phase: The head beacon broadcasts the SVM model information to all the sensors in the network. Therefore, each node S possesses all the information needed to compute $h_K(S)$ in Formula 4, except for the hop-count distance $h(S, S_i)$ to each beacon node S_i. For this purpose, each beacon node, except for the head beacon, broadcasts a HELLO message to the entire network, so that upon receipt of this HELLO message, each node can obtain the hop-count distance to the beacon.

3) Localization Phase: After receiving the SVM model information from the advertisement phase, each non-beacon node follows the x-dimension localization and y-dimension localization algorithms (see Algorithm 3.1) to estimate its location ($x'(S)$, $y'(S)$).

IV. SIMULATION STUDY

We conducted a simulation study to assess the location accuracy of our approach. A network of 1000 sensors located in a 100m $\times$ 100m 2-D area was simulated, where uniform random distribution is used for generating sensor locations and the beacon sensors. We compared the multi-class strategy (MCS) and decision-tree strategy (DTS) together under the effects of beacon population, network density, and the border problem. We considered two levels of network density, 7m and 10m communication ranges, summarized in Table II, and three different beacon populations: 5% of the network size ($k = 50$ beacons), 15% ($k = 150$ beacons), and 25% ($k = 250$ beacons). We used the $libsvm$ [21] software for SVM classification. For the multi-class strategy, we employ the one vs. one method as recommended by [21], [22]. The γ parameter in Equation 5 and C parameter in Inequality 3 were automatically determined by the mechanisms of $libsvm$. We set m = 7 (i.e., M = 128) throughout the study.

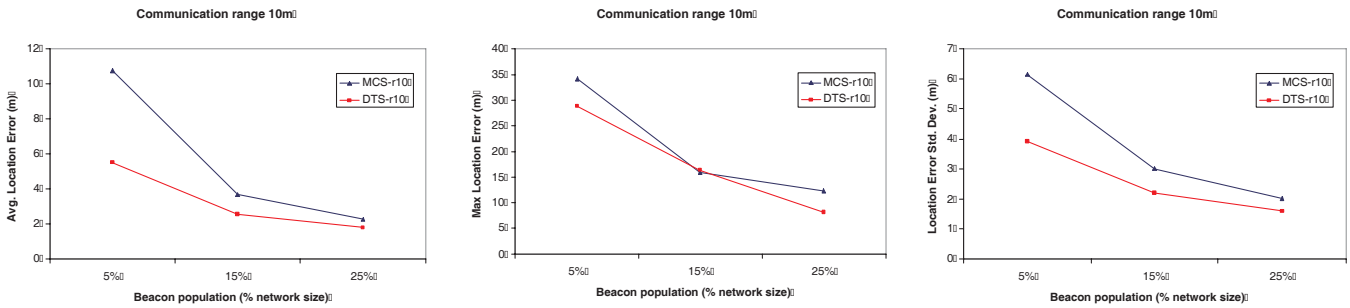
Figures 3 and 4 plot the average, max, and standard deviation of location errors for the range-10m network and range-7m network. It is observed that DTS always provides better results than MCS. When only 5% of the network serve as beacons, DTS's localization error is 6m on average while that of MCS is 11m (almost double). The main reason is probably because DTS uses a lot fewer binary SVM classifications than MCS, thus resulting in smaller accumulative error after the entire prediction process ends. The gap between these two strategies is narrowed down as more nodes serve as beacons.

Regarding the beacon population size, it is understandable that when this population is larger, the localization gets more accurate. Both strategies provide good results when 25% of the network serve as beacons, in which case the error is about 2-3m and no more than 15m. The standard deviation is also small (about 2m), which implies that most sensors' locations are very well estimated.

Often occurring in many existing localization techniques is the border problem, in which border sensors are usually positioned with larger error than sensors inside the field. Figure 5 plots the location errors of all sensors in the network sorted in the order of sensors close to the center of the field first and sensors near the edge last. If the border problem exists, we should see an up-hill curve pattern from the left to the right side of the graph. However, this pattern is rarely seen in this figure; the majority of errors have similar values, implying that the border problem is well-addressed in our proposed approach. This desirable property is understandable. Because the location of each sensor is estimated only based on distances to the beacons and independently of other sensors, whether a sensor is near the border of lies inside should not have a big impact on the location error.

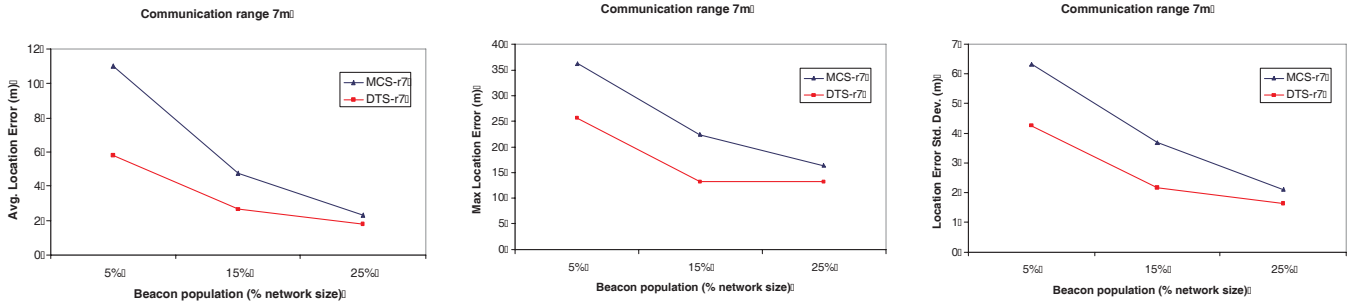
V. CONCLUSION

We have presented an approach based on the concept of SVM to localize nodes in a large-scale sensor networks. In our SVM model, the beacon nodes serve as the training points and the kernel function uses only mere connectivity information. Therefore, the proposed approach can be used for networks that do not require expensive ranging and specialized (and/or mobile) devices. It can be realized using two alternative strategies: the multi-class strategy (MCS) and the decision-tree strategy (DTS). Our simulation study have shown



(a) Average location error (b) Max location error (c) Standard deviation

Fig. 3. MCS vs. DTS (communication range 10m): Statistics on location error of each technique under different beacon populations.



(a) Average location error (b) Max location error (c) Standard deviation

Fig. 4. MCS vs. DTS (communication range 7m): Statistics on location error of each technique under different beacon populations.

that both MCS and DTS provide good localization accuracy and alleviate the border problem effectively. We, however, recommend DTS because of his superior performance over MCS. Our future research includes a comparison between DTS with other existing localization techniques as well as its performance in networks with coverage holes. We also plan a prototype implementation of DTS.

REFERENCES

[1] L. Doherty, L. E. Ghaoui, and K. S. J. Pister, "Convex position estimation in wireless sensor networks," in *IEEE Infocom*, April 2001.

[2] Shang, Juml, Zhang, and Fromherz, "Localization from mere connectivity," in *ACM Mobihoc*, 2003.

[3] C. Savarese, J. Rabaey, and J. Beutel, "Locationing in distributed ad-hoc wireless sensor networks," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Salt Lake city, UT, 2001.

[4] N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller, "Anchor-free distributed localization in sensor networks," in *ACM Sensys*, 2003.

[5] S. Capkun, M. Hamdi, and J.-P. Hubauz, "Gps-free positioning in mobile ad hoc networks," in *Hawai International Conference on System Sciences*, 2001.

[6] L. Meertens and S. Fitzpatrick, "The distributed construction of a global coordinate system in a network of static computational nodes from inter-node didstances," Kestrel Institute, Tech. Rep., 2004.

[7] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust distributed network localization with noisy range measurements," in *ACM Sensys*, Baltimore, MA, November 2004.

[8] D. Niculescu and B. Nath, "Ad hoc positioning system (aps)," in *IEEE Globecom*, 2001.

[9] N. Bulusu, V. Bychkovskiy, D. Estrin, and J. Heidemann, "Scalable ad hoc deployable rf-based localization," in *Grace Hopper Celebration of Women in Computing Conference*, Vancouver, Canada, October 2002.

[10] A. Savvides, H. Park, and M. Srivastava, "The bits and flops of the n-hop multilateration primitive for node localization problems," in *Workshop on Wireless Networks and Applications (in conjunction with Mobicom 2002)*, Atlanta, GA, September 2002.

[11] A. Savvides, C.-C. Han, and M. B. Strivastava, "Dynamic fine-grained localization in ad hoc networks of sensors," in *ACM International Conference on Mobile Computing and Networking (Mobicom)*, Rome, Italy, July 2001, pp. 166–179.

[12] S. Simic and S. S. Sastry, "Distributed localization in wireless ad hoc networks," University of California at Berkeley, Tech. Rep., 2002.

[13] C. Whitehouse, "The design of calamari: an ad hoc localization system for sensor networks," Master's thesis, University of California at Berkeley, 2002.

[14] D. Niculescu and B. Nath, "Ad hoc positioning system (aps) using aoa," in *IEEE Infocom*, 2003.

[15] R. Nagpal, H. Shrobe, and J. Bachrach, "Organizing a global coordinate system from local information on an ad hoc sensor network," in *International Symposium on Information Processing in Sensor Networks*, 2003.

[16] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher, "Range-free localization schemes in large scale sensor networks," in *ACM Conference on Mobile Computing and Networking*, 2003.

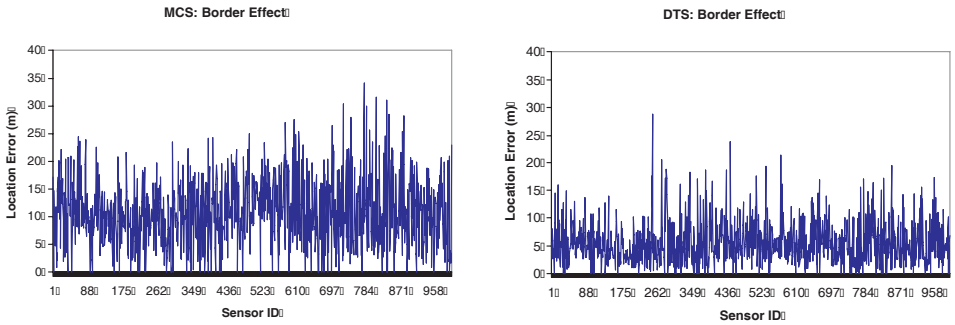
[17] R. Stoleru, J. A. Stankovic, and D. Luebke, "A high-accuracy, low-cost localization system for wireless sensor networks," in *ACM Sensys*, San Diego, CA, November 2005.

[18] N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller, "Mobile-Assisted Localization in Wireless Sensor Networks," in *IEEE INFOCOM*, Miami, FL, March 2005.

[19] X. Nguyen, M. Jordan, and B. Sinopoli, "A kernel-based learning approach to ad hoc sensor network localization," *IEEE Transactions on Sensor Networks*, vol. 1, pp. 134–152, 2005.

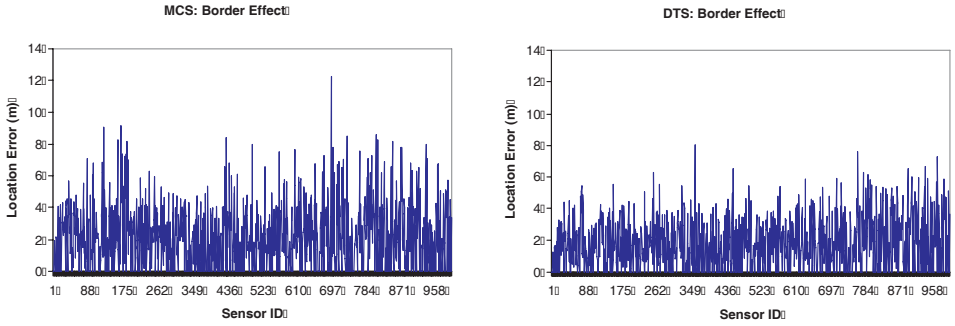
[20] V. N. Vapnik, *Statistical Learning Theory*. Wiley-Interscience, 1998.

[21] C.-C. Chang and C.-J. Lin, *LIBSVM – A library for Support Vector Machines*, National Taiwan University. [Online]. Available: http://www.csie.ntu.edu.tw/ cjlin/libsvm



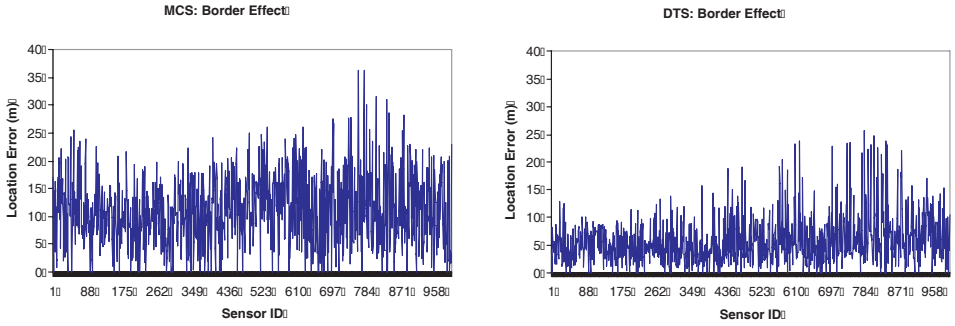
(a) MCS: 50 beacons, r = 10m

(b) DTS: 50 beacons, r = 10m



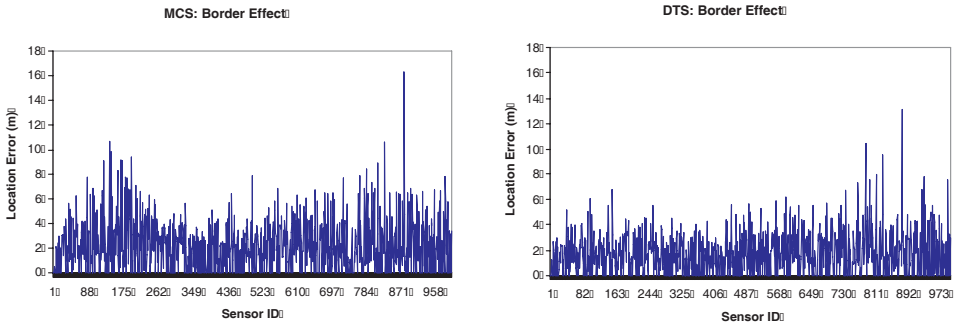
(c) MCS: 250 beacons, r = 10m

(d) DTS: 250 beacons, r = 10m



(e) MCS: 50 beacons, r = 7m

(f) DTS: 50 beacons, r = 7m



(g) MCS: 250 beacons, r = 7m

(h) DTS: 250 beacons, r = 7m

Fig. 5. The border problem: Location error for every sensor, sorted in the order of nodes near the center of the field first and nodes near the edge last.

[22] C.-W. Hsu and C.-J. Lin, "A comparison of methods for multi-class support vector machines," *IEEE Transactions on Neural Networks*, vol. 13, pp. 415–425, 2002.