Localization In Wireless Sensor Networks based on Support Vector Machines

Duc A. Tran, *Member, IEEE,* and Thinh Nguyen, *Member, IEEE*

***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. We propose LSVM – a novel solution with the following merits. First, LSVM localizes the network based on mere connectivity information (i.e., hop counts only), and, therefore, is simple and does not require specialized ranging hardware or assisting mobile devices as in most existing techniques. Second, LSVM is based on Support Vector Machine (SVM) learning. Although SVM is a classification method, we show its applicability to the localization problem and prove that the localization error can be upper-bounded by any small threshold given an appropriate training data size. Third, LSVM addresses the border and coverage-hole problems effectively. Last but not least, LSVM offers fast localization in a distributed manner with efficient use of processing and communication resources. We also propose a modified version of mass-spring optimization to further improve the location estimation in LSVM. The promising performance of LSVM is exhibited by our simulation study.**

***Index Terms*— Sensor networks, position estimation, sensor localization, SVM, mass spring optimization**

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.

A straightforward localization approach is to gather 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 using mathematical algorithms such as Semidefinite Programming [1] and Multidimensional Scaling [2]. Despite its excellent approximation, this centralized approach is impractical for large-scale sensor networks due to high computation and communication costs.

Many techniques have been proposed that attempt localization in a distributed manner. The relaxation-based techniques [3], [4] start with all the nodes in initial positions and keep refining their positions using algorithms such as local neighborhood multilateration and convex optimization. The coordinate-system stitching techniques [5]–[8] divide the network into overlapping regions, nodes in each region being positioned relatively to the region's local coordinate system (a centralized algorithm may be used here). The local coordinate systems are then merged, or "stitched", together to form a global coordinate system. Localization accuracy can be improved by using beacon-based techniques ([8]–[16]) that take advantage of nodes with known location, called beacons, and extrapolate unknown node locations from the beacon locations.

Most current techniques assume that the distance between two neighbor nodes can be measured, typically via a ranging process. For instance, pair-wise distance can be estimated based on Received Signal Strength Indication (RSSI) [13], Time Difference of Arrival (TDoA) [17], [18], or Angle of Arrival (AoA) [14], [19]. The problem with distance measurement is that the ranging process (or hardware) is subject to noise and its complexity/cost increase with accuracy requirement. For a large sensor network with low-end sensors, it is often not affordable to equip them all with ranging capability.

In this paper, we solve the localization problem with the following modest requirements: (R1) beacon nodes exist, (R2) a sensor may not hear directly from any beacon node, and (R3) only connectivity information may be used for location estimation (pairwise distance measurement not required). Requirement (R1) is for improved localization accuracy. Requirement (R2) relaxes the strong requirement on communication range of beacon nodes. Requirement (R3) avoids the expensive ranging process. All these requirements are reasonable for large networks where sensor nodes are of little resources.

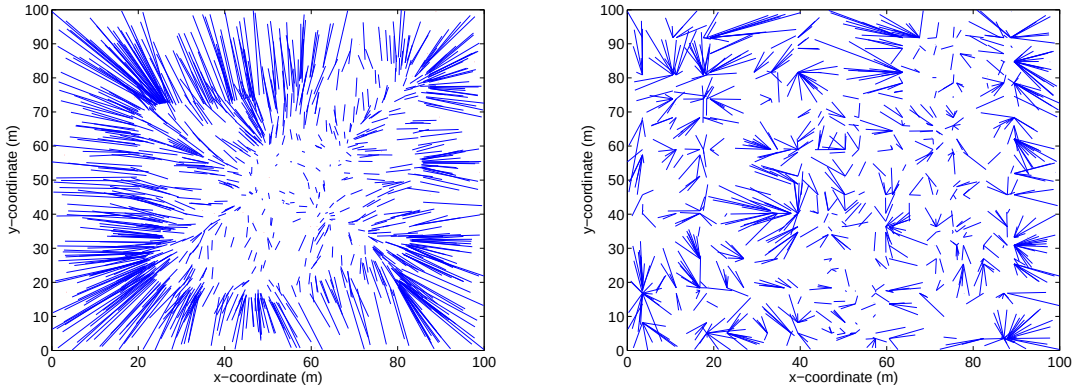
Few range-free techniques have been proposed [6], [9], [16], [20], [21]. APIT [16] assumes that a node can hear from a large number of beacons, and thus does not satisfy requirement (R2). Spotlight [20] offers good results, but requires an aerial vehicle to generate light onto the sensor field. [21] uses a mobile node to assist pair-wise distance measurements until converged to a "global rigid" state where the sensor locations can be uniquely determined. [20], [21] do not satisfy requirement (R3).

A popular approach that shares the same requirements {R1, R2, R3} with our work is Diffusion [6], [9], where each node is repeatedly positioned as the centroid of its neighbors until convergence. Figure 1(a) illustrates two main problems of this approach: the convergence problem (i.e., many averaging loops result in long localization time and significant bandwidth

This work is supported in part by the National Science Foundation under Grant No. CNS-0615055.

D. A. Tran (duc.tran@udayton.edu) is with the Department of Computer Science, University of Dayton, OH 45469.

T. Nguyen (thinhq@eecs.oregonstate.edu) is with the School of Electrical Engineering and Computer Science, Oregon State University, OR 97331.



(a) Diffusion: The border problem remains after 10,000 averaging loops

(b) LSVM: The border problem disappears

Fig. 1. 1000 sensors on a 100m x 100m field, with 50 random beacons. A line connects the true and estimated positions of each sensor node

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.

We propose LSVM – a novel solution that satisfies the requirements above, offers fast localization, and alleviates the border problem significantly (illustrated in Figure 1(b)). LSVM is also effective in networks with the existence of coverage holes or obstacles. LSVM localizes the network using the learning concept 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. To the localization problem, 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.

The latest (perhaps, only) work we are aware of, that explores the applicability of SVM to the localization problem is [22]. This technique, however, assumes that every node can measure direct signal strength from *all* the beacons, which contradicts requirement (R2). In the case of a wide network, a node may only receive signals from a small subset of beacons, and so this technique would be significantly less accurate.

Our work is more suitable for networks of larger scale because it is based on connectivity rather than direct signal strength. Our contribution includes definitions of the kernel function and classes to categorize the sensor nodes, a strategy to apply the classifiers, and a theoretical bound analysis on the localization error. We also propose a mass-spring optimization based procedure to further improve the location accuracy.

The remainder of this paper is structured as follows. We provide a brief background on SVM in the next section. We describe the details of LSVM and analyze its localization error in Section III. Evaluation results based on a simulation study 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) [23] 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 [23], there exists a feature space $\vec{X}$ where the kernel K defined above is 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 will discuss this error in the error analysis of our proposed localization technique (LSVM). 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] × [0, D] ($D > 0$)[1]. 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 *one-hop* communication range of some (or all) beacon nodes (e.g., [20], [22]). Our assumption is more flexible because we only assume that each node can communicate to a beacon node by a *multi-hop* path. 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 [24]:

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

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

We considers 2 sets of $M-1 = 2^m - 1$ classes to classify non-beacon nodes:

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

Intuitively, each x-class cx_i contains nodes that lie to the right of the vertical line $x = iD/M$, while y-class cx_i contains nodes that lie above the horizontal line $y = iD/M$. Therefore, if the SVM learning predicts that a node S is in class cx_i but not class cx_{i+1}, and in class cy_j but not class cy_{j+1}, we conclude that S is inside the square cell $[iD/M, (i + 1)D/M] \times [jD/M, (j + 1)D/M]$. We then simply use the cell's center point as the estimated position of node S. If the above prediction is indeed correct, the localization 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.

[1]We assume 2 dimensions for simplicity, even though LSVM can work with any dimensionality.

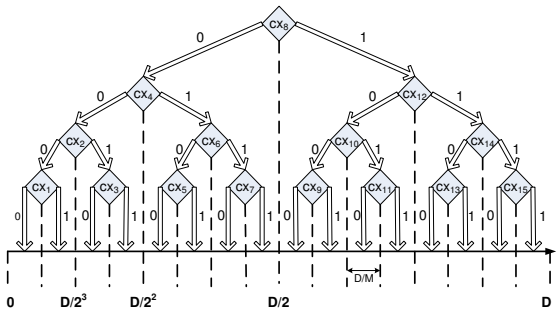


Fig. 2. Decision tree: $m = 4$.

C. Algorithms and Protocols

Let us focus on the classification along the x-dimension. We organize the x-classes into a binary decision tree, 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 → parrent → rightchild}, the result is the ordered list $cx_1 \rightarrow cx_2 \rightarrow ... \rightarrow cx_{M-1}$. 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 $log_2 M$ nodes of each decision tree, after each visit the geographic range that contains node S downsizing by a half. The parameter M (or m) controls how close we want to localize a sensor.

According to Formula 4, the information that a node S uses to localize itself is consisted of the following (called the *SVM model information*):

- 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

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, where message exchanges use the underlying unicast routing protocol. 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. Each beacon then 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. Therefore, the head beacon knows the location of every beacon and hop-count distance between every two beacons. Next, the head beacon runs the SVM training procedure on all $2M-2$ classes cx_1, cx_2, ..., cx_{M-1}, cy_1, cy_2, ..., cy_{M-1} and, for each class, computes the corresponding b^* and the information (i, y_i, α_i^*) for each support vector S_i.

The communication cost is due to the unicast delivery of k^2-1 HELLO and INFO messages. The computation cost is due to the SVM training procedure at the head beacon. SVM is known to be computationally efficient, and has the the worst-case runtime $O(k_{sv}^2 k)$, where k_{sv} is the number of support vectors (usually much smaller than k). Since we apply SVM to $2M-2$ classes, the total runtime is $O(M(k_{sv}^2 k))$.

2) Advertisement Phase: In this phase, the head beacon advertises the SVM model information by broadcasting it 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. The number of messages forwarded in the advertisement phase is kN. The amount of traffic generated also depends on the size of the SVM model information, which in turn depends on the number of support vectors for each class.

3) Localization Phase: Each non-beacon node starts this phase after receiving the SVM model information from the advertisement phase. It then follows the x-dimension localization and y-dimension localization algorithms (see Algorithm 3.1) to estimate its location $(x'(S), y'(S))$. These algorithms each require computation of $h_K(S)$ in Formula 4 for $log_2(M)$ classes. The runtime should therefore be short.

D. Error Analysis

SVM is subject to error and so is LSVM. A misclassification with respect to a class C occurs when SVM predicts that a sensor is in C but in fact it is not or predicts that the sensor is not in C but it actually is. In this section, we formulate the LSVM error under the effect of the SVM error.

Consider a sensor S and localization along the x-dimension. Without loss of generality, suppose that $x(S) \geq D/2$. Let $x = x_1 x_2 ... x_m$ be the path on the decision tree that leads to the correct interval containing S, and $x' = x'_1 x'_2 ... x'_m$ the decision path taken under the x-dimension localization algorithm. Since we estimate the x-coordinate of a sensor as the middle position in the estimated interval, the x-dimension location error is at most $e_X(x) = \mid x' - x \mid \times D/M + D/(2M) = D/M\ (1/2 + \mid x' - x \mid)$.

Figure 3(a) illustrates a case where the correct path $x = 1100$ and the decision path $x' = 0011$ for are sensor S with true $x(S) \in (12D/16, 13D/16]$. There are 2 misclassifications in the decision path: the decision $S \notin cx_8$ (i.e., $x(S) < D/2$) and the decision $S \notin cx_4$ (i.e., $x(S) < D/4$). These decisions are wrong because $x(S) > 12D/16 > D/2 > D/4$. Decisions $S \in cx_2$ (i.e., $x(S) \geq D/8$) and $S \in cx_3$ (i.e., $x(S) \geq 3D/16$) are correct.

In general, let $i > 0$ be the number of misclassifications and $p(i)$ the probability of their occurrence. Let ϵ be the worst error probability of SVM classification over all classes cx_1, cx_2, ..., cx_{M-1}, cy_1, cy_2, ..., cy_{M-1}. Since there are m independent classification steps in the localization algorithm, $p(i) = C_m^i \epsilon^i (1-\epsilon)^{m-i}$. We will analyze the worst case where $e_i(x) = |x' - x|$ is maximized. There are two cases:

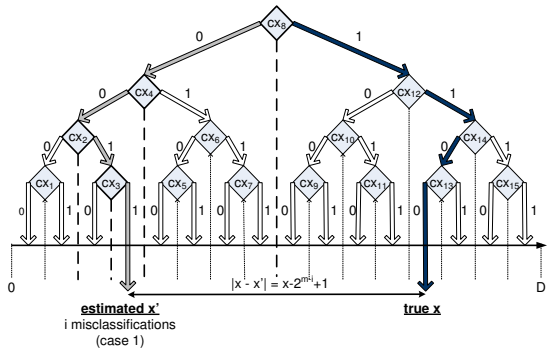
1) If $x(S) \in [3D/4, D]$ (i.e., $x_1 = x_2 = 1$): The worst case occurs when all the first i classifications give wrong results and the remaining classifications correct. That is, $x' = 0^i 1^{m-i}$, and therefore $e_i(x) = x - 0^i 1^{m-i} = x - 2^{m-i} + 1$. (See Figure 3(a).)
2) If $x(S) \in [D/2, 3D/4)$ (i.e., $x_1 = 1, x_2 = 0$): The worst case must be one of the following scenarios: (see Figure 3(b))
   a) All the first i classifications give wrong results and the remaining classifications correct: $x' = 0^i 1^{m-i} = 2^{m-i} - 1$ (e.g., the decision path 0011 in Figure 3(b).)
   b) The first classification is correct, the next i classifications wrong, and the remaining classifications correct: $x' = 1^{i+1} 0^{m-i-1} = (2^{i+1} - 1) 2^{m-i-1} = 2^m - 2^{m-i-1}$ (e.g., the decision path 1110 in Figure 3(b).)

   Therefore, $e_i(x) = \max(x - 2^{m-i} + 1,\ 2^m - 2^{m-i-1} - x)$.

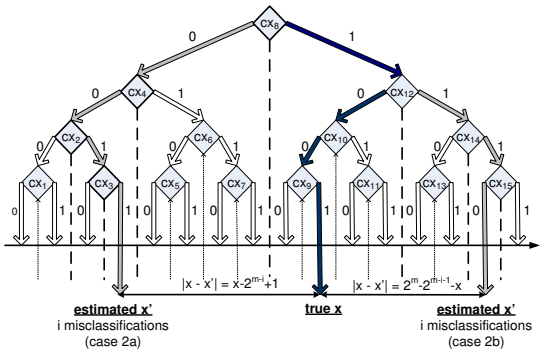
Consequently, the x-dimension location error expected for any node is bounded by

$$E_X^f = \sum_{x=M/2}^{M-1} e_X(x) f(x) \tag{6}$$

$$= \frac{D}{M} \left(\frac{1}{2} + \sum_{x=M/2}^{M-1} \sum_{i=1}^{m} p(i) e_i(x) f(x) \right) \tag{7}$$



(a) $3D/4 \leq x(S) \leq D$: For correct path 1100, the worst-case decision path is 0011



(b) $D/2 \leq x(S) < 3D/4$: For correct path 1001, the worst-case decision path is 0011 (case 2a) or 1110 (case 2b). In this example, the worst-case decision path is 0011

Fig. 3. Worst-case possibilities for classification along the x-dimension (m = 4): Assume that there are $i = 2$ misclassifications

where $f(x)$ is the probability that a sensor node S has x as the correct path given the fact that $x(S) \geq D/2$. For a uniformly distributed sensor field, $f(x) = 1/(M/2) = 1/2^{m-1}$, and therefore E_X^f becomes

$$E_X^u = \frac{D}{M}\left(\frac{1}{2} + \sum_{x=M/2}^{M-1}\sum_{i=1}^{m} p(i)e_i(x)/2^{m-1}\right) \quad (8)$$

The following lemma provides a closed form for this error expectation bound.

Lemma 3.1 (X-dimension Localization Error): The following equality is valid:

$$E_X^u = D\left(\frac{1}{2^m} + \frac{7}{8} - \frac{(1-\epsilon)^m}{2^{m+1}} - \frac{(2-\epsilon)^m}{2^m} + \frac{(4-3\epsilon)^m}{2^{2m+3}}\right) \quad (9)$$

Proof: Consider a sensor S. For the case $x(S) \in [3D/4, D]$, the error expectation is $\frac{D}{M}(\frac{1}{2}+E_1)$, where

$$E_1 = \frac{1}{2^{m-2}} \sum_{x=2^{m-1}+2^{m-2}}^{2^m-1} \underbrace{\left(\sum_{i=1}^{m} p(i)e_i(x)\right)}_{A} \quad (10)$$

The inner summation A is computed as

$$\begin{aligned} A &= \sum_{i=1}^{m} C_m^i \epsilon^i (1-\epsilon)^{m-i}(x - 2^{m-i} + 1) \\ &= (x+1)(1-(1-\epsilon)^m) - (2-\epsilon)^m + (2-2\epsilon)^m \end{aligned}$$

Therefore, it is easily derivable that

$$E_1 = (\frac{1}{2} + 9\times 2^{m-3})(1-(1-\epsilon)^m) + (2-2\epsilon)^m - (2-\epsilon)^m \quad (11)$$

If $x(S) \in [D/2, 3D/4)$, we can rewrite the error expectation for this case as $\frac{D}{M}(\frac{1}{2}+E_2)$, where

$$E_2 = \sum_{i=1}^{m} \frac{p(i)}{2^{m-2}} \underbrace{\left(\sum_{x=2^{m-1}}^{2^{m-1}+2^{m-2}-1} e_i(x)\right)}_{B} \quad (12)$$

The inner summation B is computed as

$$\begin{aligned} B &= \sum_{x=2^{m-1}}^{2^{m-1}+2^{m-2}-1} \max(x - 2^{m-i} + 1, 2^m - 2^{m-i-1} - x) \\ &= \sum_{x=2^{m-1}}^{2^{m-1}+2^{m-i-2}-1} (2^m - 2^{m-i-1} - x) \\ &\quad + \sum_{x=2^{m-1}+2^{m-i-2}}^{2^{m-1}+2^{m-2}-1} (x - 2^{m-i} + 1) \\ &= 2^{m-2}\left(\frac{1}{2} + 2^m(\frac{5}{8} + \frac{1}{4^{i+1}} - \frac{1}{2^i})\right) \end{aligned}$$

Therefore,

$$\begin{aligned} E_2 &= \sum_{i=1}^{m} p(i)\left(\frac{1}{2} + 2^m(\frac{5}{8} + \frac{1}{4^{i+1}} - \frac{1}{2^i})\right) \\ &= \sum_{i=1}^{m} C_m^i \epsilon^i (1-\epsilon)^{m-i}\left(\frac{1}{2} + 2^m(\frac{5}{8} + \frac{1}{4^{i+1}} - \frac{1}{2^i})\right) \\ &= \frac{1}{2} + 5\times 2^{m-3} + (2^{m-3} - \frac{1}{2})(1-\epsilon)^m \\ &\quad + 2^{m-2}(1-3\epsilon/4)^m - 2^m(1-\epsilon/2)^m \end{aligned}$$

Consequently, for an arbitrary sensor S where $x(S)$ can be anywhere in $[D/2, D]$ with the same probability, the location error expectation is bounded by $E_X^u = \frac{D}{M}\left(\frac{1}{2} + \frac{E_1+E_2}{2}\right)$ which is equal to

$$D\left(\frac{1}{2^m} + \frac{7}{8} - \frac{(1-\epsilon)^m}{2^{m+1}} - \frac{(2-\epsilon)^m}{2^m} + \frac{(4-3\epsilon)^m}{2^{2m+3}}\right) \quad (13)$$

The lemma is proved. ■

Since the case $x(S) \in (0, D/2)$ is symmetric to the case $x(S) \in$ [D/2, D), for a uniformly distributed sensor field, we obtain the same E_X^u shown in Lemma 3.1 for every sensor node. Similarly, $E_Y^u = E_X^u$ for the y-dimension. Therefore, the total location error in both dimensions is bounded by $E^u = \sqrt{2}E_X^u = \sqrt{2}E_Y^u$. The theorem below is trivial to prove.

Theorem 3.2 (Localization Error): For a uniformly distributed sensor field, LSVM's location error expected for any

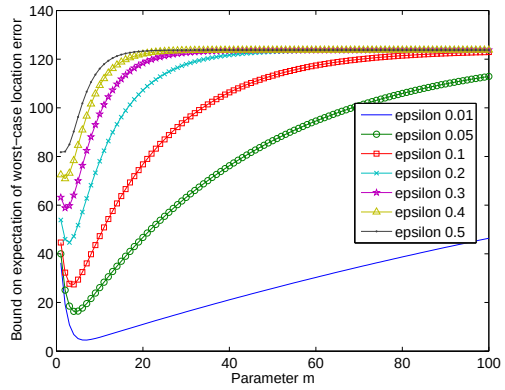


Fig. 4. Bound on the expectation of worse-case location error under various values of SVM classification error ϵ. A lower value of ϵ corresponds to a lower-appearing curve. Note the value m that minimizes the bound.

node is bounded by

$$E^u = \sqrt{2}D\left(\frac{1}{2^m} + \frac{7}{8} - \frac{(1-\epsilon)^m}{2^{m+1}} - \frac{(2-\epsilon)^m}{2^m} + \frac{(4-3\epsilon)^m}{2^{2m+3}}\right) \tag{14}$$

The error bound E^u depends on two factors: the worst-case SVM classification error ϵ and the parameter m. Consider the linear function $h_K(x)$ (see Equality 4) found by the SVM algorithm presented in Section II. The following theorem exhibits a nice property of SVM that ϵ of the classification based on $h_K(x)$ is bounded and the bound can approach zero if the training size is large enough.

Theorem 3.3 (SVM Error Bound): There exists a constant c, such that for any probability distribution D on $X \times \{-1, 1\}$ with support in a ball of radius R around the origin, with probability $1-\delta$ the SVM has error

$$\epsilon \leq \frac{c}{k}\left(\frac{R^2 + \| \xi \|_1^2 \, log(1/\Gamma)}{\Gamma^2} log^2 k + log\frac{1}{\delta}\right) \tag{15}$$

where the values of Γ and vector ξ are as follows:

$$\Gamma = \left(\sum_{i,j \in support\ vector} y_i y_j \alpha_i^* \alpha_j^* K(x_i, x_j)\right)^{-1/2}$$
$$\xi = \langle \max(0, \Gamma - y_1 h_K(x_1)), ..., \max(0, \Gamma - y_k h_K(x_k)) \rangle$$

Proof: This theorem is a direct consequence of combining Theorem 4.24 and Proposition 6.12 presented in [25]. Details are omitted due to the space limit. ■

Theorem 3.3 implies that almost sure $\epsilon \to 0$ when $k \to \infty$. Therefore, if the number of beacon sensors k is large, we can have highly accurate SVM classification, and therefore a small localization error since E^u is a monotonically increasing function of ϵ. Having fixed k, we can improve the location error by controlling m. A natural question is what should be the best value for m? Our safe choice is to choose m^* such that the bound E^u in Theorem 3.2 is minimized. Mathematically, it can be shown that such m^* exists and can be estimated (proof and algorithm are skipped due to limited space). Figure 4 illustrates the error bounds corresponding to seven different values of ϵ. Note that each bound has a minimizing point m^* which gets larger as ϵ decreases. It also implies that we should always use a value less than 8 for parameter m. The process of finding m^* is in the Training Phase presented earlier in Section III-C.1. It is detailed as follows:

1) Conduct the Training Phase as usual but with the value $m = m_{max}$ where m_{max} is the best estimated m^* for the case $\epsilon = 0.01$ (based on minimizing Formula 14).
2) Compute the maximum ϵ_{max} of ϵ among all classifications based on the value of m selected in Step (1).
3) Set m to the corresponding m^* of ϵ_{max} (based on minimizing Formula 14). This value of m will be used for the entire process.

E. Improving LSVM: Modified Mass-Spring Optimization

Mass-spring optimization (MSO) has been used successfully by localization techniques to improve their accuracy (e.g., [4]). This optimization is not directly applicable to LSVM because MSO requires that the distance between adjacent nodes be measurable while LSVM does not so assume. In this section, we propose a modified version of MSO so it can work with LSVM. But, first, we briefly present MSO.

1) Mass-Spring Optimization (MSO): Each sensor node is considered a "mass" and repeatedly adjusts its location based a "spring force" computed from the locations of the neighbors. The ultimate goal of MSO is that, after sufficient repetitions, the distance $dist_{est}(S_i, S_j)$ between every pair of adjacent nodes S_i and S_j, computed based on their estimated locations, converges to the their true distance $dist_{true}(S_i, S_j)$. In other words, we need to minimize $E = \sum_{i=1}^{N} E(S_i)$, where

$$E(S_i) = \sum_{neighbor\ S_j} (dist_{est}(S_i, S_j) - dist_{true}(S_i, S_j))^2 \tag{16}$$

E is called the total energy of the system and $E(S_i)$ the local energy of each node S_i.

In a mass-spring system, the force on sensor S_i in the direction toward S_j is defined as

$$\overrightarrow{f(S_i, S_j)} = (dist_{est}(S_i, S_j) - dist_{true}(S_i, S_j)) \times \overrightarrow{u(S_i, S_j)} \tag{17}$$

where $\overrightarrow{u(S_i, S_j)}$ denotes the unit vector from point S_i to point S_j in the 2D space. The resultant force on node S_i pulled by all its neighbors, therefore, is

$$\overrightarrow{F(S_i)} = \sum_{neighbor\ S_j} \overrightarrow{f(S_i, S_j)} \tag{18}$$

To minimize the total energy E in a distributed manner, each sensor node S_i concurrently minimizes its local energy $E(S_i)$. For this purpose, each node S_i translates its position by a vector $\alpha_i \overrightarrow{F(S_i)}$ where $\alpha_i \in (0, 1)$ is chosen such that the new energy $E(S_i)$ is smaller than the old energy. This position-translation process is repeated until the energy cannot be improve significantly further or the number of repetitions exceeds some threshold. An example choice for α_i, as recommended and evaluated by [4], is $\alpha_i = \frac{1}{2m_i}$ where m_i is the number of neighbor nodes of S_i.

TABLE I

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)

2) Modified Mass-Spring Optimization (MMSO): Since LSVM assumes no knowledge on the true distance $dist_{true}(S_i, S_j)$, we use the communication range r of each sensor in lieu of $dist_{true}(S_i, S_j)$. The energy of a sensor S_i is now defined as

$$E(S_i) = \sum_{neighbor\ S_j} (dist_{est}(S_i, S_j) - r)^2 \qquad (19)$$

The force on S_i pulled by S_j is redefined as

$$\overrightarrow{f(S_i, S_j)} = (dist_{est}(S_i, S_j) - r) \times \overrightarrow{u(S_i, S_j)} \qquad (20)$$

Since we do not know the true distance, we only want to fix those adjacent sensors whose estimated distance exceeds r. We therefore redefine the resultant force as follows:

$$\overrightarrow{F(S_i)} = \sum_{neighbor\ S_j\ :\ dist_{est}(S_i,S_j)\ >\ r} \overrightarrow{f(S_i, S_j)} \qquad (21)$$

The MMSO algorithm conducted at node S_i is summarized below (for the case of stopping when the number of translations exceeds some threshold):

Algorithm 3.2 (Modified Mass-Spring Optimization): Improve the location estimation of sensor S_i:

1) Let $threshold_{count}$ be the system-defined maximal number of iterations
2) Let $(x_{new}(S_i), y_{new}(S_i))$ be the location of S_i estimated using the LSVM localization algorithms presented in previous sections (see Algorithm 3.1)
3) Compute m_i the number of neighbors of S_i, the current energy $E(S_i)$ according to Formula 19, and the current force $\overrightarrow{F(S_i)}$ according to Formula 21. Let f_X and f_Y be the x-dimension and y-dimension magnitudes of this force, respectively.
4) Compute possible new location

$$\begin{aligned} x_{new}(S_i) &= x_{current}(S_i) + \frac{f_X}{2m_i} \\ y_{new}(S_i) &= y_{current}(S_i) + \frac{f_Y}{2m_i} \end{aligned}$$

5) Compute possible new energy $E_{new}(S_i)$ according to Formula 19 using the new location $(x_{new}(S_i), y_{new}(S_i))$
6) IF $(E(S_i) > E_{new}(S_i))$, update the current position

$$\begin{aligned} x_{current}(S_i) &= x_{new}(S_i) \\ y_{current}(S_i) &= y_{new}(S_i) \end{aligned}$$

7) IF (number of iterations exceeds $threshold_{count}$) QUIT
8) ELSE GOTO Step 3

TABLE II

SVM CLASSIFICATION ACCURACY (MIN/AVG) PER CLASS FOR DIFFERENT RANGES AND BEACON POPULATIONS

	Axis	5%	10%	15%	20%	25%
Range 10m	x	.89/.95	.92/.96	.95/.98	.95/.98	.96/.98
	y	.90/.96	.91/.97	.93/.98	.95/.98	.95/.98
Range 7m	x	.91/.96	.92/.97	.94/.98	.95/.98	.95/.98
	y	.90/.96	.91/.97	.93/.98	.95/.98	.96/.98

TABLE III

AVERAGE NUMBER OF SUPPORT VECTORS PER CLASS FOR DIFFERENT RANGES AND BEACON POPULATIONS

	Axis	5%	10%	15%	20%	25%
Range 10m	x	14.29	20.48	22.55	33.31	42.62
	y	13.85	23.16	27.08	25.30	29.08
Range 7m	x	12.40	17.57	23.44	34.70	29.55
	y	16.55	24.92	30.96	27.34	26.37

IV. SIMULATION STUDY

We conducted a simulation study on a network of 1000 sensors located in a 100m $\times$ 100m 2-D area. We assumed uniform random distribution for the sensor locations and the selection of the beacon sensors. We considered two levels of network density (7m and 10m communication ranges), summarized in Table I. We also considered five different beacon populations: 5% of the network size ($k = 50$ beacons), 10% ($k = 100$ beacons), 15% ($k = 150$ beacons), 20% ($k = 200$ beacons), and 25% ($k = 250$ beacons). The computational cost of LSVM is theoretically analyzed in Section III-C. The communication cost is mainly due to k broadcasts where k is the number of beacons. We, therefore, focused more on the localization accuracy.

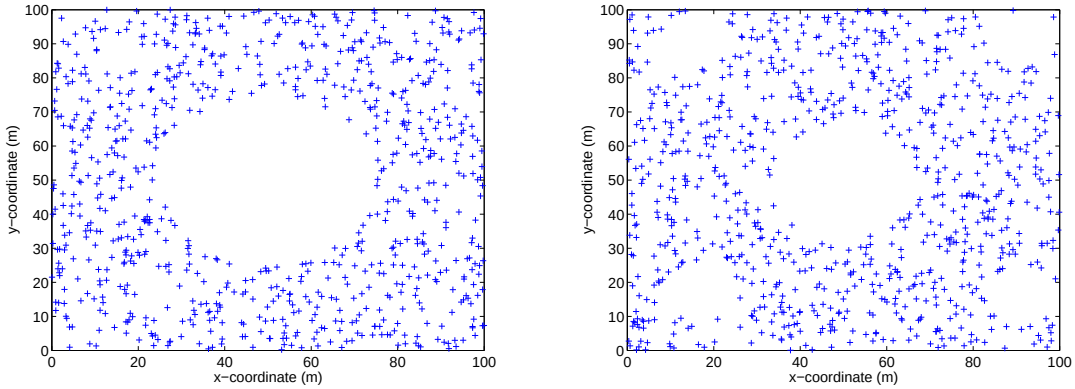
We used the algorithms in the $libsvm$ [24] software for SVM classification. The γ parameter in Equation 5 and $\mathcal{C}$ parameter in Inequality 3 were automatically determined by the mechanisms of $libsvm$. We set m = 7 (i.e., M = 128) by default, the rationale for which will be explained in Section IV-A.

Many networking protocols such as routing and localization suffer from the existence of coverage holes or obstacles in the sensor field. We considered three sensor networks: a network with no coverage hole and two following networks with hole existence: (1) a network with one hole centered at position (50,50) of radius 25m (Figure 5(a)), and (2) a network with 5 holes, one at (50,50) of radius $\frac{100}{6}$m and the other four of radius $\frac{100}{12}$m at the four corners of the field (Figure 5(b)).

In the following sections, we discuss the results (in *present tense* for ease of presentation) of the following studies: quality and efficiency of SVM classification, comparisons between LSVM and two existing techniques (Diffusion [6], [9] and AFL [4]), and, finally, the performance of LSVM under the effects of beacon population, network density, the border problem, and coverage holes.

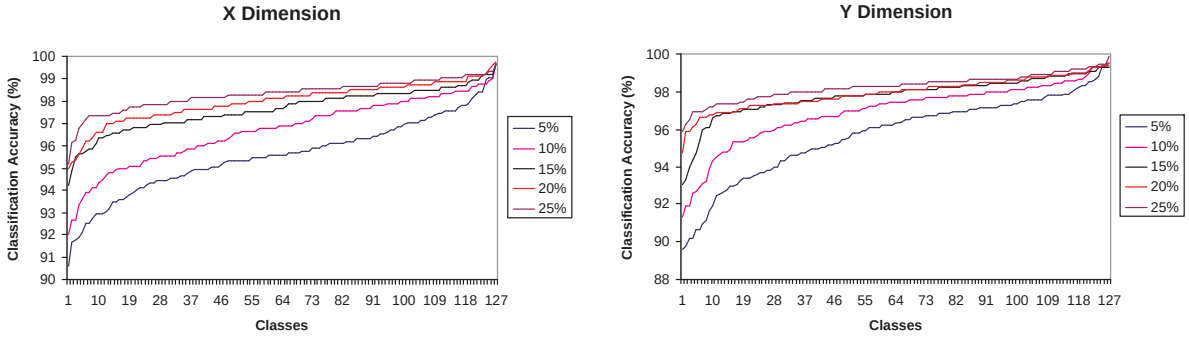
A. Quality and efficiency of SVM classification

As analyzed in Section III-D, the SVM classification error is an important factor to the accuracy of LSVM. The quality of SVM is demonstrated in Figure 6, in which the accuracies



(a) One coverage hole in the middle (b) Five coverage holes: one in the middle, four in the four corners

Fig. 5. Two settings for a sensor network with coverage holes



(a) X-dimension classes cx_1, cx_2, ..., cx_{127} (b) Y-dimension classes cy_1, cy_2, ..., cy_{127}

Fig. 6. The accuracy of SVM for 127 classes per dimension under different beacon populations (5%, 10%, 15%, 20%, and 25%). Curves for higher beacon populations appear above that for smaller populations

for the 127 x-classifications and 127 y-classifications are sorted in the non-decreasing order. It is understandable that SVM's accuracy increases with the beacon population, but the accuracy is remarkably high in all scenarios (more than 89%, mostly more than 95%). This nice result strongly supports our approach of using SVM classification for the sensor localization problem. The worst-case and average per-class SVM accuracy are summarized in Table II. Based on the worst-case error values, we can find the best parameter m for each case of beacon population, that minimizes the error bound in Theorem 3.2 (see Figure 4). In the present paper, however, we report the results for the choice m = 7 and note that better results for LSVM can be obtained if we use the right value for m for each beacon population.

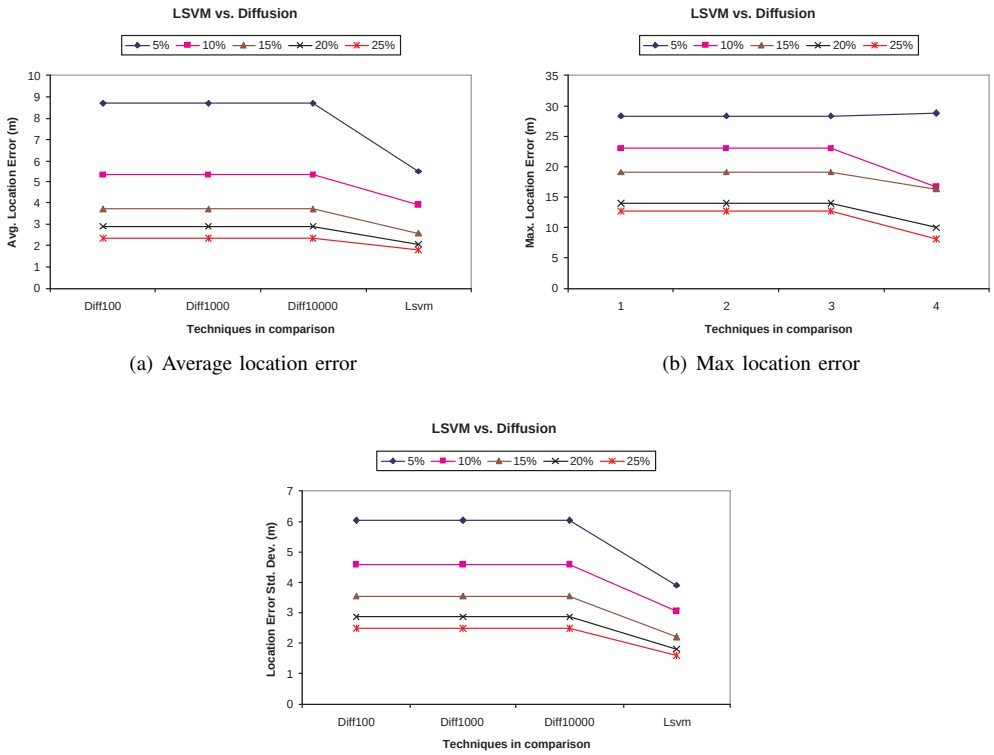
We also compute the number of support vectors (k_{sv}) for each class. Since SVM prediction is based only on the support vectors, but not all the beacons, a small k_{sv} per class is desirable because it helps reduce (1) the amount of SVM model information which is transmitted during the Advertising Phase (see Section III-C.2), and (2) the computational cost during the Localization Phase at each non-beacon sensor (see Section III-C.3). Table III shows that SVM training for most classes results in a small number of support vectors (between 12 and 42 for all scenarios on average).

TABLE IV
LOCATION-ERROR IMPROVEMENT OF LSVM OVER DIFFUSION FOR NETWORKS WITH COVERAGE HOLES

%	Improve	5%	10%	15%	20%	25%
1 hole	Avg	30.97%	31.30%	34.88%	33.91%	26.50%
	Worse	27.35%	21.40%	34.35%	33.42%	23.83%
	StdDev	35.74%	36.29%	37.66%	37.40%	28.34%
5 holes	Avg	32.30%	38.82%	38.46%	31.84%	24.51%
	Worst	22.11%	0.18%	55.38%	48.63%	37.62%
	StdDev	31.01%	48.04%	50.84%	47.87%	35.13%

B. LSVM vs. Diffusion [6], [9]

Because Diffusion is an existing technique that shares the same assumptions and requirements with LSVM, we compare them together. For this comparison, we consider LSVM without using our mass-spring optimization algorithm. We consider 3 versions of Diffusion: Diffusion with 100 averaging iterations (Diff100), with 1000 averaging iterations (Diff1000), and with 10000 averaging iterations (Diff10000). We note that each iteration in Diffusion generates the same traffic as a broadcast over the entire network. Using LSVM, the number



(a) Average location error

(b) Max location error

(c) Standard deviation

Fig. 7. LSVM vs. Diffusion (range 10m): Statistics on location error of each technique under different beacon populations

of broadcasts incurred is the number of beacon sensors, which is much smaller than in the case of Diffusion.

Figure 7 shows the difference in localization accuracy between LSVM and Diffusion. LSVM is more accurate than Diffusion in both the average case (Figure 7(a)) and worst case (Figure 7(b)). The improvement of LSVM over Diffusion is more significant when the beacon population is small. For instance, when 5% of the network size serve as beacons, LSVM results in an average error of 6m while Diff1000 results in 9m. The localization error of LSVM is better distributed across all the sensors than is Diffusion (Figure 7(c)). This is consistent with the fact that in Diffusion the border sensors are localized much worse than the inner sensors (Figure 1(a)). This problem is alleviated in LSVM (Figure 1(b)). Diffusion seems to stall at 10,000 rounds, in which its error is still higher than that of LSVM.

We further compare how LSVM and Diffusion would perform in networks with coverage holes. Table IV shows a significant improvement in location error that LSVM offers over Diff10000. A consistent 20%-50% reduction in error is achieved for all statistics (average, max, and standard deviation).

TABLE V

GLOBAL ENERGY RATIO OF LSVM UNDER DIFFERENT NETWORK DENSITIES AND BEACON POPULATIONS:

Range	5%	10%	15%	20%	25%
10m	0.000365	0.000287	0.000199	0.000179	0.000157
7m	0.000378	0.000266	0.000202	0.000184	0.000164

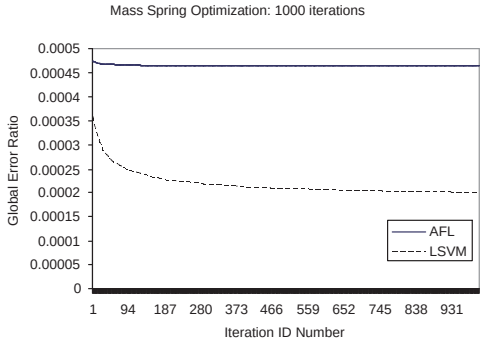


Fig. 8. GER comparison of LSVM (using MMSO) vs. AFL (using MSO) under 1000 iterations of mass-spring optimization

C. LSVM vs. AFL [4]

Anchor-Free Localization (AFL) [4] is a popular existing technique that uses only hop-count information as does LSVM. However, AFL does not use beacon nodes; instead, it assumes that pairwise geographic distances among adjacent nodes can be measured. On the contrary, LSVM uses beacon nodes but not pairwise distances. The purpose of our comparison is to show that LSVM, even with a small beacon population, can still provide better localization accuracy than AFL albeit true geographic distances being used by the latter.

The authors of AFL, Priyantha et al. [4], proposed that the

Global Energy Ratio (GER) be used to assess the localization accuracy. Therefore, we use GER to compare AFL and LSVM:

$$GER = \frac{\sqrt{\sum_{i<j}\left(\frac{dist_{est}(S_i,S_j)-dist_{true}(S_i,S_j)}{dist_{true}(S_i,S_j)}\right)^2}}{N(N-1)/2} \quad (22)$$

GER captures the distance error; that is, to how close the estimated geographic distance $dist_{est}(S_i, S_j)$ between a pair of sensors is to their true geographic distance $dist_{true}(S_i, S_j)$. GER also captures the structural error of the graph induced by the estimated locations.

As reported in [4], the GER of AFL varies between 0.0001 and 0.0012 depending on the error of edge-length measurement and increases with the network density, but only slightly when average nodal degree exceeds 9. Table V provides the GER results of LSVM, which shows that the GER of LSVM is in the top accuracy range of AFL.

AFL must rely on an iterative process of mass-spring optimization (MSO) to refine its location estimation. LSVM has an option of using the modified mass-spring optimization (MMSO) algorithm (i.e., Algorithm 3.2) to refine its localization. Figure 8 plots the GER values for AFL and 50-beacon LSVM during the optimization process of 1000 iterations. Although we use the true geographic distances for the measurement of pairwise distances in AFL, it remains far less accurate than LSVM even though only 50 beacon nodes are used with LSVM. Additionally, LSVM improves significantly faster than AFL as we continue the MSO process. AFL does not seem to ever approach the accuracy of LSVM even when LSVM does not use MMSO. This study suggests that (1) our proposed MMSO algorithm is effective, and (2) when beacon nodes are available, LSVM is much more accurate and faster to converge than AFL. AFL has been effective to initialize the sensor locations in small networks [21]. For large-scale networks, however, LSVM is a more suitable choice.

D. LSVM: Effect of network density and beacon population, the coverage-hole problem, and the border problem

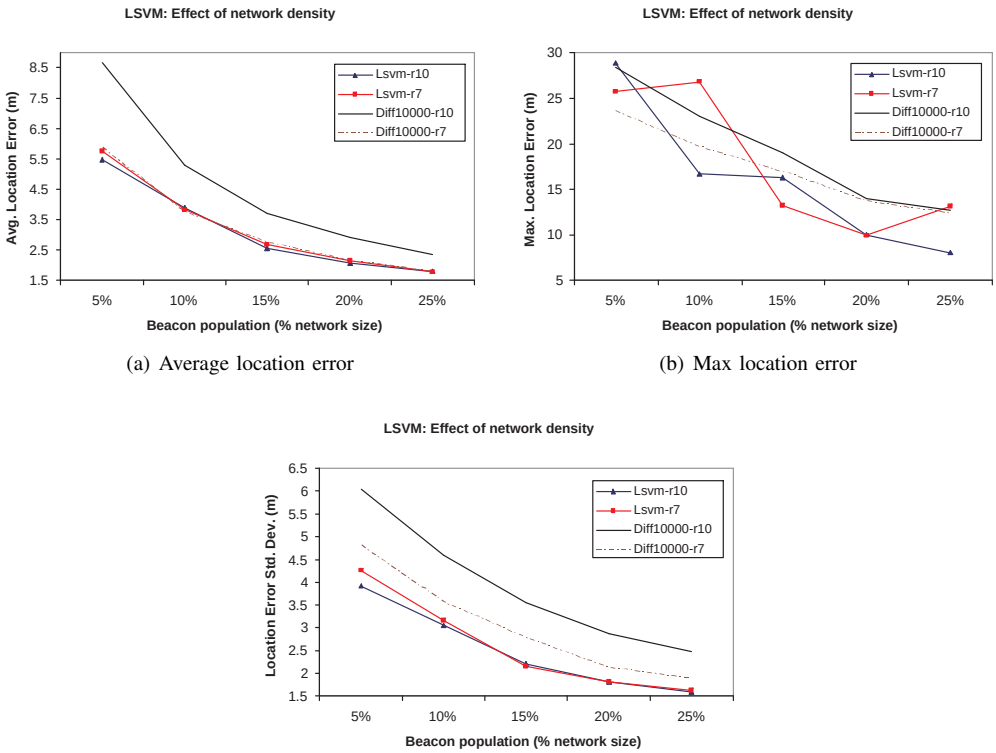
In the remainder of this section, we evaluate LSVM under the effect of network density and beacon population, the coverage-hole problem, and the border problem. We also include the results for Diff1000 (Diffusion with 1000 loops) as a reference. The results are presented below.

1) Network density: We investigate LSVM under two levels of network density. In one network, the communication range of each sensor is set to r = 7m; in the other, set to r = 10m. The connectivity summary of these two networks is shown in Table I. The latter network is two times more connected than the former (i.e., doubling the node degree and edges). Figure 9 suggests that reducing the network density only slightly increase location error of LSVM. In contrary, it is observed that its counterpart Diffusion is more accurate in less network density. In comparison, LSVM in a less dense network (i.e., Lsvm-r7) still performs no worse than the best of its counterpart Diffusion (i.e., Diff1000-r7) in all statistics: average error (Figure 9(a)), worst-case error (Figure 9(b)), and error distribution (Figure 9(c)). In terms of standard deviation, Lsvm-r7 is remarkably better than Diff1000-r7.

2) Number of beacon sensors: Figure 9 also illustrates an obvious feature of LSVM that the localization accuracy gets better as more beacon sensors are used. Using only 5% ($k = 50$), we can locate a sensor within a mean error of 5.5m in a 10,000m^2 square field. When $k = 100$ beacons, the error is reduced to 3.6m. The reduction is less significant when we continue to increase k. This study suggests that for cost effectiveness between 5% and 15% of the network size be used as beacon sensors.

3) The border problem: The border problem, where sensors close to the edge of the sensor field are poorly positioned compared to those deep inside the field, is a challenge for many techniques. We investigate this problem for two types of network (communication range 7m and 10m) and two sizes of beacon population (5% and 25% network size). Figures 10 and 11 show the location error for every sensor node, sorted in the increasing order of sensor distance to the field's origin, for two different network densities ($r = 10$m and $r = 7$m). In other words, those sensors closer to the origin appear before those closer to the edge. In all scenarios, Diff10000 suffers from the border problem severely (i.e., significantly larger error for border-close sensors). On the other hand, LSVM addresses this problem much better, as illustrated in Figures 10(a), 10(c), 11(a), 11(c). The differences between the location errors of sensors closer to the edge and that of sensors inside are much less significant. This property can be explained. In Diffusion, the localization of a sensor is based on its neighbors, whose estimated location in turn depends on other neighbors. Sensors near the border have less neighbor information than those inside the field. Therefore, the former's location estimate should be less accurate. LSVM does not suffer from this problem because the localization of a sensor is only based on the beacon nodes directly, which is independent of other sensors. Therefore, whether a sensor is near the border or the field's center should not have a big impact on its location estimation.

4) Existence of network holes: We consider the two networks demonstrated in Figure 5: a network with 1 hole (Figure 5(a)) and a network with 5 holes (Figure 5(b)). Figure 12 shows the localization error of LSVM in these networks. We also use the error of Diff10000 in the hole-less network as the reference line. It is understandable that LSVM may be less accurate in the networks with holes; however, the existence of coverage holes does not seem to have an impact on LSVM. The reduction in error, if there exists, is very minor. Moreover, LSVM in the networks with holes provides even much better accuracy and error deviation than Diff10000 in the hole-less networks. For example, with 50 beacons, while Diff10000 in the case of no coverage hole has an average error of 8.5m and standard deviation of 6m, the average error and standard deviation of LSVM in the case of networks with holes are less than 6.5m and 5m, respectively. Therefore, LSVM can handle not only the border problem, but also the coverage-hole problem. Similar to the border issue, the coverage-hole issue can be explained as well. Although there are network holes, the

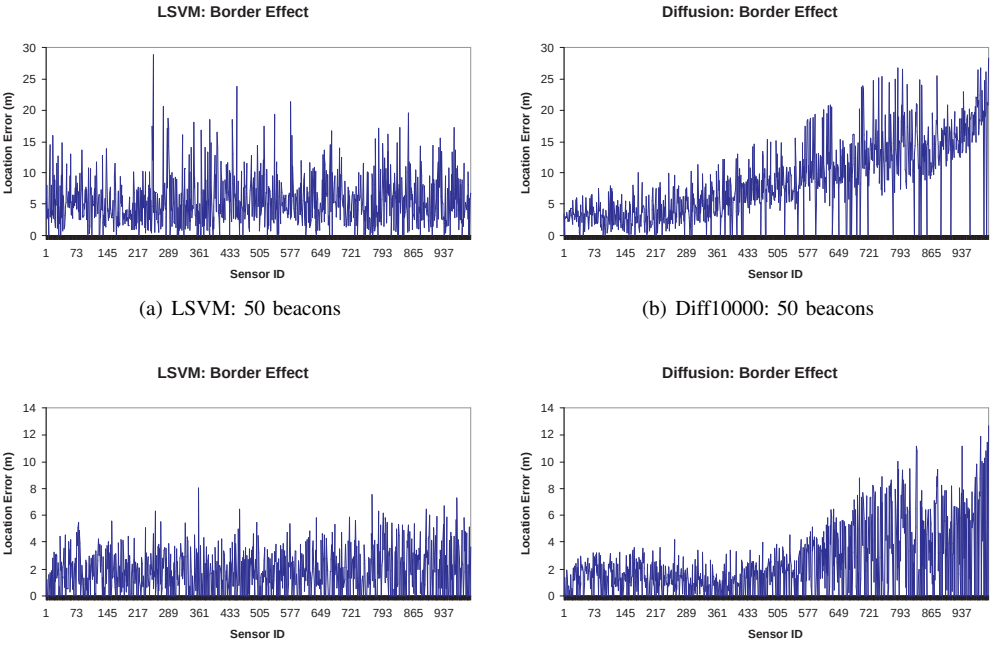


(a) Average location error

(b) Max location error

(c) Standard deviation

Fig. 9. LSVM under the effect of network density (r = 7m and r = 10m): Statistics on location error per each beacon population



(a) LSVM: 50 beacons

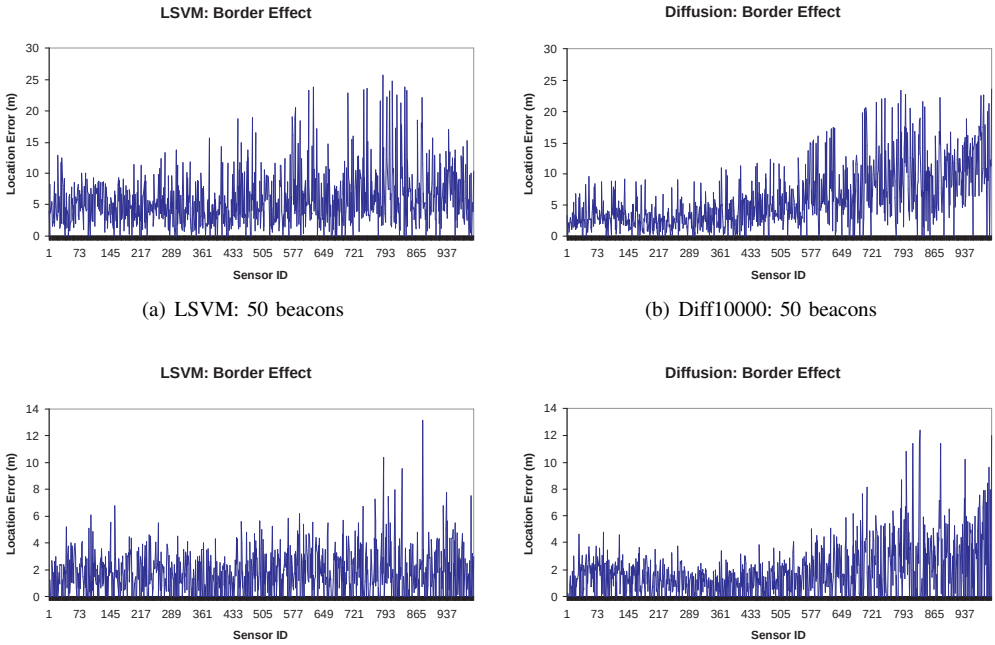
(b) Diff10000: 50 beacons

(c) LSVM: 250 beacons

(d) Diff10000: 250 beacons

Fig. 10. The border problem (LSVM vs. Diffusion, 10m range): Location error for every sensor, sorted in the order of nodes close to the origin of the field first and nodes close to the edge of the field last. The border problem is resolved well with LSVM; it is severe in Diffusion because the error is significantly increased toward the edge

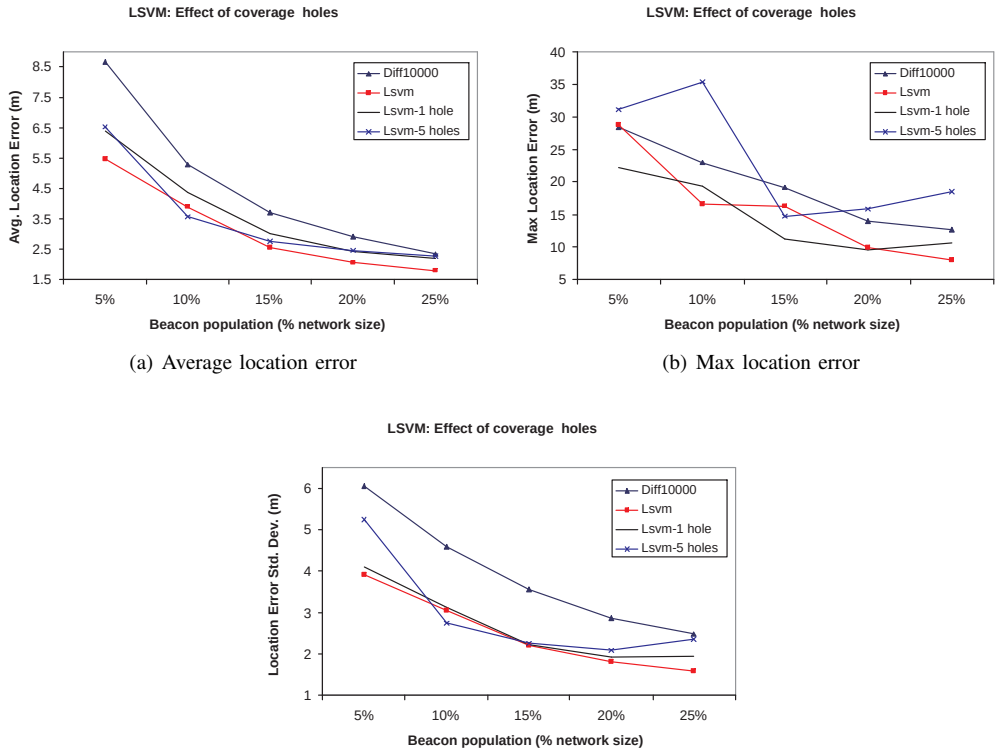
hop-count distances between a sensor and only a small number of beacons become less representative of their true geographic



(a) LSVM: 50 beacons (b) Diff10000: 50 beacons

(c) LSVM: 250 beacons (d) Diff10000: 250 beacons

Fig. 11. The border problem (LSVM vs. Diffusion, 7m range): In the less-dense network, the border problem is also well-resolved with LSVM; it remains severe in Diffusion



(a) Average location error (b) Max location error

(c) Standard deviation

Fig. 12. LSVM's accuracy in networks with coverage holes. These networks are illustrated in Figure 5

distances. The majority of beacons should have their hop-count distance to the sensor same as if no network hole existed. SVM classification should remain highly accurate, and, so should the accuracy of LSVM.

V. Conclusion

We have presented LSVM – a distributed localization technique for sensor networks based on the concept of Support Vector Machines. LSVM assumes the existence of a number of beacons and uses them as training data to the learning process.

Only mere connectivity information is used in LSVM, making it suitable for networks that do not require pairwise distance measurement and specialized (and/or mobile) assisting devices. LSVM yet provides encouraging results. Our simulation study have shown that LSVM outperforms Diffusion, a popular approach that shares the same assumptions with LSVM. With a small beacon population, LSVM has been shown to also be faster converging and more accurate than AFL, a popular technique that requires pairwise distance measurement. LSVM alleviates the border problem and remains effective in networks with coverage holes/obstacles, which many other techniques currently suffer from. The communication and processing overheads are kept small. We have also evaluated LSVM analytically and provided theoretical bounds on its localization accuracy.

In network scenarios that can afford ranging and assisting devices to measure geographic pair-wise distances, because of its simplicity and fast convergence, LSVM can be used to provide good starting locations for the sensors. The location estimation of LSVM can, optionally, be refined using our modified mass-spring optimization algorithm.

The next step in our research is to evaluate LSVM in other network scenarios, including various distribution models for the sensor locations, coverage holes, and network density. We would also like to implement LSVM as a real prototype system and investigate its applicability in fading environments as well as networks of moving sensors with different communication ranges.

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] N. B. Priyantha, "The cricket indoor location system," Ph.D. dissertation, Massachussette Institute of Technology, 2005.

[18] Y. Kwon, K. Mechitov, S. Sundresh, W. Kim, and G. Agha, "Resilient localization for sensor networks in outdoor environments," University of Illinois at Urbana-Champaign, Tech. Rep., June 2004.

[19] N. Priyantha, A. Miu, H. Balakrishnan, and S. Teller, "The cricket compass for context-aware mobile applications," in *ACM conference on mobile computing and networking (MOBICOM)*, 2001.

[20] 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.

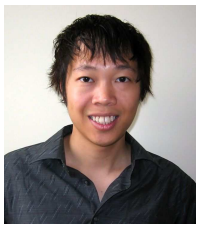
[21] N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller, "Mobile-Assisted Localization in Wireless Sensor Networks," in *IEEE INFOCOM*, Miami, FL, March 2005.

[22] 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.

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

[24] 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

[25] N. Cristianini and J. Shawe-Taylor, *An introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, 2003.



Duc A. Tran is an Assistant Professor at the University of Dayton, where he conducts research in the areas of Computer Networks, Distributed Systems, and Multimedia Systems. His current work is funded by the NSF and the Ohio Board of Regents. Dr. Tran has served as a Vice Program Chair for IEEE AINA 2007, journal guest-editor, TPC member for 12 international conferences, and referee for numerous ACM/IEEE international conferences and journals. His PhD degree in Computer Science was from the University of Central Florida in May 2003, where he also received the Distinguished Doctoral Research Award, IEEE-Orlando Outstanding Graduate Student Award, and the Order of Pegasus Award.

Thinh Nguyen has been an Assistant Professor at Oregon State University since 2004. He earned a B.S. from the University of Washington, and an M.S. and Ph.D. from U.C. Berkeley in 2000 and 2003, respectively. During 2003-2004, he was a post-doctoral research associate at Lawrence Livermore National Laboratory. During 1996-1998, he was a graphics researcher at Intel's Microcomputer Research Lab. He also spent 6 months at Microsoft, optimizing DirectX6 for Pentium III. Dr. Nguyen's current research interests include networking, signal processing, computer graphics, machine learning, data analysis and data mining.