A Kernel-Based Learning Approach to Ad Hoc Sensor Network Localization

XUANLONG NGUYEN, MICHAEL I. JORDAN, and BRUNO SINOPOLI
University of California, Berkeley

We show that the coarse-grained and fine-grained localization problems for ad hoc sensor networks can be posed and solved as a pattern recognition problem using kernel methods from statistical learning theory. This stems from an observation that the kernel function, which is a similarity measure critical to the effectiveness of a kernel-based learning algorithm, can be naturally defined in terms of the matrix of signal strengths received by the sensors. Thus we work in the natural coordinate system provided by the physical devices. This not only allows us to sidestep the difficult ranging procedure required by many existing localization algorithms in the literature, but also enables us to derive a simple and effective localization algorithm. The algorithm is particularly suitable for networks with densely distributed sensors, most of whose locations are unknown. The computations are initially performed at the base sensors, and the computation cost depends only on the number of base sensors. The localization step for each sensor of unknown location is then performed locally in linear time. We present an analysis of the localization error bounds, and provide an evaluation of our algorithm on both simulated and real sensor networks.

Categories and Subject Descriptors: C.2.1 [**Computer Communications Networks**]: Network Architecture and Design—*Network communications, Distributed networks*; I.5.4 [**Pattern Recognition**]: Applications

General Terms: Algorithms

Additional Key Words and Phrases: Ad hoc wireless sensor networks, localization, kernel methods, statistical machine learning, position estimation

1. INTRODUCTION

A sensor network can be viewed as a distributed pattern recognition device. In the pattern recognition approach, rather than transforming sensor locations and sensor readings into Euclidean, world-centric coordinates, we work directly with the (non-Euclidean) coordinate system given by the physical sensor readings themselves. Using the methodology of "kernel functions," the topology implicit in sets of sensor readings can be exploited in the construction of signal-based function spaces that are useful for the prediction of various extrinsic

This work was supported by Grant 0412995 from the National Science Foundation.
Authors' addresses: X. Nguyen, Department of Electrical Engineering and Computer Science, 387 Soda Hall, University of California, Berkeley, CA 94720-1776; email: xuanlong@cs.berkeley.edu.

quantities of interest, using any of a variety of statistical algorithms for regression and classification. In the current article, we illustrate this approach in the setting of a localization problem [Hightower and Borriello 2000; Bulusu et al. 2000; Savarese et al. 2002].

The localization problem that we study is that of determining the location of a (large) number of sensors of unknown location, based on the known location of a (small) number of base sensors. Let $X_1, \dots, X_m$ denote a set of m sensors, and let x_i denote the position in $\mathbb{R}^2$ of sensor X_i. Suppose that the locations of the first n sensors are known, that is, $X_1 = x_1, \dots, X_n = x_n$, where $n \ll m$. We want to recover the positions of $X_{n+1}, \dots, X_m$ solely on the basis of the receive/transmit signals $s(x_i, x_j)$ between pairs of sensors.

An important characteristic of radio or light signal strength is the relationship of the signal attenuation as a function of distance [Seidel and Rappaport 1992]. For instance, for radio signals in an idealized environment, given that the sending and receiving antennas are focused on the same radio frequency, we have

$$s \propto Pd^{-\eta}, \tag{1}$$

where $\eta > 2$ is a constant, and P is the sending signal voltage. Such relationships provide the basis for a variety of localization algorithms in the literature which consist of two main steps: (1) a ranging procedure which involves estimating the distance from a sensor to another sensor based on the signal strength of the signals transmitted/received between the two, and (2) a procedure that recovers the locations of the sensors based on their pairwise distance estimates either by triangulation or by least-squares methods [Priyantha et al. 2000; Girod and Estrin 2001; Savvides et al. 2001; Whitehouse 2002]. Unfortunately, however, the idealized model in Eq. (1) can be highly inaccurate due to variability caused by multipath effects and ambient noise interference as well as device-specific factors such as the frequencies of node radios, physical antenna orientation, and fluctuations in the power source [Bulusu et al. 2000; Priyantha et al. 2000]. Methods based on ranging inherit these inaccuracies, and improvements are possible only if difficult problems in signal modeling are addressed.

In this article, we propose a method that bypasses the ranging step altogether. We show that it is possible to pose a coarse-grained localization problem as a discriminative classification problem that can be solved using tools from the statistical machine learning literature. Fine-grained localization is then achieved by a second application of the coarse-grained localization technique. Our localization algorithm thus involves two phases. First, there is a training phase that chooses discriminant functions for classifying positions using arbitrarily constructed target regions. This phase is performed either online at the base stations, or taken offline, and takes $O(n^3)$ computational time, where n is the number of base sensors. Hence, our assumption is that the base sensors have sufficient power and processing capability (indeed, these are also the nodes that might have GPS-capability to determine their own exact locations). Second, once the training phase is completed, other location-unknown low-power sensors can determine their own position locally, and the computation takes only $O(n)$ time for each of these sensors.

Our approach makes use of kernel methods for statistical classification and regression [Schölkopf and Smola 2002], an example of which is the "support vector machine (SVM)." Central to this approach is the notion of a *kernel function* which provides a generalized measure of similarity for any pair of entities (e.g., sensor locations). The functions that are output by the SVM and other kernel methods are sums of kernel functions, with the number of terms in the sum equal to the number of data points. Kernel methods are examples of *nonparametric* statistical procedures—procedures that aim to capture large, open-ended classes of functions.

Kernel functions typically used in practice include Gaussian kernels and polynomial kernels. A technical requirement of these functions is that they are positive semidefinite which is equivalent to the requirement that the $n \times n$ *Gram matrix* formed by evaluating the kernel on all pairs of n data points is a positive semidefinite matrix. Intuitively, this requirement allows a kernel function to be interpreted as a generalized measure of similarity. The kernel function imposes a topology on the data points which is assumed to be useful for the prediction of extrinsic quantities such as classification labels.

Given that the raw signal readings in a sensor network implicitly capture topological relations among the sensors, kernel methods would seem to be particularly natural in the sensor network setting. In the simplest case, the signal strength would itself be a kernel function and the *signal matrix* $(s(x_i, x_j))_{ij}$ would be a positive semidefinite matrix. Alternatively, the matrix may be well approximated by a positive semidefinite matrix (e.g., a simple transformation that symmetrizes the signal matrix and adds a scaled identity matrix may be sufficient). More generally, and more realistically, derived kernels can be defined based on the signal matrix. In particular, inner products between vectors of received signal strengths necessarily define a positive semidefinite matrix and can be used in kernel methods. Alternatively, generalized inner products of these vectors can be computed—this simply involves the use of higher-level kernels whose arguments are transformations induced by lower-level kernels. In general, hierarchies of kernels can be defined to convert the initial topology provided by the raw sensor readings into a topology more appropriate for the classification or regression task at hand. This can be done with little or no knowledge of the physical sensor model.

Our focus is on the discriminative classification problem of locating sensors in an ad hoc sensor network. It is worth noting that similar methods have been explored recently in the context of tracking one or more objects (e.g., mobile robots) that move through a wireless sensor field.[1] Systems of this type include Active Badge [Want et al. 1992; Ward et al. 1997], RADAR [Bahl and Padmanabhan 2000], Cricket [Priyantha et al. 2000], and UW-CSP [Li et al. 2002]. In Bahl and Padmanabhan [2000], the authors describe a simple nearest neighbor classification algorithm to obtain coarse localization of objects.

[1]The alternative to discriminative classification is classification using *generative* probabilistic models. This is a well-explored area that dates back to contributors such as Wiener and Kalman [Poor 1994]. Recent work in this vein focuses on the distributed and power-constrained setting of wireless sensor networks (e.g. Sheng and Hu [2003]; D'Costa and Sayeed [2003]).

Most closely related to our approach is the work of Li et al. [2002] in which a number of classification algorithms are used for tracking moving vehicles, including k-nearest neighbor and support vector machines. We elaborate on the connections between this work and ours in the description of our algorithm.

The article is organized as follows. We begin with a brief background of classification using kernel methods, and motivate our application of kernel methods to the localization problem based on sensor signal strength. Next, the localization algorithm and its error analysis are described. We then present details of the implementation of the algorithm and its computational cost, followed by an evaluation of our algorithm with simulated and real sensor networks. Finally, we present our conclusions in the final section.

2. CLASSIFICATION USING KERNEL METHODS

In a classification algorithm, we are given as training data n samples $(x_i, y_i)_{i=1}^n$ in $\mathcal{X} \times \{\pm 1\}$, where $\mathcal{X}$ denotes the input space. Each y_i specifies whether the data point $x_n \in \mathcal{X}$ lies in a class $C \subseteq \mathcal{X}$ ($y_i = 1$) or not ($y_i = -1$). A classification algorithm involves finding a discriminant function $y = \text{sign}(f(x))$ that minimizes the classification error $P(Y \neq \text{sign}(f(X)))$.

Central to a kernel-based classification algorithm (e.g., the SVM) is the notion of a kernel function $K(x, x')$ that provides a measure of similarity between two data points x and x' in $\mathcal{X}$. Technically, K is required to be a symmetric positive semidefinite function.[2] For such a function, Mercer's theorem implies that there must exist a feature space $\mathcal{H}$ in which K acts as an inner product, that is, $K(x, x') = \langle \Phi(x), \Phi(x') \rangle$ for some mapping $\Phi(x)$. The SVM and related kernel-based algorithms choose a linear function $f(x) = \langle \mathrm{w}, \Phi(x) \rangle$ in this feature space. That is, they find a vector w which minimizes the loss

$$\sum_{i=1}^n \phi(y_i f(x_i))$$

subject to $||\mathrm{w}|| \leq B$ for some constant B. Here ϕ denotes a convex function that is an upper bound on the 0-1 loss $\mathbb{I}(y \neq \text{sign}(f(x)))$.[3] In particular, the SVM algorithm is based on the hinge loss $\phi(yf(x)) = (1 - yf(x))_+$.[4] By the Representer Theorem (cf. Schölkopf and Smola [2002]), it turns out that the minimizing f can be expressed directly in terms of the kernel function K

$$f(x) = \sum_{i=1}^n \alpha_i K(x_i, x) \tag{2}$$

for an optimizing choice of coefficients α_i.

There are a large number of kernel functions that satisfy the positive semidefinite property required by the SVM algorithm. Examples include

[2]For a translation-invariant kernel, that is, $K(x, x') = h(x - x')$ for some function h, K is a positive semidefinite kernel if the Fourier transform of h is nonnegative.

[3]The indicator function is defined as $\mathbb{I}(A) = 1$ if A is true, and 0 otherwise.

[4]The subscript + notation means that $x_+ = \max(x, 0)$.

the Gaussian kernel,

$$K(x, x') = \exp -(||x - x'||^2/\sigma),$$

as well as the polynomial kernel,

$$K(x, x') = (\gamma + ||x - x'||)^{-\sigma},$$

for parameters σ and γ. Both of these kernel functions decay with respect to the distance $||x - x'||$, a property that is shared by most idealized signal strength models. In particular, the radio signal model (1) has a form similar to that of a polynomial kernel. In Sheng and Hu [2003], the authors justify the use of an acoustic energy model for localization that has the form of the Gaussian kernel. These relationships suggest a basic connection between kernel methods and sensor networks. In particular, a naive usage of kernel methods could be envisaged in which signal strength is used directly to define a kernel function. In general, however, signal strength in real sensor networks need not define a positive semidefinite function. Nonetheless, it is the premise of this article that signal strength matrices provide a useful starting point for defining kernel-based discriminant functions. We show how to define derived kernels which are stacked on top of signal strength measurements in the following section.

Finally, it is worth noting that multimodal signals are naturally accommodated within the kernel framework. Indeed, suppose that we have D types of sensory signals, each of which can be used to define a kernel function $K_d(x, x')$ for $d = 1, \ldots, D$. Then any conic combination of K_d yields a new positive semidefinite function:

$$K(x, x') = \sum_{d=1}^{D} \beta_d K_d(x, x').$$

There are methods for choosing the parameters $\beta_d > 0$ based on empirical data [Lanckriet et al. 2004].

3. LOCALIZATION IN AD HOC SENSOR NETWORK

3.1 Problem Statement

We assume that a large number of sensors are deployed in a geographical area. The input to our algorithm is a set of m sensors, denoted by $X_1, \ldots, X_m$. For each i, we denote by x_i the position in $\mathbb{R}^2$ of sensor X_i. Suppose that the first n sensor locations are known, that is, $X_1 = x_1, \ldots, X_n = x_n$, where $n \ll m$. For every pair of sensors X_i and X_j, we are given the signal $s(x_i, x_j)$ that sensor X_j receives from X_i. We want to recover the positions of $X_{n+1}, \ldots, X_m$.

3.2 Algorithm Description

We first aim to obtain a coarse location estimate for $X_{n+1}, \ldots, X_m$. Given an arbitrarily constructed region $C \subseteq \mathbb{R}^2$, we ask whether $X_i \in C$ or not, for $i = n+1, \ldots, m$. This can be readily formulated as a classification problem. Indeed, since the location of the base sensors $X_1, \ldots, X_n$ are known, we know whether or not each of these base sensors are in C. Hence we have as our training data

n pairs $(x_i, y_i = \text{sign}(x_i \in C))_{i=1}^n$. For any sensor X_j, $j = n+1, \ldots, m$, we can predict whether $X_j \in C$ or not based on the sign of the discriminant function $f(x_j)$:

$$f(x_j) = \sum_{i=1}^{n} \alpha_i K(x_i, x_j). \tag{3}$$

We emphasize that the value of $f(x_j)$ is known because the values of the kernels, $K(x_i, x_j)$, are known despite the fact that we do not know the position x_j per se.

Next, we turn to the definition of the kernel matrix $K = (K(x_i, x_j))_{1 \le i,j \le m}$. In general, we envision a hierarchy of kernels based on the signal matrix. An example of such a hierarchy follows.

(1) We might simply define $K(x_i, x_j) = s(x_i, x_j)$. We call this naive choice a *first-tier* kernel. If the signal matrix $S = (s(x_i, x_j))_{1 \le i,j \le m}$ is a symmetric positive semidefinite Gram matrix, then this approach is mathematically correct although it may not yield optimal performance. If S is not symmetric positive semidefinite, then a possible approximation is $(S + S^T)/2 + \delta I$. This matrix is symmetric and is positive semidefinite for sufficiently large $\delta > 0$ (in particular, for δ larger in absolute value than the most negative eigenvalue of $(S + S^T)/2$).

(2) Alternatively, define $K = S^T S$ to be refered to as a *second-tier* linear kernel. K is always symmetric positive semidefinite. This kernel can be interpreted as an inner product for a feature space $\mathcal{H}$ which is spanned by vectors of the form:

$$\Phi(x) = (s(x, x_1), s(x, x_2), \ldots, s(x, x_m)).$$

Specifically, we define

$$K(x_i, x_j) = \sum_{t=1}^{m} s(x_i, x_t) s(x_j, x_t).$$

Intuitively, the idea is that sensors that are associated with similar vectors of sensor readings are likely to be nearby in space.

(3) Finally, it is also possible to evaluate any kernel function (e.g., Gaussian) on the feature space $\mathcal{H}$ induced by the second-tier kernel. This yields a symmetric positive semidefinite matrix to be refered to as a *third-tier* kernel. Specifically, a third-tier Gaussian kernel has the following form, for a parameter σ:

$$\begin{aligned} K(x_i, x_j) &= \exp\left\{ -\frac{\|\Phi(x_i) - \Phi(x_j)\|^2}{\sigma} \right\} \\ &= \exp\left\{ -\frac{\sum_{t=1}^{m} (s(x_i, x_t) - s(x_j, x_t))^2}{\sigma} \right\}. \end{aligned}$$

Given training data $(x_i, y_i)_{i=1}^n$ and a kernel function K, we apply the SVM algorithm to learn a discriminant function $f(x)$ as in Eq. (2). The algorithmic details and computational costs are discussed in Section 4.

Our classification formulation has several noteworthy characteristics. First, the training points correspond to the base sensors and thus may be limited in number, making the learning problem nominally a difficult one. However, because we are free to choose the target region C, the problem can in fact be made easy. This ability to design the geometry of the boundary to fit the geometry of the classifier distinguishes this problem from a traditional pattern recognition problem.

The second characteristic is that we require that the network be relatively dense. As seen in Eq. (3), the prediction of position is based on a sum over sensors, and an accurate prediction can be achieved in general only if there are enough nonzero terms in the sum for it to be statistically stable.

A related point is that it is not necessary that the network be completely connected. If the sensor reading $s(x_i, x_j)$ is generally small or zero for a pair of sensors, then that term does not perturb the kernel calculation or the discriminant calculation. If readings fluctuate between small values and large nonzero values, then the prediction will generally be degraded. Given that the approach is a statistical approach, however, with predictions based on an aggregation over neighboring sensors, it should be expected to exhibit a certain degree of robustness to fluctuations. This robustness should be enhanced by the protocol for fine-grained estimation as we now discuss.

We turn to the fine-grained estimate of sensor positions. We use the coarse-grained solution presented previously as a subroutine for a localization algorithm for sensors $X_j (j = n+1, \ldots, m)$. The idea is as follows. We fix a number of overlapping regions $C_\beta (\beta = 1, \ldots, U)$ in the geographical region containing the sensor network. For each β, we formulate a corresponding classification problem with respect to class C_β and predict whether or not $X_j \in C_\beta$. Hence, X_j has to be in the intersection of regions that contain it. We might, for example, assign its location x_j to be the centroid of such an intersection. Given an appropriate choice of granularity and shapes for the regions C_β, if most of the classification labels are correct, we expect to be able to obtain a good estimate of x_j.

As we have seen in our experiments on both simulated data (using a Gaussian or polynomial kernel) and real sensor data (using kernels that are constructed directly from the signal matrix), given a sufficient number of base sensors (i.e., training data points), the SVM algorithm can fit regions of arbitrary shape and size with reasonable accuracy. When the number of base sensors is limited, the SVM algorithm can still fit elliptic shapes very well. This can be turned to our advantage for fine-grained localization. By picking appropriate regions C_β such as ellipses that are easy to classify, we do not need many base sensors to achieve reasonable localization performance for the entire network. In the sequel, we will show that this intuition can be quantified to give an upper bound on the expected (fine-grained) localization error with respect to the number of base sensors.

3.3 Localization Error Analysis

Suppose that the sensor network of size $L \times L$ is covered uniformly by k^2 discs with radius R. Then any given point in the sensor network is covered by approximately $\pi(Rk/L)^2$ discs. Each of these discs are used to define the region for a region classification problem. To obtain a fine-grained location estimate for all remaining sensors, X_j for $j = n+1, \ldots, m$, we need to solve k^2 region classification problems. Let e_β be the training error for each of these problems, for $\beta = 1, \ldots, k^2$. That is,

$$e_\beta = \sum_{i=1}^{n} \phi(\text{sign}(x_i \in C_\beta) f(x_i)).$$

Since the size and shape of the regions are ours to decide, it is reasonable to assume that the training error for these classification problems are small. For instance, the circle/elliptic shape is particularly suited for Gaussian or polynomial kernels. Define $\epsilon(R)$ to be the upper bound for all training errors

$$\epsilon(R) = \max_{1 \le \beta \le k^2} e_\beta.$$

From statistical learning theory [Vapnik 1998], for each $\beta = 1, \ldots, k^2$, the probability of misclassification for each new sensor X_j and region C_β is $e_\beta + O(1/\sqrt{n})$, where n is the number of training points (i.e., number of base sensors). Since each location is covered by $\pi R^2 k^2 / L^2$ discs, the probability of misclassification for at least one of these covering discs is, by the union bound, less than $\frac{\pi R^2 k^2}{L^2}(\epsilon(R) + O(1/\sqrt{n}))$. If a given sensor is correctly classified with respect to all of its covering discs then we assign the sensor's location to be the center of the intersection of all these discs in which case the localization error is bounded by $O(L/k)$.

Hence, the expectation of the localization error is bounded by

$$O\left(\frac{L}{k}\right) + \frac{\pi R^2 k^2}{L}(\epsilon(R) + O(1/\sqrt{n})).$$

This asymptotic bound is minimized by letting $k \propto L^{2/3} R^{-2/3}(\epsilon(R) + O(1/\sqrt{n}))^{-1/3}$. The bound then becomes $O(L^{1/3} R^{2/3}(\epsilon(R) + O(1/\sqrt{n}))^{1/3})$.

In summary, we have proved the following:

PROPOSITION 3.1. *Assume that all sensor locations are independently and identically distributed according to an (unknown) distribution. For any sensor location x, let $\hat{x}$ be the location estimate given by our algorithm, then.*

$$\mathbb{E}||x - \hat{x}|| \le O(L^{1/3} R^{2/3}(\epsilon(R) + O(1/\sqrt{n}))^{1/3}).$$

This result has the following consequences for the expected variation of the fine-grained localization error as a function of the parameters n (the number of base sensors), R (the size of the discs), and k^2 (the number of discs).

(1) The fine-grained localization error decreases as the sensor network becomes more densely distributed (i.e., n increases). In addition, the localization error increases with the size of the network, but this increase is at most linear.
(2) The fine-grained localization error increases as R increases; on the other hand, as R increases, the optimal value of k decreases, resulting in a smaller

computational cost because there are k^2 discs to classify. Hence, variation in R induces a trade-off between localization accuracy and computational complexity.

(3) We would expect the localization error to increase at a rate $O(R^{2/3})$ if $\epsilon(R)$ were to remain constant. However, as R increases, the length of the boundary of the regions C_β also increases, and the training error $\epsilon(R)$ is expected to increase as well. As a result, we expect the localization error to actually increase faster than $O(R^{2/3})$.

Note that our analysis makes some simplifying assumptions—it assumes a uniform distribution for the locations of regions C_β, and it assumes circular shapes. While the analysis can be readily generalized to other specific choices, it would be of substantial interest to develop a general optimization-theoretic approach to the problem of choosing the regions.

4. ALGORITHM DETAILS AND COMPUTATIONAL COST

During the training phase associated with each coarse localization subroutine, that is, classification with respect to a fixed region C_β, we construct the training data set based on the locations of the base sensors as described in the previous section. This is achieved by having all base stations send the signal matrix entries $s(x_i, x_j)$ and their known locations to a central station, a procedure which involves receiving and storing $n^2 + n$ numbers at the central station. The central station then solves the following optimization problem:

$$\min_{\mathrm{w}} ||\mathrm{w}||^2 + \frac{c}{n}\sum_{i=1}^{n} \phi(y_i f(x_i)),$$

where $f(x) = \langle \mathrm{w}, \Phi(x)\rangle$, $\phi(yf(x)) = (1 - yf(x))_+$, and c is a fixed parameter.[5] This is a convex optimization problem, which has the following dual form [Schölkopf and Smola 2002]:

$$\max_{0 \le \alpha \le c} 2\sum_{i=1}^{n} \alpha_i - \sum_{1 \le i,\, j \le n} \alpha_i \alpha_j y_i y_j K(x_i, x_j). \tag{4}$$

The algorithm finds optimizing values of $\{\alpha_i\}$ which are then used to form the discriminant function in Eq. (2).

It is known that the solution to this optimization problem can be found in the worst case in $O(n^3)$ computational time. Thus if there are k^2 regions to classify, this result suggests a total training time of $O(n^3k^2)$. However, this generic worst-case estimate is overly conservative in our setting. Indeed, an estimate based on the number of support vectors n_s returned by each classification algorithm (those X_i such that $\alpha_i \neq 0$) reveals that the computational complexity is $O(n_s^3 + n_s^2 n)$ instead of $O(n^3)$. Usually $n_s \ll n$. Our simulation experience (to be presented in the next section) shows that when discs with radius R are used, the support vectors reside mostly along the boundaries of the discs, hence

[5]The parameter c is a regularization parameter associated with the SVM algorithm. In all our experiments, we fix $c = 10$.

```
Coarse localization algorithm
Input: $X_i = x_i \in \mathbb{R}^2$ for $i = 1, \ldots, n$; signal matrix $[s(x_i, x_j)]_{1 \le i,j \le m}$ where $n \ll m$;
a region $C \subseteq \mathbb{R}^2$.
Output: $y_j \in \{\pm 1\}$ for $j = n+1, \ldots, m$.

(1) For $i = 1, \ldots, n$, let $y_i = \text{sign}(x_i \in C)$.
(2) Define a positive semidefinite kernel matrix $[K(x_i, x_j)]_{1 \le i,j \le m}$ based upon
    $[s(x_i, x_j)]_{ij}$.
(3) Solve the optimization problem (4) for optimum $\{\alpha_i\}_{i=1}^n$.
(4) For $j = n+1, \ldots, m$, $y_j = \text{sign}\left(\sum_{i=1}^n \alpha_i K(x_i, x_j)\right)$.
```

Fig. 1. Summary of the coarse localization algorithm.

$n_s \approx O(\min(n\pi R^2/L^2, 2\pi R))$ in which case the overall training phase takes only $O(R^2 nk^2)$ time. Note also that this training phase is the most expensive part of our algorithm and is performed at a central station.

Once the training phase is complete, each base sensor is required to store the n parameters $(\alpha_1, \ldots, \alpha_n)$ for the purpose of classification of the remaining (location-unknown) sensors. If the first-tier kernel is used, a new sensor X_j for $j = n+1, \ldots, m$ records the signal $s(x_i, x_j)$ from the n_s base sensors $i \in \{1, \ldots, n\}$ and combines these with the nonzero values α_i, resulting in a cost of $O(n_s)$ in time and storage. If a second-tier linear kernel or a third-tier Gaussian kernel is used, a new sensor X_j records n-element signal vectors $(s(x_j, x_1), \ldots, s(x_j, x_n))$ from the n_s base stations, resulting in a $O(n_s n)$ cost in time and storage. The kernel values $K(x_i, x_j)$ are then readily computable from the received signals $s(x_i, x_j)$ in $O(1)$, $O(n)$, $O(n^2)$ time for the first-tier, second-tier and third-tier kernel, respectively. Then a simple computation (Eq. (3)) determines for sensor X_j whether it resides in the region C or not. The attractive feature of the localizing step is that it is done locally (in a distributed fashion), taking only linear (for the first-tier and second-tier kernels) or quadratic (for the third-tier Gaussian kernel) time and storage space (in terms of n). Since the localization is done on an as needed basis, its time and storage cost do not depend on the total number of sensors m in the network. A summary of our algorithm is provided in Figure 1.

Now we turn to fine-grained localization. At both algorithmic and system levels, this involves invoking the coarse localization subroutine k^2 times with respect to regions $C_1, \ldots, C_{k^2}$. Therefore, for each region $\beta = 1, \ldots, k^2$, we have a set of parameters $(\alpha_i)_{i=1}^n$. Each sensor X_j can then determine its location by setting x_j to be the centroid of the intersection of all regions C_β that it finds itself residing in. In the case in which C_β are discs with centers c_β, this yields

$$x_j := \frac{\sum_{\beta=1}^{k^2} c_\beta \mathbb{I}(X_j \in C_\beta)}{\sum_{\beta=1}^{k^2} \mathbb{I}(X_j \in C_\beta)}.$$

Clearly, the computational cost of a fine-grained localization algorithm is k^2 times as much as the computational cost of each coarse localization step. In summary, our fine-grained localization algorithm is shown in Figure 2.

```
Fine-grained localization algorithm
Input: $X_i = x_i \in \mathbb{R}^2$ for $i = 1, \ldots, n$; signal matrix $[s(x_i, x_j)]_{1 \le i,j \le m}$ where $n \ll m$; $k$; $R$.
Output: $x_j \in \mathbb{R}^2$ for $j = n+1, \ldots, m$.

(1) Let $A_1 = \min\{(x_i)_1\}_{i=1}^n$; $B_1 = \max\{(x_i)_1\}_{i=1}^n$; $A_2 = \min\{(x_i)_2\}_{i=1}^n$; $B_2 = \max\{(x_i)_2\}_{i=1}^n$.
(2) Let $C_\beta$ for $\beta = 1, \ldots, k^2$ be $k^2$ discs with radius $R$ distributed uniformly in a grid of coordinates $[A_1, B_1] \times [A_2, B_2]$.
(3) For $\beta = 1, \ldots, k^2$, run the coarse localization algorithm with respect to region $C_\beta$ to get values $\{y_{\beta,j}\}_{j=n+1}^m$.
(4) Letting $c_\beta$ be the center of $C_\beta$ for $\beta = 1 \ldots, k^2$, then:
    $x_j = \frac{\sum_{\beta=1}^{k^2} c_\beta \mathbb{I}(y_{\beta,j}=1)}{\sum_{\beta=1}^{k^2} \mathbb{I}(y_{\beta,j}=1)}$.
```

Fig. 2. Summary of the fine-grained localization algorithm.

5. EXPERIMENTAL RESULTS

We evaluate our algorithm on simulated sensor networks in the first two sections, and then on a real network using Berkeley sensor motes.

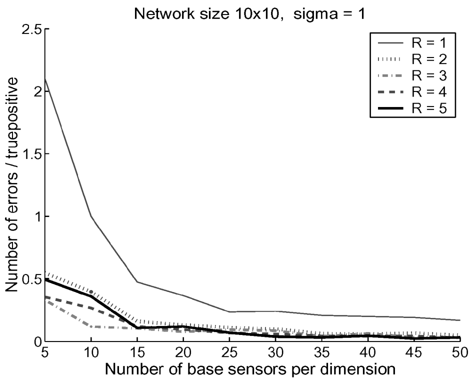
5.1 Coarse Localization

Simulation set-up. We consider a network of size 10×10 square units. The base sensors are distributed uniformly in a grid-like structure. There are a total of n such sensors. We are concerned with recognizing whether a sensor position x, characterized by the signal reading $s(x_i, x)$ for $i = 1, \ldots, n$, lies in a region C or not.

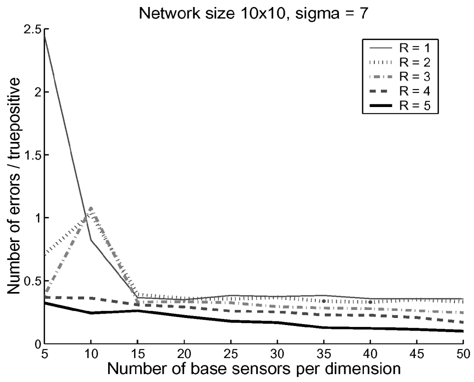
We first define a signal model: Each sensor location x is assumed to receive from a sensor located at x' a signal value following a fading channel model: $s(x, x') = \exp -\frac{||x-x'||^2}{\sigma} + N(0, \tau)$, where $N(0, \tau)$ denotes an independently generated normal random variable with standard deviation τ. This signal model is a randomized version of a Gaussian kernel. We have also experimented with a signal strength model that is a randomized version of the polynomial kernel: $s(x, x') = (||x - x'||)^{-\sigma} + N(0, \tau)$. The results for the polynomial kernels are similar to the Gaussian kernels, and are not presented here. It is emphasized that, although the use of these models have been motivated elsewhere as signal models [Seidel and Rappaport 1992; Sheng and Hu 2003], in our case, they are used merely to generate the signal matrix S. Our algorithm is not provided with any knowledge of the procedure that generates S.

Next, we define a region C to be recognized. In particular, C consists of all locations x that satisfy the following equations: $(x - v)^T H_1(x - v) \le R$ and $(x - v)^T H_2(x - v) \le R$, where $v = [5\ 5]^T$, $H_1 = [2\ -1; -1\ 1]$ and $H_2 = [2\ 1; 1\ 1]$. The radius R is used to describe the size of C. For each simulation set-up (n, R, σ, τ), we learn a discriminant function f for the region C using the training data given by the base sensor positions. Once f is learned, we test the classification at 100×100 sensor locations distributed uniformly in the region containing the network.

Figure 4(a) illustrates C as a shaded region for $R = 2$, while the black boundary represents the region learned by our localization algorithm. Qualitatively,

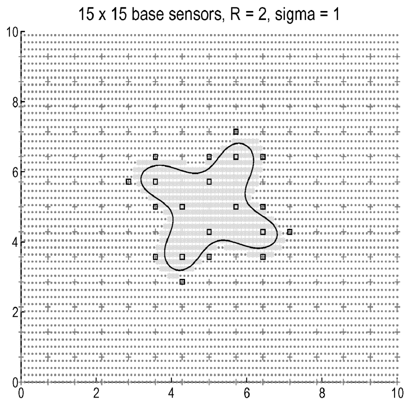


(a)

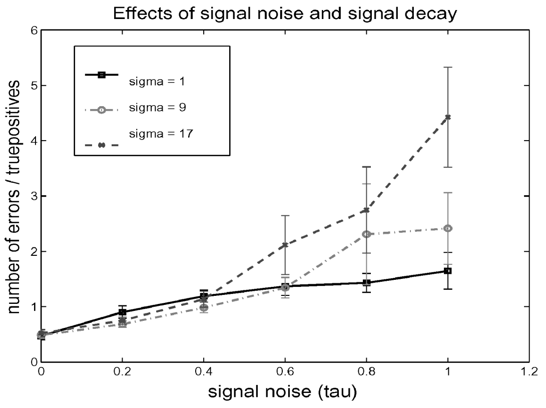


(b)

Fig. 3. Simulation results with (randomized) Gaussian models. The x-axis shows the number of sensors employed along each dimension of the network. The y-axis shows the ratio between the number of incorrectly classified points and the number of points inside the area to be recognized. (Note that this ratio is larger than the overall failure rate; in the latter, the denominator includes the points outside the area to be recognized).



(a)



(b)

Fig. 4. (a) Illustration of a simulated sensor network with 15×15 base sensors and the recognized boundary in black (with $R = 2$) using a Gaussian kernel with $\sigma = 1$. The black squares are the support vector base sensors. The test error in this figure is 0.27. (b) Plots show the effect of the sensor fading signal parameter σ and signal noise parameter τ on coarse localization performance.

the algorithm has captured the shape of the target region C. We now present a quantitative analysis of the effects of n, R, σ, and τ on the localization (i.e., classification) performance:

Effects of n. The plots in Figure 3 show the localization (test) error with respect to the number of base sensors deployed in the network. The test error is defined to be the ratio between the number of misclassified points and the number of points located within the area C (out of 100×100 locations distributed uniformly in the grid). In this set of simulations, we fix the noise parameter $\tau = 0$, and let $\sigma = 1$ and $\sigma = 7$, while varying n. The plots confirm that the localization error tends to decrease as the sensor network becomes more densely distributed. Note that if we need to recognize a particular area, we only need to plant base sensors in the area near the boundary because these are the likely locations of support vectors. Of course, in our context, coarse-grained localization is only a subroutine for fine-grained localization, and it is in our interest to have base sensors spread throughout the whole geographical area.

Effects of σ and τ. The parameter σ is used to describe the sensitivity of the signal strength with respect to the sensor distance. In particular, for a Gaussian signal function, a small value of σ implies that the signal strength fades very quickly for distant sensors. The plots in Figure 4(b) display the effects of both σ and τ on the localization performance. In this set of simulations, we fix the number of base sensors along each dimension to be 10, and set the radius of C to be $R = 2$, while varying σ and τ. The localization performance degrades as we increase the noise parameter τ, and the degradation is more severe for the least sensitive signal, that is, when σ is large.

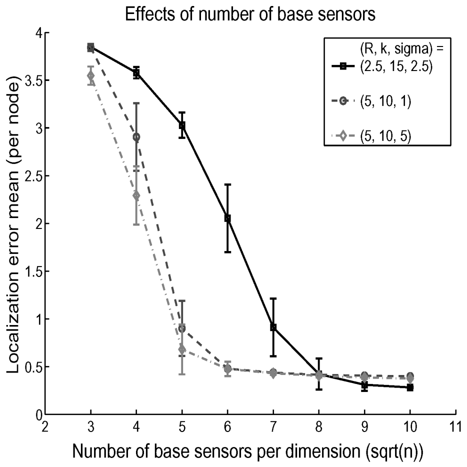
5.2 Fine-Grained Localization

Simulation set-up. The network set-up is the same as the previous section, except that the n base sensors are now distributed approximately uniformly at random in the whole area. By this we mean that each base sensor is initially planted at a grid point in the $L \times L$ square, where $L = 10$, and then perturbed by Gaussian noise $N(0, L/(2\sqrt{n}))$. There are 400 other sensors whose locations are to be determined using our algorithm. These 400 sensors are deployed uniformly in the grid. Again, we assume the signal strength follows a Gaussian signal model with noise parameter $\tau = 0.2$.

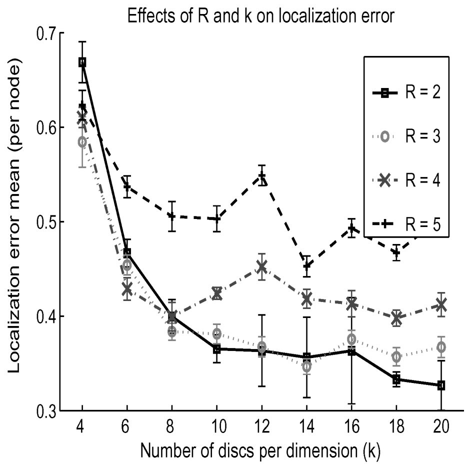
We applied the algorithm described in Section 3 for fine-grained localization. The algorithm involves repeated coarse localizations with respect to a set of regions that cover the whole network area. We choose these regions to be discs of radius R and distributed uniformly over the network. Let k be the number of discs along each dimension such that there are a total k^2 discs to recognize. In this simulation, we study the effects of R, k and the number of base sensors n on the localization performance. Specifically, we examine the trade-off between the computational cost and localization accuracy of our algorithm by varying these parameters, as suggested by the theoretical analysis in Section 3.3.

Effects of n. Figure 5(a) shows that the mean localization error (averaged over all sensor networks and over all sensors) decreases monotonically as more base sensors are added to the network. This agrees with the theoretical result presented in Section 3.3. Figure 7 illustrates the localization results for each node in the networks with 25 and 64 base sensors. The mean localization error (averaging over all sensors) for these two networks are 0.47 and 0.39, respectively.

Effects of R and k. Figure 5(b) shows the effects of R and k on the localization performance. In this set of simulations, we fix $\tau = 0.2, \sigma = 2$, and $n = 100$, while varying R and k. The analysis in Section 3.3 suggests that, for each value of R, there exists an optimal value for k that increases as R decreases. Since there are



(a)



(b)

Fig. 5. The left panel shows the effect of the number of base sensors n on fine-grained localization error mean and standard deviation (for all nodes). The right panel shows the effects of the size of discs (by radius R) and the number of discs (k^2) distributed uniformly on the field. The means and variances are collected after performing the simulation on 20 randomly generated sensor networks.

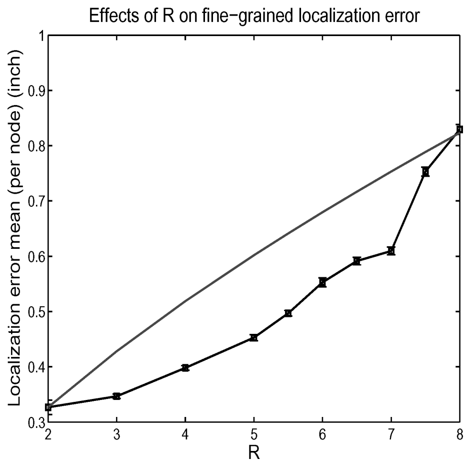
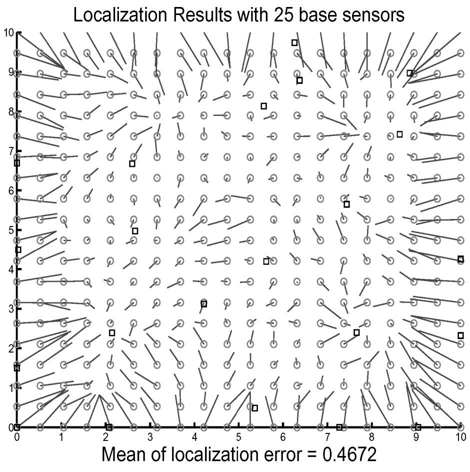


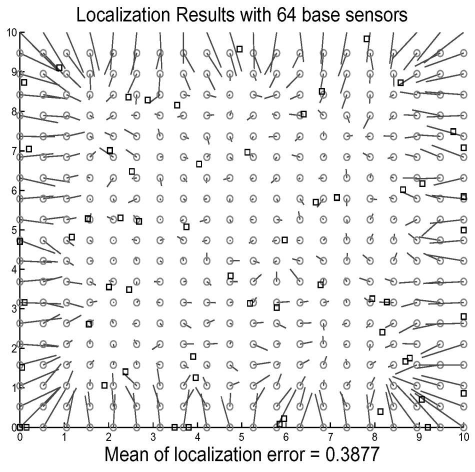
Fig. 6. This figure shows the effects of the size of discs (R) on the fine-grained localization error. The number of disks (k^2) is chosen so that the mean localization error (per node) is smallest. The error rate is compared with the curve $O(R^{2/3})$.

k^2 classification problems to solve, the computational cost generally increases as R decreases. However, the mean localization error improves as R decreases. Hence, there is a trade-off between computational cost and localization accuracy as manifested by the behavior of R and k.

To gain more insight into the effects of the size of discs (R) on the fine-grained localization error, we plot the mean localization error for the optimal value of k in Figure 6. This figure shows that the optimal mean localization error increases as R increases. We also compare the rate of increase with that



(a)



(b)

Fig. 7. Localization results for a simulated sensor network of size 10 × 10 square units with 25 base sensors (left figure) and 64 base sensors (right figure). The base sensors are the black squares. Each line connects a true sensor position (in circle) and its estimate. The signal model is Gaussian. The mean localization error is 0.4672 in the left figure and 0.3877 in the right figure.

of $R^{2/3}$. As shown in Figure 6, the rate is approximately that of $R^{2/3}$ in a middle range and eventually surpasses $R^{2/3}$. Recall from the analysis in Section 3.3 that we expect this increase in rate due to the increase in $\epsilon(R)$. On the other hand, the analysis does not predict the smaller rate of increase observed for small values of R.

5.3 Localization with Berkeley Sensor Motes

Experiment set-up. We evaluated our algorithm on a real sensor network using Berkeley tiny sensor motes (Mica motes) as the base stations. The goal of the experiment is to estimate the positions of light sources, given the light signal strength received by a number of base sensors deployed in the network. Our hardware platform consists of 25 base sensors placed 10 inches apart on a 5×5 grid in a flat indoor environment. Each sensor mote is composed of one Atmel ATmega 103 8-bit processor running at 4MHz, with 128Kb of flash and 4Kb of RAM, RFM TR1000 radio, EEprom and a sensor board that includes light, temperature, microphone sensors, and a sounder. Our experiment makes use of light sensor data received by the motes. The measured signals are a scalar field produced by a light source shining on the sensor network from above; the height and intensity of the light source were constant. Only the position of light sources placed at the base sensors are given as training data. To be estimated are 81 light source positions distributed uniformly in a 9×9 grid, spread over the whole network.

A range-based algorithm. We compared our algorithm to a state-of-the-art algorithm that epitomizes a majority of localization algorithms in the literature. This algorithm was described in Whitehouse [2002] and consists of two main

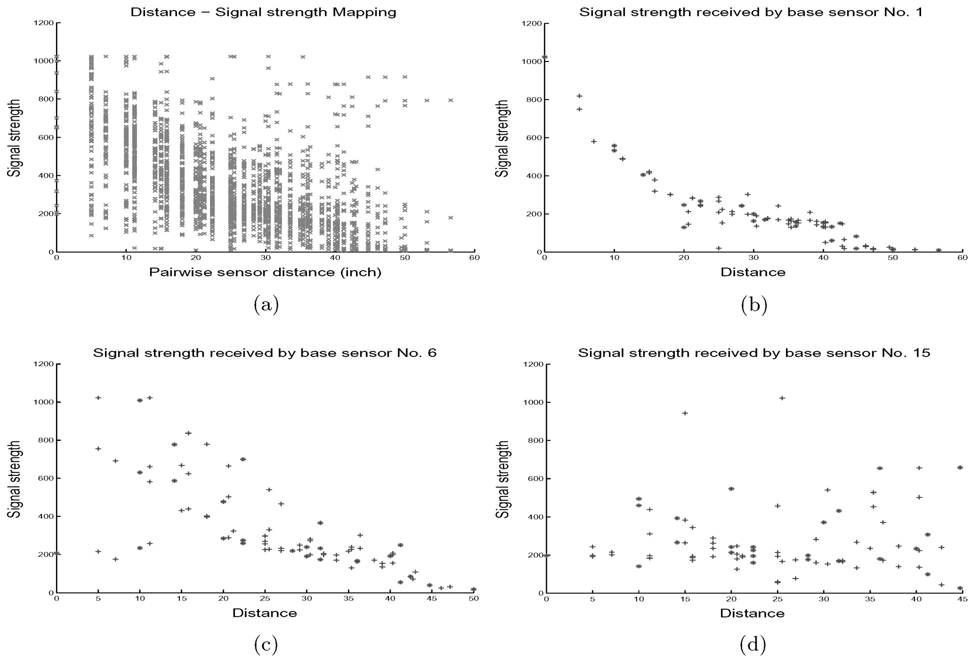


Fig. 8. Panel (a) shows the noisy relationship between signal strength received by sensors and the distances. Few sensors exhibit a clear signal strength-distance functional pattern as in panel (b), while most are like those in panels (c) and (d).

steps: (1) a ranging procedure aimed at establishing a mapping between the signal strength received by a base sensor and the distance to the light source, and (2) a localization procedure giving the distance estimates using least-squares methods.

Figure 8 illustrates the difficulty of the ranging problem—the functional relationship between distances and signal strengths is very noisy. Much of this noise is device-specific. As shown in Figure 8, a few sensors exhibit a clear distance-to-signal-strength pattern, while most others exhibit a very noisy pattern. As presented in Whitehouse [2002], improvement in the ranging step can be achieved by accounting for properties of specific base sensors. This is done by introducing regression coefficients for each of these base sensors. Once the ranging step is completed, we have estimates of the distance between the base sensors and the positions of the light source. The initial position estimates are obtained using the Bounding-Box algorithm and are then iteratively updated using a least-squares method (see Whitehouse [2002]; Savvides et al. [2001]). Figure 9(a) shows the localization results for this algorithm.

Results for the kernel-based algorithm. Three different kernels are used in our algorithm. The first is a first-tier symmetric positive semidefinite approximation of the signal matrix. In particular, as discussed in Section 3, given a signal matrix S, we define $S' := (S + S^T)/2 + \delta I$. The remaining kernels are a second-tier linear and third-tier Gaussian kernel, with the parameter σ

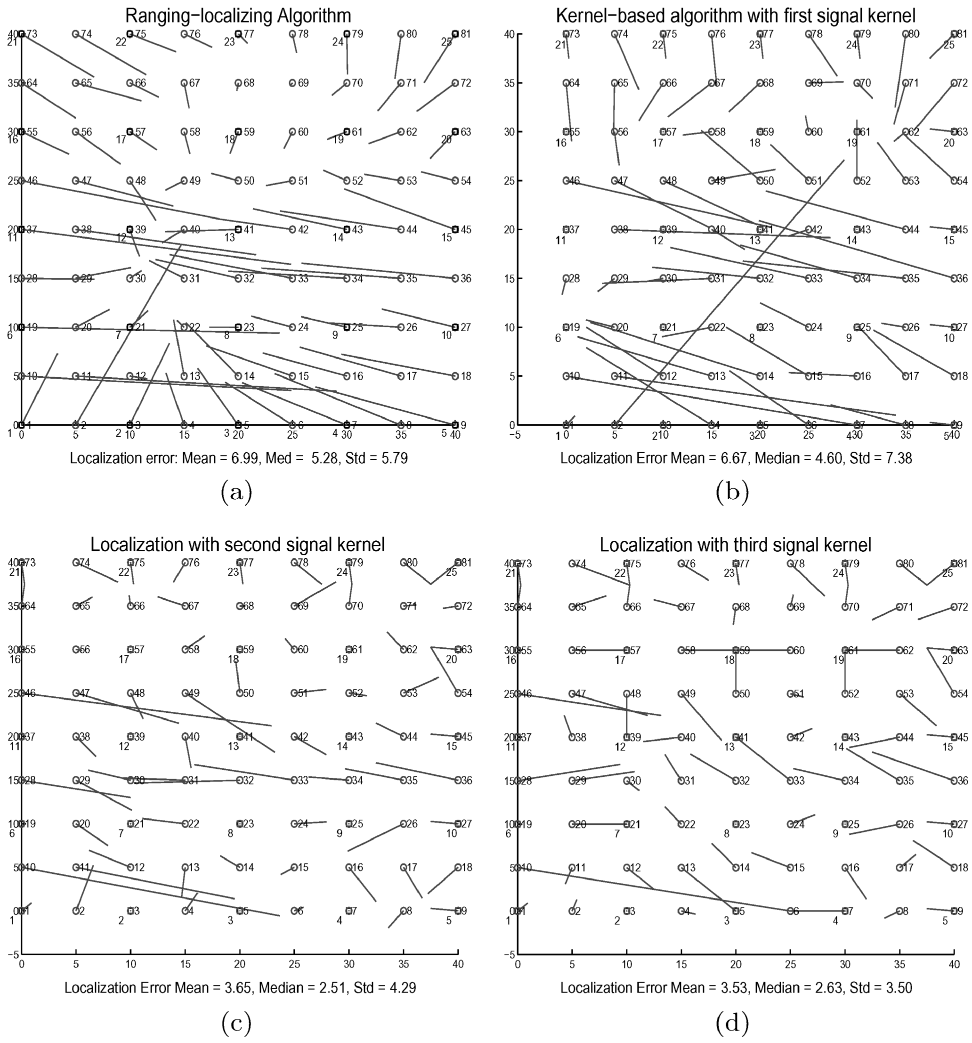


Fig. 9. Localization result for a real sensor network covering a 40 × 40 square-inch area. There are 25 base sensors (Berkeley motes) spaced in a 5 × 5 grid. Each line connects a true position (in circle) and its estimate. Panel (a) shows the results given by a traditional 2-step localization algorithm, while panels (b), (c), (d) show the localization results obtained by our algorithm using three different kernels (the first-tier, second-tier, and third-tier Gaussian kernel, respectively).

fixed to 0.5 in the latter case. For fine-grained localization, coarse localization is repeatedly applied for discs of radius $R = L/2 = 20$ inches that cover part of the network area. The centers of these discs are five inches apart in both dimensions, and there are 10 discs along each dimension (i.e., $k = 10$).

Table I shows that the localization error achieved by the kernel-based approach is smaller than that of the two-step algorithm. Among the three choices of signal kernels, the second-tier kernels are much better than the simple first-tier kernel. The localization results are depicted spatially in Figure 9. Note that

Table I.

Method	Mean	Median	Std
Two-step ranging-based	6.99	5.28	5.79
First-tier signal kernel	6.67	4.60	7.38
Second-tier linear kernel	3.65	2.51	4.29
Third-tier Gaussian kernel	3.53	2.63	3.50

Comparison between a two-step ranging-based algorithm and our kernel-based localization algorithm in a sensor network with 25 base sensors covering a 40×40 square-inch area. The localization error mean, median, and standard deviation are taken over all position estimates and measured in inches.

the minimum distance between two neighboring base sensors is about 10 inches, and the localization error of our algorithm (using second-tier kernels) is slightly over one third of that distance.

6. DISCUSSION

We have presented an algorithm for coarse-grained and fine-grained localization for ad hoc wireless sensor networks. Our approach treats the signal strength as measured by sensor motes as a natural coordinate system in which to deploy statistical classification and regression methods. For the localization problem, this approach avoids the ranging computation, a computation which requires accurate signal models that are difficult to calibrate. Instead, we use signal strength either directly to define basis functions for kernel-based classification algorithms, or indirectly via derived kernels that operate on top of the signal strength measurements. We show how a kernel-based classification algorithm can be invoked multiple times to achieve accurate localization results, and we present an error analysis for the accuracy that can be achieved as a function of base sensor density. Our algorithm is particularly suitable for densely distributed sensor networks and is appealing for its computational scaling in such networks: The preprocessing computations are performed at the base sensors which are assumed to have sufficient processing and power capability, while the localizing step at location-unknown sensors can be achieved in linear time.

We have argued for a simple approach to localization that dispenses with ranging computations and sensor modeling. We do not necessarily believe, however, that our statistical approach is always to be preferred. In particular, the level of accuracy that we appear to be able to obtain with our approach is on the order of one third the distance between the motes. While this accuracy is sufficient for many potential applications of sensor networks, in some applications, higher accuracy may be required. In this case, ranging-based approaches offer an alternative, but only in the setting in which highly accurate models of the relationship between sensor signals and distances are available.

REFERENCES

Bahl, P. and Padmanabhan, V. N. 2000. RADAR: An in-building RF-based user location and tracking system. In *INFOCOM '00 (2)*. 775–784.

Bulusu, N., Heidemann, J., and Estrin, D. 2000. GPS-less low cost outdoor localization for very small devices. Tech. Rep. 00-729, Computer Science Department, University of Southern California.

D'Costa, A. and Sayeed, A. 2003. Collaborative signal processing for distributed classification in sensor networks. In *2nd International Workshop on Information Processing in Sensor Networks (IPSN'03)*. 193–208.

Girod, L. and Estrin, D. 2001. Robust range estimation using acoustic and multimodal sensing. In *IEEE/RSI International Conference on Intelligent Robots and Systems (IROS'01)*.

Hightower, J. and Borriello, G. 2000. Real-time error in location modeling for ubiquitous computing. In *Location Modeling for Ubiquitous Computing—Ubicomp'01 Workshop Proceedings*. 21–27.

Lanckriet, G., Cristianini, N., Ghaoui, L. E., Bartlett, P., and Jordan, M. 2004. Learning the kernel matrix with semi-definite programming. *J. Machine Learn. Res. 5*, 27–72.

Li, D., Wong, K., Hu, Y., and Sayeed, A. 2002. Detection, classification, and tracking of targets. *IEEE Signal Process. Mag. 19*, 17–29.

Poor, H. V. 1994. *An Introduction to Signal Detection, 2nd Ed.* Springer-Verlag.

Priyantha, N., Chakraborty, A., and Balakrishnan, H. 2000. The Cricket location-support system. In *ACM International Conference on Mobile Computing and Networking*. ACM Press, New York.

Savarese, C., Rabaey, J., and Langendoen, K. 2002. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In *USENIX Annual Technical Conference*. Monterey, CA, 317–327.

Savvides, A., Han, C., and Srivastava, M. 2001. Dynamic fine grained localization in ad-hoc sensor networks. In *Proceedings of the 5th International Conference on Mobile Computing and Networking*. 166–179.

Schölkopf, B. and Smola, A. 2002. *Learning with Kernels*. MIT Press, Cambridge, MA.

Seidel, A. and Rappaport, T. 1992. 914MHz path loss prediction models for indoor wireless communications in multi-floored buildings. *IEEE Trans. Antenn. Propag. 40*, 207–217.

Sheng, X. and Hu, Y. 2003. Energy based acoustic source localization. In *2nd International Workshop on Information Processing in Sensor Networks (IPSN'03)*. 285–300.

Vapnik, V. 1998. *Statistical Learning Theory*. John Wiley & Sons, New York.

Want, R., Hopper, A., Falcao, V., and Gibbons, J. 1992. The active badge location system. *ACM Trans. Inform. Syst. 10*, 91–102.

Ward, A., Jones, A., and Hopper, A. 1997. A new location technique for the active office. *IEEE Personn. Comm. 4*, 42–47.

Whitehouse, C. 2002. The design of Calamari: An ad-hoc localization system for sensor networks. MS. thesis, Department of Electrical Engineering and Computer Science, University of California, Berkeley.

Received April 2004; revised August 2004, May 2005; accepted May 2005