Distributed Boundary Estimation using Sensor Networks

Subhasri Duttagupta, Krithi Ramamritham
Dept of Computer Sc. and Engg
Indian Institute of Technology
Mumbai 400076, India
Email: {subhasri, krithi}@cse.iitb.ac.in

Parmesh Ramanathan
Dept of Electrical and Comp. Engg
University of Wisconsin-Madison
Madison, WI 53706, USA
email:parmesh@ece.wisc.edu

***Abstract*— We examine the problem of determining boundaries occurring in natural phenomena using sensor networks. Sensor nodes remotely collect data about various points on the boundary. From this data, we estimate the boundary along with the confidence intervals using a regression relationship among sensor locations and the distances to the boundary. The confidence intervals are guaranteed to be narrower than a specified maximum width. Our distributed boundary estimation strategy uses a hierarchical structure of clusters of sensor nodes and requires $20-50\%$ less messages as compared to a centralized scheme. The computed intervals show desired coverage of the true boundary points. Further, motivated by the practical need to estimate the boundary with a minimum number of sensors, we develop an adaptive approach for turning sensors on and off. The number of ON sensors in this scheme is only about 15% more than what a Practical Oracle needs, to evaluate the boundary and confidence intervals around it. Our algorithms are also evaluated using data from real sensors on a testbed.**

I. Introduction

Sensor networking applications involve estimating spatial variation of a physical field. In many situations, we encounter a field that is composed of two regions, with one having the value of the field parameter less than a $threshold$ and another having the same greater than the $threshold.$ The regions are delineated by a boundary. Examples of applications where such boundaries need to be detected include monitoring the spread of toxic gases [1], oil spills in oceans [2], or even tracking storm cloud front. Oil companies are interested in deploying off-shore sensor networks to monitor the area where oil spills normally happen. The focus in these cases is detecting the spill as well as tracking it. Such an application using sensor networks involves selecting a proper set of data *sources*, gathering latest observations from these sensors, estimating the boundary using a suitable *aggregation* technique, and finally delivering the result to one or more *sink* sensors where users can access the information. Designing such an application has many challenges such as dealing with resource constraints of sensors, estimating boundaries with the desired accuracy, providing a scalable solution etc.

The solution space for the boundary estimation problem can be examined along four orthogonal dimensions: (1) the *characteristic* of the sensors being used - either static or mobile; (2) *sensing capabilities* - either point sensing or range/remote sensing; (3) the *accuracy* of the estimation - either exact or approximate; and (4) the *nature* of the boundary - either static or dynamic. In this paper, we explore the solution space where static boundaries are estimated with bounds on the inaccuracy using static sensors with range/remote sensing capabilities.

One of the initial papers [3] in this area discusses how to estimate a boundary with a desired accuracy provided a large density of static sensors is available. Recognizing that mobile sensors could help in relaxing the requirement of dense deployment, an approach for tracking dynamic boundary using uncoordinated mobile sensors is explored in [4]. Using mobile actuator-sensor networks for tracking the diffusion of a plume is discussed in [1]. But in all these cases, sensors with point-sensing capabilities are being used, which implies that the sensors can detect the boundary only if some of them have physical contact with it. In some applications, for example, detecting the presence of oil slick under the ocean surface or predicting trajectory of weather parameters, it may not be possible for sensors to have physical contact with the boundary. The **non-contact** property of range or remote sensors for example, using radars [5] or laser pulses [6], is useful in these cases. Figure 1 shows an example boundary and the relationship between the positions of sensors and their observations: a sensor at

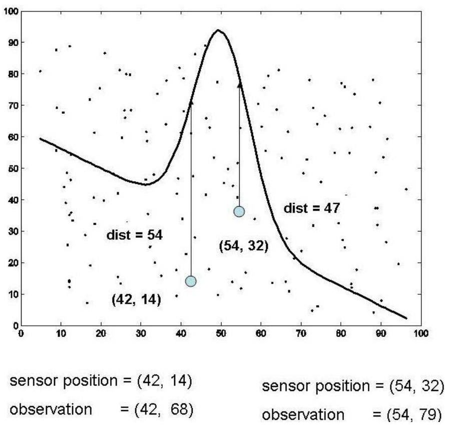


Fig. 1. Boundary Example

position $(42, 14)$ finds that the distance to the boundary is 54 and generates an observation $(42, 68)$. Essentially, we can view the sensor as having a directional antenna facing the y direction.

If we assume that a sensor locates a boundary point having the same value of x and the boundary point depends only on the x coordinate of the sensor, then the relationship between x and y coordinates of the observed point can be modeled as a regression function. Here, the x coordinate of a boundary point is the independent variable, whereas the y coordinate of the same is the value of the dependent variable.

The value of the regression function can be determined in two ways: *parametric* and *non-parametric*. The *parametric* approach assumes that the relationship has a functional form that can be obtained based on a set of observations whereas in the non-parametric approach the relationship function is estimated without reference to a specific functional form of the boundary. Moreover, in both of these techniques the regression function can be computed either in a centralized or a distributed fashion.

The contributions of this paper lie in addressing the following issues connected with the problem of boundary estimation using data from distributed sensor sources:

1) Given erroneous observations from sensors how can we estimate the boundary?
   We use a non-parametric regression-based approach that estimates the boundary without explicitly referring to its functional form, using a network of sensors capable of range sensing.
2) Can we associate confidence intervals with the estimated boundary?
   Our estimation technique provides confidence intervals around the estimated boundary and ensures that the width of the intervals is less than a specific bound δ.
3) How do observations from sensors get disseminated to node(s) estimating the boundary?
   The novelty of a regression-based approach is that it is amenable to distributed evaluation of the boundary and the confidence intervals (CIs) using an in-network aggregation strategy that needs $20-50\%$ less messages as compared to a centralized scheme.
4) Can we obtain CIs having a desired maximum width of δ at user-specified locations, using an optimal number of deployed sensors?
   We present an efficient distributed algorithm for estimating the CIs with bounded width δ at a set of selected locations. This method uses only about 15% more sensors than a *Practical* Oracle would require. We also extend our algorithm to a more general solution where it is ensured that CIs around the estimated boundary at *any* location has width less than a specific bound.

The developed algorithms are evaluated using an implementation on the TOSSIM simulator and data from real sensors on a testbed.

The paper is organized as follows. Section II provides the problem definition and the basic solution approach. Section III explains our proposed distributed solution strategy and Section IV provides an approach for estimating the boundary using a minimal number of sensors. The experimental results presented in Section V show the effectiveness of our technique. Section VI concludes the paper.

II. Problem Formulation and Approach

We assume that sensor nodes are distributed over a two dimensional field measuring a field parameter and they are aware of their physical locations using, for example, a GPS enabled antenna or local positioning system. Every sensor is directional along the y axis and based on its position, determines the y coordinate of the first point where the field parameter meets the definition of the boundary as specified by a *threshold*. Further, we assume that the observation of i^{th} sensor consists of (x_i, y_i) where x_i is also the x coordinate of the position of the sensor.[1] Given n such sensor observations with errors, our objective is to find a technique to evaluate the boundary at arbitrary x values with a certain accuracy. In this paper, we have addressed the problem of computing pointwise confidence intervals (p_j, L_j) and (p_j, U_j)

[1]y_i is the y coordinate of the measured boundary point's location and not the relative distance from the sensor

which are respectively the lower and upper portions of the intervals for a desired confidence $(100-\epsilon)\%$ around the estimated boundary at p_j.

We model the relationship between x_i and y_i as a nonparametric regression. With n observations, the regression relationship can be written as

$$y_i = m(x_i) + \alpha_i, \quad i = 1, \dots, n \tag{1}$$

where m is the regression relationship between x_i and y_i, and α_i are the observation errors. If the errors have mean zero distribution, then the mean value of the distance to the boundary at x_i is $m(x_i)$. Assuming the boundary to be smooth, the observations at points close to x contain information about the value of m at x. Thus, intuitively it is possible to use a local average of the data near x to construct an estimate for $m(x)$. We define a set of weights such that $m(x)$ is estimated using a weighted average of the observations within a small neighborhood of x. The basic step involved in the estimation of $m(x)$ could be as follows:

$$\hat{m}(x) = \frac{1}{n}\sum_{i=1}^{n} W_i(x) y_i \tag{2}$$

where $\{W_i(x)\}_{i=1}^{n}$ denotes a sequence of weights that may depend on all the observations at $\{x_i\}_{i=1}^{n}$. We use $\hat{m}(x)$ to denote the estimation of $m(x)$. The technique of *kernel smoothing* [7] is useful in this respect where the shape of the weight function W_i is defined using a function called *kernel* K. Typical kernel functions are zero outside some range which implies the weight sequence is non-zero only for observations within a small neighborhood around x. This neighborhood is referred to as *h-neighborhood* of x and the parameter h is called the *bandwidth* in the field of nonparametric regression.

The weight sequence proposed by *Nadaraya-Watson estimator* [8] is defined as $W_i(x) = K(\frac{x-x_i}{h})/\hat{f}(x)$ where,

$$\hat{f}(x) = \frac{1}{n}\sum_{i=1}^{n} K(\frac{x-x_i}{h}) \tag{3}$$

The numerator of weight sequence is a kernel function and denominator of the weight sequence is $\hat{f}(x)$ which is the kernel density estimator of x. It serves two purposes: (1) the weights adapt to the local density of x (2) it satisfies that the weights sum to one. The expression for $\hat{m}(x)$ after plugging in the weight sequence in Equation (2) becomes,

$$\hat{m}(x) = \frac{\frac{1}{n}\sum_{i=1}^{n} K_h(x-x_i) y_i}{\frac{1}{n}\sum_{i=1}^{n} K_h(x-x_i)} \tag{4}$$

Here $K_h(u) = K(\frac{u}{h})$ is used for simplification.

Example: Let (x_i, y_i) values for four observations be $(10,13), (11,15), (12,18), (13,20)$. Suppose we want to estimate y value for $x = 12.75$.

If we take $h = 1.25$; and a simple kernel $K(u) = .75(1-u^2)I(|u| \leq 1)$, then the weight sequences for estimating $\hat{m}(12.75)$ are: $W_1(12.75) = 0; W_4(12.75) = 0;$
$\hat{f}(12.75) = (.64 + .96)/4 = .4$
$W_2(12.75) = .64/.4 = 1.6$; $W_3(12.75) = .96/.4 = 2.4$;
$\hat{m}(12.75) = (1.6 \times 18 + 2.4 \times 20)/4 = 19.2$

The computation of the CIs around the estimated $\hat{m}(x)$ requires the knowledge of how far this estimation is from the true $m(x)$. From the asymptotic normal distribution of $|m(x) - \hat{m}(x)|$ based on a theorem from [9], the upper and lower CIs $U(x)$, $L(x)$ around $\hat{m}(x)$ for a specific confidence $(100-\epsilon)\%$ can be obtained as,

$$\begin{aligned} U(x) &= \hat{m}(x) + \frac{c_\epsilon c_K^{1/2}\hat{\sigma}(x)}{(nh\hat{f}(x))^{1/2}} \\ L(x) &= \hat{m}(x) - \frac{c_\epsilon c_K^{1/2}\hat{\sigma}(x)}{(nh\hat{f}(x))^{1/2}} \end{aligned} \tag{5}$$

where $\hat{\sigma}^2(x)$ is the estimated error variance at x, c_ϵ is the $(100-\epsilon)$-quantile of a normal distribution and c_K is a kernel constant. The bias term is ignored while obtaining the above equations from the normal distribution.

The error variance for a given value of x can be calculated using conditional variance of y,

$$\sigma^2(x) = E(y^2|x) - m^2(x) \tag{6}$$

$$\hat{\sigma}^2(x) = \frac{1}{n}\sum_{i=1}^{n} W_i(x) y_i^2 - \hat{m}^2(x) \tag{7}$$

where $\hat{\sigma}(x)$ is the estimate of $\sigma(x)$.

The *coverage* of the CIs is formally defined here. Suppose all sensors make $\mathcal{N}$ instances of observations and CIs at x are estimated in every instance. In η out of those instances, the following condition is true: $L_i(x) \leq m(x) \leq U_i(x)$ that is, the actual boundary is within the estimated CIs. Then,

$$coverage(x) = \frac{\eta}{\mathcal{N}} \times 100 \tag{8}$$

Next, the relevant question is: how do we evaluate the regression relationship and the conditional variance in a sensor network. The expressions for $\hat{m}(x)$ and $\hat{\sigma}^2(x)$ reflect that they are aggregate functions of sensor observations. Hence, we need a suitable aggregation technique to realize these expressions from sensor observations. This is addressed in the following section.

III. Distributed Boundary Estimation Strategy

In this section, we discuss how observations from the nodes are used to estimate the boundary points and their conditional variances in a distributed fashion. This involves (i) deciding where the data aggregation should take place, (ii) how the observations from individual nodes reach the nodes that perform such aggregation, and (iii) what computation should take place at these nodes. There are two basic approaches for disseminating the data in sensor networks. In a centralized solution, all the observations are sent to a base station that performs the computation. This solution is not scalable as the total number of messages sent by all nodes increases with the size of the network. Conversely, in a distributed solution, the observations can be combined through in-network aggregation and the computations can be done in an incremental fashion at intermediate nodes.

For efficient data aggregation, hierarchical clustering is preferable as it helps in reducing *energy* consumption and in efficient resource utilization (number of nodes being used) [10]. So, we explore a cluster-based data dissemination technique which is discussed in Section III-A. For a distributed implementation, Equations (2) and (7) are amenable to being split into expressions that can be evaluated at the cluster heads (CHs) in the routing tree. This is described in sections III-B and III-C. The evaluation of the proposed approach is done in section III-D.

A. Routing using Clusters

How the nodes in the network disseminate information using clusters is discussed here. Initially, the sensors are organized into clusters and cluster heads (CH) are selected. We require the nodes within a cluster to be one or two hops away from the CH so that the observations from individual nodes can easily be aggregated at the CH. In our case, different CHs may contribute to the partial aggregates relating to different boundary points. So it is not clear whether a general clustering scheme with the goal of minimizing the energy will meet our needs. Given this consideration, we use a cluster formation algorithm suggested in [11] as it has the characteristic that the nodes within a cluster are within the communication range of the CH and it generates an optimal number of clusters. We assume that there is a base station (BS) that is always a CH. All other CHs form a routing tree structure rooted at BS. This implies that every CH has a parent node through which it can send data to a CH closer to BS.

In the routing tree, each leaf node sends its observation to the respective CH. Non-leaf nodes and CHs wait for their child nodes to send their data before forwarding the aggregated data to their respective parent nodes.

B. Computation of Distance to the Boundary

Since the regression function, $\hat{m}(p_j)$ and the conditional variance $\hat{\sigma}^2(p_j)$ are duplicate-sensitive aggregates we require that observation from any node is included in the computation at-most once. This is ensured by the cluster-based routing scheme used here. The technique for computing the regression function at p_j is as follows. The observation from a node with x_i coordinate is used for computing boundary point $\hat{m}(p_j)$ if it satisfies the relationship $p_j - h \leq x_i \leq p_j + h$. The expression for $\hat{m}(p_j)$ in (4) can be written in terms of a fraction $\frac{N}{D}$ where numerator N is the sum of individual observations multiplied by the kernel at x_is and denominator D is the sum of kernel values evaluated at x_is in the range $[p_j - h, p_j + h]$.

If all the leaf nodes within the neighborhood of p_j send their observations to two CHs and sets of x_is of nodes within these clusters are: $S_1 = \{x_i | i = 1, \ldots, M\}$ and $S_2 = \{x_i | i = M+1, \ldots, n\}$ where their union is denoted by $S = S_1 \cup S_2$. Then $\hat{m}(p_j)$ can be evaluated as follows:

$$\begin{aligned} \hat{m}(p_j) &= \frac{\sum_{i \epsilon S} K_h(p_j - x_i) y_i}{\sum_{i \epsilon S} K_h(x - x_i)} \\ &= \frac{\sum_{i \epsilon S_1 \cup S_2} K_h(p_j - x_i) y_i}{\sum_{i \epsilon S_1 \cup S_2} K_h(p_j - x_i)} \\ &= \frac{N_1(p_j) + N_2(p_j)}{D_1(p_j) + D_2(p_j)} \end{aligned} \quad (9)$$

where $N_l(p_j), D_l(p_i)$ for $l = 1, 2$ are as given below:

$$\begin{aligned} N_l(p_j) &= \sum_{i \epsilon S_l} K_h(p_j - x_i) y_i \\ D_l(p_j) &= \sum_{i \epsilon S_l} K_h(p_j - x_i) \end{aligned} \quad (10)$$

This shows that multiple CHs can compute N_i, D_i terms and these *partial aggregates* (PA) for $\hat{m}(p_j)$ can be combined using the above strategy to obtain a new PA and sent through the routing tree to BS. A CH calculates the PA for $\hat{m}(p_j)$ only if it has at least one observation within its cluster for computing this.

C. Computation of Conditional Variance

Using a similar technique used in the previous section, multiple nodes in the network can compute the conditional variance in a distributed fashion. The computation

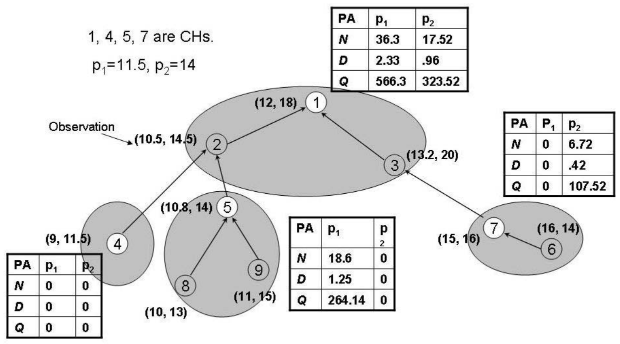


Fig. 2. CHs computing $\hat{m}$ and $\hat{\sigma}^2$ for $p_1 = 11.5$ and $p_2 = 14$

of variance as in (7) involves taking the square of observations from the individual nodes and using the value of final $\hat{m}(p_j)$. Again, if S is the set of relevant observations available at two CHs, then the variance of a boundary point is computed using PAs from these two nodes using the following approach:

$$\begin{aligned}\hat{\sigma}^2(p_j) &= \frac{\sum_{i\epsilon S} K_h(p_j - x_i)y_i^2}{\sum_{i\epsilon S} K_h(x - x_i)} - \hat{m}^2(p_j) \\ &= \frac{Q_1(p_j) + Q_2(p_j)}{D_1(p_j) + D_2(p_j)} - \hat{m}^2(p_j) \qquad (11)\end{aligned}$$

where $D_1(p_j)$ and $D_2(p_j)$ are provided by Equation (10) and $Q_l(p_j)$ for $l = 1, 2$ is given as:

$$Q_l(p_j) = \sum_{i\epsilon S_l} K((p_j - x_i)/h)y_i^2 \qquad (12)$$

Thus, many nodes in a sensor network can compute PAs that contain (N_i, D_i, Q_i) terms for $\hat{m}(p_j)$ and $\hat{\sigma}^2(p_j)$ in a distributed manner using the above aggregation technique.

A crucial step in estimating the true boundary is choosing the parameter h that controls how much of the neighborhood around p_j has to be considered in estimating the boundary at p_j. We used an iterative *plug-in* approach [12] for finding the optimal h that minimizes the expression for the *average square error* (ASE). It requires estimating $\hat{m}''(x)$ the second derivative of the regression curve, using the kernel method. Due to lack of space, we have not provided the details of this technique in this paper.

An example of computing PA at CHs for two points $p_1 = 11.5$ and $p_2 = 14$ using $h = 1.5$ is shown in Figure 2. After the information is sent to Node 1, it estimates $\hat{m}(11.5) = 15.5$ and $\hat{\sigma}_2(11.5) = 2.8$, $\hat{m}(14) = 18.3$ and $\hat{\sigma}_2(14) = 3.9$.

D. *Evaluation of the Basic Approach*

Our proposed technique is evaluated through experiments to verify whether it provides an effective mechanism for estimating a physical boundary using sensor networks. We experiment with boundaries of two types of shape - regular and irregular. Regular shapes are generated using a few mathematical functions. Boundaries of arbitrary shapes are generated using real temperature values from sensors over a time period. From such a temperature trace, the time axis is mapped to the x values of sensors between $(0, 100)$ and the observations are obtained from the temperature at corresponding x values. A network of 120 sensors is used to estimate CIs around a boundary at different values of x. Figure 3(a) and Figure 3(b) illustrate a smooth boundary and a non-smooth boundary respectively computed using our technique. Here the x axis represents x values of the field and y axis shows $\hat{m}(x)$ and the CIs around it. Note that most of the portions of the true boundary lie within the CIs. The width of CIs is proportional to the confidence level as well as to the error variance of observations as expected from Equation (5). We observe the effect of varying confidence level $(100 - \epsilon)\%$ in Figure 3(a) and the effect of varying the error variance σ^2 in Figure 3(b). In experimental evaluations, we see that these confidence intervals also show good coverage of the actual boundary.

IV. Optimizing Sensor Usage

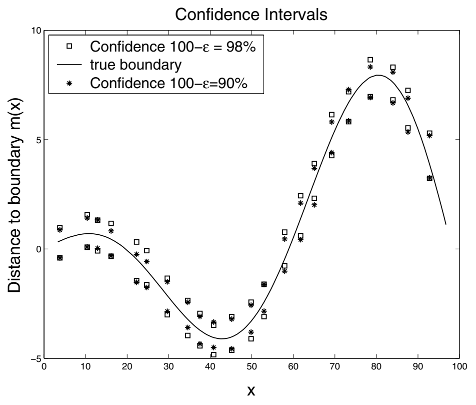
In this section, we extend our algorithm such that the confidence intervals (U_j, p_j) and (L_j, p_j) at k values $p_1, \ldots, p_k$ satisfy the following conditions:

1) the true boundary lies within the above interval with confidence $(100 - \epsilon)\%$, and
2) $max|U_j - L_j| \leq \delta$

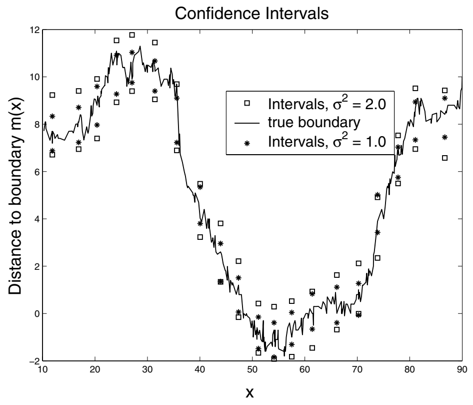
here δ and ϵ are specified by the application.

In addition, with the objective of tracking a boundary for the longest possible time, when the CIs are being estimated at many locations, they should be done using a minimal set of ON sensors. If a sensor's observation is not required for boundary estimation, the node is turned off.

We extend our cluster-based evaluation scheme discussed in the previous section to address the following issues: (i) if CIs have width much lower than δ, it may be computed with less number of observations, hence, some sensors can be turned off, (ii) if the width of a CI is more than δ, some additional sensors in the neighborhood have to be turned on. (iii) if sensors are contributing to



(a) CIs for smooth boundary, $\sigma^2 = 1.0$



(b) CIs for non-smooth boundary, $100 - \epsilon = 98\%$

Fig. 3. Estimated Confidence Intervals

more than one boundary point, some optimization of the number of ON sensors could be obtained while satisfying δ criterion for a set of points. Clustering of sensors helps in deciding the ON/OFF status of neighboring sensors within a cluster.

A. *Selecting a Minimal set of ON sensors*

For simplicity, we assume that the errors in observations as given in Equation (1) have identical distributions. Since $\hat{m}()$ is the weighted average of all observations, using the statistical property of variance of an average, the width of CIs around $\hat{m}()$ reduces with increase in the number of observations. The challenge here is to select appropriate number of sensors for turning on (and turning off) so that all the intervals meet the width criterion. Since sensors in *h-neighborhood* of a particular boundary point may be spatially apart and may belong to different clusters, each CH has to make its own decision locally on how many sensors to turn on(off) and yet globally minimize the total number of ON sensors.

We use an *iterative* approach for turning on(off) nodes. Initially, all the CHs send the PAs to the BS but in subsequent iterations, only the PAs that have changed due to turning on/off nodes are sent. In every iteration, BS recomputes the confidence intervals and communicates the information about the width of the intervals (whether greater than or less than δ) to individual CHs using an efficient bitmap-based technique. Different bitmaps are used to indicate the distance between the actual width and δ. The main components of our solution for obtaining a minimal set of ON nodes are as follows:

- There should be a route from any node to the BS using only currently ON nodes. This is achieved by not turning off any non-leaf CHs and the intermediate nodes used for routing.
- If an OFF node contributes to the computation of a large number of boundary points for which the width is $> \delta$, then it should be turned on. This is achieved by each CH keeping a *score* list for all the nodes in its cluster where the *score* of a node indicates the number of boundary points having x in the h-neighborhood that have not met the δ limit.
- An ON node is selected for turning off, if all the boundary points in its *h-neighborhood* have width $< \delta$ by a certain amount.
- To reduce the communication overhead, the CHs closer to BS initiate the process for turning on followed by the CHs farther from BS.
- In every iteration, based on the information sent by BS and local *score* list, each CH decides to turn on additional sensors or turn OFF redundant sensors.

The iterative process converges when the following condition is satisfied: if there is some p_j for which the width criterion is not met, then there should be no OFF sensors within h distance from p_j. Note, this does not necessarily imply that all the intervals are computed with the minimum number of ON sensors. However, experimental evaluations indicate that the proposed solution does indeed optimize the number of ON sensors.

B. *Satisfy δ Criterion for CIs at every x*

What should be the k points such that if the width of CIs around the boundary at these points are $\leq \delta$, then the same would be true for CIs estimated at *any* x?

The ON/OFF status of the sensors in the network is decided while estimating CIs at k locations. So the value of k and location of these k points should be chosen in such a way that would allow us to ensure that the CIs at any x meet the δ criterion. This requirement, if satisfied, would provide a *band* around the estimated boundary with width $\leq \delta$ such that at any point, the boundary is covered by the band with confidence $(100 - \epsilon)\%$.

The width of CIs at a particular x as given in (5) is inversely proportional to square root of the density of sensors and proportional to the variability of the boundary. So the CIs at a particular x would be less than δ if we ensure that the estimated density of sensors at x is more than a lower bound and the conditional variance at x is less than an upper bound. The variance at x is nothing but the sensor noise variance. Assuming that the noise variance is upper bounded by the maximum value of the same at any of the k points , we focus on satisfying the condition on density.

The density depends on a specific deployment and availability of sensors. In addition, the technique of selecting nodes for turning on/off affects the density of sensors. An estimation of density as given in Equation (3) takes into account all ON nodes within h distance. This indicates that if we have at least one boundary point at every h distance for which the δ criterion is met, the local density of sensors is expected to meet a lower bound at any intermediate points. It suggests that a choice for k as $\lceil X_{range}/h \rceil$ where X_{range} gives the range of x values of the deployed sensors. Using this approach, we estimate the boundary at equi-distant k values.

Through experimental results, we also verify our claim that if sufficiently large number of boundary points are estimated, the CI at any arbitrary x meets the δ criterion. This is verified by estimating CIs at a large set (> 200) of equi-distant x values. We observe that as k increases, the width criterion is satisfied by more points of the boundary. When $k > \lceil X_{range}/h \rceil$, the criterion is satisfied by about 99% points.

V. Simulation Results

Here we look at the performance of the distributed boundary estimation technique presented above, showing that it provides an effective technique for estimating boundaries of arbitrary shapes. We compare our scheme with a centralized one in terms of number of messages. The number of (ON) sensors required in our technique is compared using two standard techniques - Simulated Annealing and a *Practical* Oracle. In addition, we conduct experiments for choosing suitable values for parameters like δ and k. We also evaluate the strategy with real sensor data on a testbed.

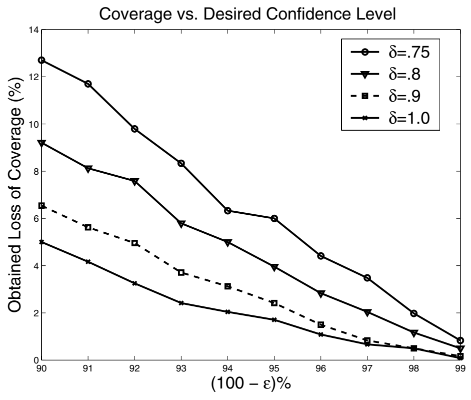
We implement our distributed cluster-based algorithm in MATLAB as well as in TOSSIM [13], the TinyOS simulator for *motes*. Our entire implementation including the estimation of boundary points and turning on/off nodes is done in TOSSIM but as it is time consuming, we use the MATLAB simulation for generating Figure 4.

We consider a field where sensors are deployed in a two-dimensional $100m \times 100m$ square grid. Their locations are obtained from a random distribution over the area. We assume error-free communication but the simulator includes the possibility of loss of messages due to collisions. We assume that a message can contain only one sensor observation and one PA. We take the errors in sensor observations from a normal distribution $N(0, \sigma^2)$, where σ^2, the error variance, is varied between .5 and 2.0. The number of boundary points, k is varied between 10 and 50.

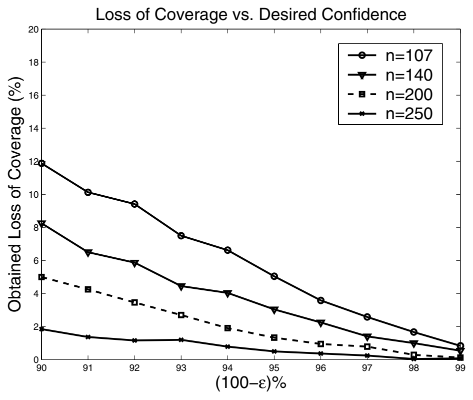
Metric: The metrics we consider are the *accuracy* in estimation, the communication overhead and the number of ON sensors. Accuracy is expressed in terms of *coverage* of the actual boundary points by the CIs as given in (8). Communication overhead is measured by the total number of messages used in estimating a boundary point and its CIs. The other metric is the total number of ON sensors used in obtaining CI for meeting δ requirement at a selected set of points.

A. *Coverage of Intervals increases with δ and ON nodes*

The goal of this experiment is to measure the accuracy of the CIs estimated using the proposed distributed strategy. The coverage score of our algorithm is calculated with respect to k points $p_1, \ldots, p_k$. We generate 100 datasets for these k points and we take $\sigma^2 = 1.0$. The average loss of coverage (in percent) resulting from our algorithm vs. the desired confidence $(100-\epsilon)\%$ is shown in Figure 4. The width of CIs would vary based on the density of ON sensors and the specified δ. To show only the impact of δ on the coverage, we perform the first set of experiments with a fixed number of ON nodes. The δ mentioned here is the value used for 99% confidence level. For lower confidence, the δ value is reduced in proportion to the area of normal distribution. Figure 4(a) shows that when CIs are wider, the coverage score is better. As the width of CIs increases, the probability that the actual boundary point is covered by the intervals increases. In the next set of experiments, we vary the density of ON nodes by keeping δ fixed. From Figure 4(b) we notice that the coverage score improves with an increase in the number of nodes because with more observations, the estimated boundary point converges to the actual boundary point. From this experiment, we conclude that δ should be chosen higher than the sensor error variance and we require > 100 nodes to get a good



(a) Variation of δ, n= 200



(b) Variation of n, $\delta = 1.0$

Fig. 4. Loss of Coverage vs. Desired Confidence $(100 - \epsilon)\%$

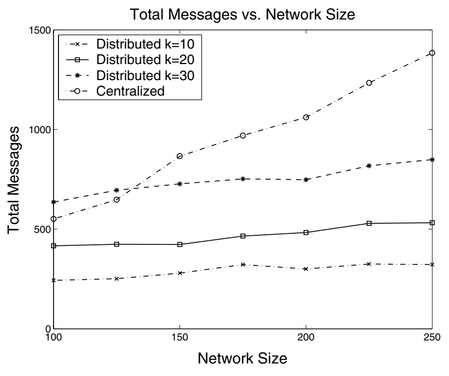


Fig. 5. Messages vs. Nodes

coverage score.

B. Distributed Estimation is Superior to Centralized

The communication overhead of the proposed distributed solution is compared with a scheme where all the observations are sent to a central site and the computation takes place there. It is worth mentioning here that there is no loss of coverage from the centralized solution to the distributed solution as similar computation is done in both the cases. In Figure 5, we compare the number of messages in these two schemes for different values of k keeping the communication range as $10m$. The x axis is the size of the network and the y axis is the total number of messages. For this experiment, we assume all the nodes in the network are turned on and total messages in the distributed version does not include messages required for computing the optimal h parameter assuming this is done only once in the initial phase. We observe that the number of messages increases as k increases. But we notice that it does not change much with the increase in network size. This can be explained as follows.

The total messages include messages for gathering observations by the CHs and then sending the aggregates to the base station. The number of nodes providing observations for a specific boundary point around its *h-neighborhood* is approximately $2nh$ where n is the total number of ON nodes. Here h is expressed as a fraction of total range of x values. Since an increase in n (effectively the node density) leads to a reduction in the optimal value of h, the first contributor of the messages does not increase in proportion with n. The second contributor for messages depends on the levels of the routing tree which is not affected by node density. Thus, the distributed scheme offers a scalable solution overall. As the aggregates for each boundary point is routed to BS separately, the number of messages increases in proportion with the value of k. Our algorithm uses $20 - 50\%$ less messages as compared to a centralized approach for a network size over 100.

C. Comparing Algorithms for using a minimal set of ON sensors

We like to verify whether the total number of ON sensors in our approach after turning off redundant ones, actually forms a minimum set of sensors required to satisfy δ criteria at all k points. So we compare number of ON sensors in our approach with the same required by a Simulated Annealing (SA) technique and by a *Practical* Oracle (PO) that is aware of the network topology.

The SA technique is a standard approach that performs a random search in the solution space based on a cost function. In this case, the cost function is proportional to the number of ON nodes and the number of p_j values for which the width criterion is not met. Hence, SA reaches the optimum state having the lowest cost when

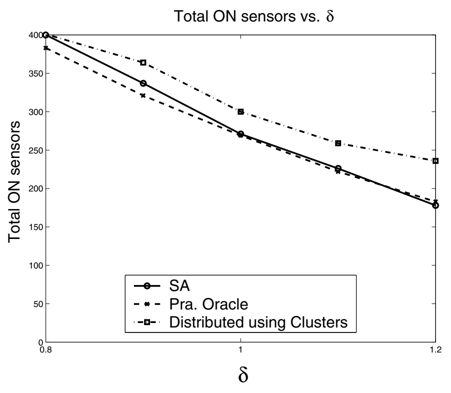


Fig. 6. Total ON sensors vs Max width of Confidence Intervals

the maximum number of boundary points meet the δ width requirement with the minimum number of ON sensors. Since a state in SA is defined by ON/OFF state of all the nodes, the state with the lowest cost is expected to provide the optimal combination of ON nodes to meet the δ criterion.

But SA is unaware of the neighborhood information. It does not use the fact that nodes contributing to a number of boundary points, if turned on, can help in meeting the width requirement of CIs of multiple points. Hence, we implement a heuristic-based Oracle program that obtains the network topology from the implementation in TinyOS. The Oracle also has global information about the observations of all sensors and their ON (OFF) state. It makes use of a *score* list of all OFF sensor nodes indicating their impact on the computed intervals and at each step, a node with the highest score is selected for turning on. Thus, the Oracle meets the width criterion on all the CIs by turning on a minimal set of nodes.

Figure 6 shows the total number of ON sensors of these three algorithms vs. the value of δ. The initial number of nodes is 500 and the error variance is $\sigma^2 = 1.0$. We observe that PO and SA need around the same number of nodes. However, our scheme needs $5-15\%$ more ON nodes than PO. We also notice that if the bound is tight i.e., the value of δ is less than σ^2, the difference between PO and our approach is low. Besides the comparison, this figure also shows the basic *energy-accuracy* trade-off in our approach. Though we have not explicitly derived the expression for the energy consumption in this paper, the number of ON sensors is an indicator of energy consumption. For a specific error variance, the narrower the CIs needed, the larger the number of sensors that have to be turned on. As more sensor observations are gathered by the CHs, the total energy consumption also goes up.

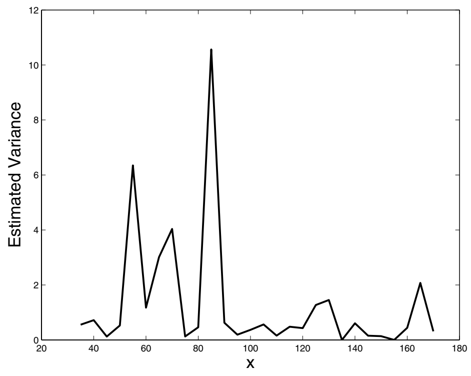
D. Experiment with Real sensors

To verify how our proposed strategy works with real sensors, we use a robot equipped with linear infrared distance measuring sensors and a boundary is created using a wall-like obstacle of non-linear shape. The robot moves along a defined path and distances to the target are measured. The actual distances to the boundary are also taken for reference. Using this dataset in our experimental setup, the boundary is estimated. We observe 65% coverage of the actual boundary when the number of observations is less than 50 and but it improves to 91% when the number of observations is increased to more than 100. In both the cases the observations are taken over the same range of x values. Figure 7(a) shows the estimated variance for different x values. We notice that the variance is $1.0-4.0$ for most of the values of x. This suggests that the value of δ should be chosen > 2.0. The variance is higher when the rate of change of the boundary with respect to the x coordinate is high. In Figure 7(b), the estimated CIs for $\delta = 3.0$ are shown. We observe that the actual boundary is covered by the CIs if δ is chosen suitably.

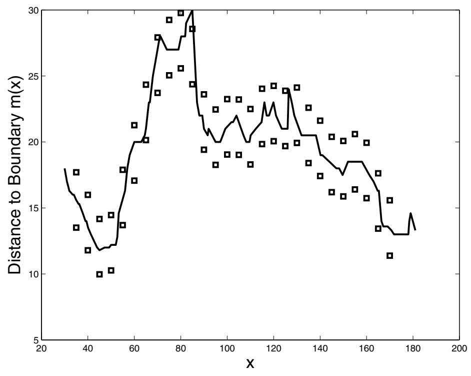
In summary, the experimental results reveal the following and thereby show the practicality of our approach for boundary estimation: 1) the computed intervals cover the actual boundary points with the desired confidence level, 2) the technique offers high fidelity if the number of boundary points is higher than the suggested k value. 3) our distributed solution uses $20-50\%$ less messages as compared to a centralized scheme, 4) the efficiency of the technique for using a minimal number of ON sensors is comparable with Simulated Annealing and Oracle approaches, 5) it allows a trade-off between energy consumption and accuracy in estimating boundaries using sensor networks.

VI. Conclusion and Future Work

We have developed a technique for boundary estimation in sensor networks where observations from sensors are aggregated and confidence intervals around the true boundary are obtained for a set of points. This strategy allows estimation of the boundary with a desired accuracy. We have provided a distributed strategy for realizing the non-parametric regression relationship in sensor networks. We have also tackled the problem of satisfying accuracy constraints in terms of the width of CIs, using optimal number of ON sensors. Moreover, our proposed distributed strategy uses significantly less number of messages as compared to a centralized solution for estimating CIs at a set of boundary points.



(a) Estimated variance vs. x



(b) CIs with $100 - \epsilon = 98\%$

Fig. 7. Experimental results from a testbed

Even though the mathematical model and data dissemination technique described here are for a specific problem of boundary estimation, the solution approach can be used for any other generic problem where erroneous observations are being used to obtain a better estimate of a physical field parameter using sensor networks. For example, the solution can be used to answer the following question. What is the average temperature in a specific region X? This can be done by gathering observations within an optimally chosen small neighborhood of the region X and computing the weighted average of these observations.

We now compare our approach qualitatively with the parametric technique for polynomial regression model described in [14]. The non-parametric approach assumes the observations are erroneous and it tries to reduce the error by aggregation. In the parametric case, the observations are taken to be the actual values of the sensed quantity and the coefficients of basis functions are computed in order to minimize the root mean square (RMS) between the observations and values of the actual regression function at sensor locations. As part of our future work, we would like to find out the ranges on observation errors for which these two methods give comparable results.

Our current strategy is being extended to the case where the boundary changes with time. We include a time parameter t and the requirement is that the estimated boundary point lie within the estimated confidence interval for all time t. The main challenge in tracking a dynamic boundary is to update the CIs with minimal communication overhead. Moreover, we would explore real-time processing of the distributed aggregation strategy outlined here to ensure that the estimation of boundary is completed before a deadline.

References

[1] K. Moore, Y. Chen, and Z. Song, "Diffusion-based path planning in mobile actuator-sensor networks (mas-net): Some preliminary results," in *Intelligent Computing: Theory and Application II. SPIE Defense and Security Symposium*, 2004.

[2] M. F. Fingas and C. E. Brown, "Review of Oil Spill Remote Sensing," in *Eighth Int. Oil Spill Conference, SPILLCON* , 2000.

[3] R. Nowak, U. Mitra, and R. Willett, "Estimating inhomogeneous fields using wireless sensor networks," *IEEE Journal on Selected Areas in Communications, vol. 22, no. 6, pp. 999-1006*, 2004.

[4] K. Wang and P. Ramanathan, "Collaborative sensing using sensors of uncoordinated mobility," in *Intl. Conference on Distributed Computing in Sensor Systems*, June 2005.

[5] J. Curlander and R. McDonough, *Synthetic Aperture Radar: Systems and Signal Processing*. John Wiley and Sons, Inc., 1991.

[6] S. Babichenko, A. Leeben, and L. Poryvkina, "Laser remote sensing of coastal and terrestrial pollution by fls-lidar," in *EARSeL eProceedings, 3(1):1-8*, 2003.

[7] W. Härdle, *Applied Nonparametric Regression*. Cambridge University Press, 1990.

[8] E. A. Nadaraya, "On estimating regression," *Theory Prob. Appl. 10, 186-90*, 1964.

[9] T. Gasser and H. G. Müller, "Estimating regression functions and their derivatives by the kernel method," *Scandanavian Journal of Statistics, 11, 171-85* , 1984.

[10] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "An application-specific communication protocol for wireless microsensor networks," *IEEE Transactions on Wireless Communications, vol. 1, no. 4, pp. 660-670* , Oct 2002.

[11] B. Deb, S. Bhatnagar, and B. Nath, "A topology discovery algorithm for sensor networks with applications to network management," in *IEEE CAS Workshop on Wireless Communication and Networking*, Sept 2002.

[12] T. Gasser, A. Kneip, and W. Kohler, "A flexible and fast method for automatic smoothing," *Journal of the American Statistical Association, Vol. 86, No. 415*, pp. 643–652, Sep 1991.

[13] P. Levis, N. Lee, A. Woo, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire tinyos applications," in *Sensys*, Nov 2003.

[14] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden, "Distributed regression: an efficient framework for modeling sensor network data," in *IPSN*, 2004.