ACE in the Hole: Adaptive Contour Estimation Using Collaborating Mobile Sensors

Sumana Srinivasan, Krithi Ramamritham and Purushottam Kulkarni
Department of Computer Science and Engineering
Indian Institute of Technology Bombay, Mumbai - 400076, INDIA
sumana, krithi, puru@cse.iitb.ac.in

Abstract

This paper focuses on the use of mobile sensors to estimate contours in a field. In particular, we focus on strategies to estimate the contour with minimum latency and maximum precision. We propose a novel algorithm, ACE (Adaptive Contour Estimation), that (a) estimates and exploits information regarding the gradients in the field to move towards the contour and (b) uses a spread component to surround the contour in order to optimize latency. While it is possible for sensors to spread as they approach the contour, it is crucial to judiciously determine when *and* how much *to spread. Spreading too early or too much may result in increasing the latency or affecting the precision. ACE dynamically makes this decision using local sensor measurements, history of measurements as well as collaboration between sensors while adapting to different types of deployment, distance from the contour and shapes of the contour. We demonstrate that ACE, in the absence of energy constraints precisely determines the contour with a lower latency than when only gradients are used for movement or when the sensors spread out right from the start of estimation. Additionally, we show that ACE significantly improves precision of contour estimation in the presence of energy constraints. We also demonstrate a proof of concept implementation on a mobile robot testbed.*

1. Introduction

Contour estimation is the estimation of the boundary formed by connecting a set of points of equal value, e.g., all points with the same concentration level in a pollutant spill or all points with the same height in a terrain. The set of points on the contour define a level set. Consider the scenario of an oil spill. Following the occurrence of the spill we must determine the extent of the spill (in particular the most hazardous level set) and track it as it moves. A contour estimation technique that quickly and accurately estimates the contour is vital to prevent the spread of contaminants and to take corrective actions immediately. Estimation and tracking of contours is used to contain spread of oil-spills [7], determine level of pollutants and plankton population [9] in water.

One of the techniques for contour estimation is remote sensing [6]. In [7], radar images of an oil spill are analyzed to obtain contours of different concentration levels. Pixel properties of the images such as intensity, color are correlated to oil concentration, density etc. While remote sensing has the advantage of spanning large geographical regions, measurement accuracy may be limited due to being distant from the phenomenon and the dependence on correlation between the actual and measured quantities. Further, observations can be hampered due to inclement weather conditions affecting accuracy of estimation. Also, the cost of deployment (of satellites and radars) can be quite high.

In-situ sensing, where sensors are in direct contact with the phenomenon addresses some of disadvantages of remote sensing based contour estimation. Recent advances in wireless sensor technology has resulted in the integration of insitu sensing, computing and communicating operations into low-power low-cost sensing platforms and have enabled numerous applications in areas like environmental monitoring [11], structural monitoring [16] etc. Static in-situ sensor networks are also used for contour estimation [19, 13]. The sensors measure the phenomenon and coordinate with each other to build a model of the field. This model is used to generate contour maps based on level sets of interest. The larger the density of the sensors covering the area of interest, the higher is the accuracy of estimation [15]. For high accuracy of estimation and coverage, there is a need for a large number of sensors to be deployed over a large geographical area which is costly and complex. Additionally, if the phenomenon itself is dynamic, for example a moving oil spill, static sensor nodes are not useful once the spill moves out of coverage and redeploying the network may be prohibitively expensive and time consuming. One potential technique to address the disadvantages of static sensors is to enable the sensors with mobility. A few mobile sensors can be deployed at different locations in the field. The sensors can move in the field and collect measurements. Once sufficient measurements are available, a model of the field can be built and used to generate contour maps.

While all of the above techniques can be used to estimate the contour, they cannot be directly used in cases where we need to reach the contour to take preventive actions such as spraying of anti-pollutants or deploying booms to contain an oil spill. In such cases, the mobile sensors need to physically move to the contour, cover the contour and take preventive actions to reduce the time required to start corrective operations. In this paper we focus on techniques to maneuver mobile sensors to locate and trace a contour of a given value in the field. The benefits of mobility come with a cost of increased energy usage and latency in estimation. The energy consumption of a mobile sensor is due to mobility, communication, sensing and computation with mobility being at least as expensive as communication. As the sensors work autonomously in the environment, the amount of energy available for sensor operation is limited. As a result, it becomes important for the sensors to perform these tasks (together referred to as contour estimation) such that the contour is precisely determined and the incurred latency is minimized.

In this paper, we explore the challenges and implications of enabling sensor devices with mobility to perform contour estimation. We propose a novel algorithm ACE, that intelligently combines information from local measurements of sensors, history of measurements collected over time and coordination with other sensors to locate and trace the desired level set while minimizing latency and maximizing coverage in the presence of energy constraints.

1.1 Challenges and Contributions

The task of contour estimation using mobile sensors essentially consists of two steps— (i) movement towards the contour in the field (*converge phase*) and (ii) tracing the contour (*coverage phase*). The difficulty of these tasks depend on the amount of information available to the sensors like variation of the field value in the sensor field, number of sensors deployed, type of deployment, size of the field and size of the contour. If the exact characteristics of the sensor field are known a-priori, for e.g., a perfectly gaussian field, then the contour can be located easily by following the gradient direction. If no knowledge regarding the sensor field is available, we may have to perform an exhaustive or a random search of the entire field to determine the points on the contour, which results in high latency and energy consumption. The question then is, *can we accurately estimate the contour with partial information, such as local measurements and measurements from other neighboring sensors? Can the estimation be done with minimum latency and energy usage?* The overall latency of estimation is the sum of the latency in converge and coverage phases. Some of the factors that influence latency and energy usage of contour estimation are (i) the extent of spread of sensors in the field, (ii) the distance of sensors from the contour and (iii) the size of the contour.

Since the sensors have no previous knowledge as to where the contour lies, the first challenge is — *how does a sensor know in which direction to move so as to quickly arrive at the contour?* In ACE, sensors estimate the direction of movement based on a series of local readings and by coordinating with other sensors to move towards the contour as quickly as possible. If many sensors happen to be collocated in the field as they move towards the contour, they will also converge on to the same point on the contour thereby not benefiting from the presence of multiple sensors. Hence there is a need for the sensors to spread out as they approach the contour to enable different sensors to trace different parts of the contour in parallel and minimize the overall latency of estimation. If the sensors are deployed far away from the contour, spreading out very early will increase the converge phase latency and it may be beneficial for the sensors to spread closer to the contour. If the size or extent of the contour is very large, the sensors need to spread out right from the start to cover the contour efficiently. Hence there is a need for each sensor to balance between moving towards the contour and spreading out based on the collocation of sensors in the field, distance of the sensor from the contour and the size of the contour. The important question is — *how do sensors adaptively decide to move towards or spread around the contour?*

The main contribution of this work is *ACE*, a novel algorithm which adaptively decides whether the sensor should move towards the contour or spread out. ACE uses local measurements, history and collaboration between sensors to estimate the factors that affect latency as mentioned above. Our experimental evaluation shows that ACE significantly reduces latency and increases precision even in the presence of energy constraints compared to the strategies that do not use adaptive spread.

The rest of the paper is organized as follows. Section 3 discusses the assumptions and problem formulation. Section 4 describes our approaches. Section 5 presents a description of our experimental setup along with the results and observations. Section 6 describes related work and we conclude with pointers to future work in Section 7.

2. Problem Definition

Our system model consists of mobile sensors deployed in a two dimensional scalar field with bounded area and capable of (i) measuring an attribute f within a sensing radius r_{sense}, (ii) communicating within a transmission radius of r_{trans} and (iii) moving with a finite number of steps r_{step} in any direction.

Once deployed, any given sensor can sense the attribute value f at its current location (x, y) as well as within its sensing neighborhood defined by a circle of radius r_{sense} and center at (x, y). As sensor measurements are error prone, we assume that the sensor measurement at any location is the average of multiple sample measurements taken at that location. Sensors are calibrated and the measured values are within the linear operating range of the sensors (no saturation). We assume that sensors have no odometry errors and have enough energy to move a maximum of n_{max} number of steps. Sensors are aware of their locations. Any given sensor can communicate with all other sensors and can exchange location and field value information. We characterize the energy consumed by the sensor as the sum of mobility, communication and sensing costs. The mobility cost per sensor is the number of steps taken by the sensor for estimation. We define our problem as follows:

Let N sensors be deployed in a bounded two dimensional field, $f : \Re^2 \rightarrow [L, U]$ with bounded field amplitude with upper and lower bounds U and L respectively. Let $C \equiv \{(x, y) \in \Re^2 : f(x, y) = \tau\}$ for some $\tau \in [L, U]$ represent the set of points on the contour with field value τ. The task is to determine C with minimum latency. Next we define the metrics to evaluate performance of strategies for contour estimation.

The accuracy of estimation is measured by *precision* which is defined as follows. If C_{est} represents the set of points on the contour estimated by a given algorithm, then, in the absence of sensing errors,

$$\text{Precision} = \frac{|C \cap C_{est}|}{|C|} \qquad (1)$$

where C is the set of all points on the actual contour.

The *latency* of estimation is defined as the maximum number of steps taken by the sensors in the network to estimate the contour. Since the energy consumed is directly proportional to the number of steps taken, minimizing latency implies minimizing energy consumed for estimation. If, t_i is the number of steps taken by the i^{th} sensor,

$$\text{Latency} = argmax_i(t_i) \text{ where, } i = \{1, 2, \cdots, N\} \qquad (2)$$

In the next section, we describe three contour estimation approaches in detail.

3 Approaches

In our first approach *Direct Descent*, the sensors explore their neighborhood and move along the gradient direction to approach the contour.

3.1 Direct Descent Algorithm (DD)

A scalar field associates a scalar value to every point in space. This is often used to indicate the distribution of a physical quantity such as temperature, concentration in space. In a scalar field, the gradient direction is perpendicular to the contour and gradient descent is a technique often used to approach the contour [12].

In the converge phase of DD, each sensor makes use of local measurements which includes the measurement of the field value at the current location as well as the measurement of field values sampled at finite number of points in the sensing neighborhood. The sensor moves to the neighboring location whose field value is closest to that of the target contour value (greedy heuristic). The movement in DD is based on a parameter called the *distance function* which is computed for each of the neighboring points that are sampled.

The **distance Function** d_f at location (x, y) is defined as the difference between the field value at the sensor's current position (x, y) and the field value at the contour, τ and is given by

$$\mathbf{d_f(x, y)} = \begin{cases} (1 - \frac{f(x,y)}{\tau})^2 & \text{if } f(x, y) \leq \tau \\ (1 - \frac{\tau}{f(x,y)})^2 & \text{if } f(x, y) > \tau \end{cases}$$

where $\tau > 0$ is the level set value, $f(x, y)$ is the field value at a location (x, y). $d_f(x, y) = 0$ when $f(x, y) = \tau$.

At every step, any given sensor moves to that neighboring location which minimizes the distance function. If the sensor revisits a location, then the sensor is trapped in a local minimum and it stops moving or else if the sensor moves to a location on the contour, then it begins the coverage phase as described below.

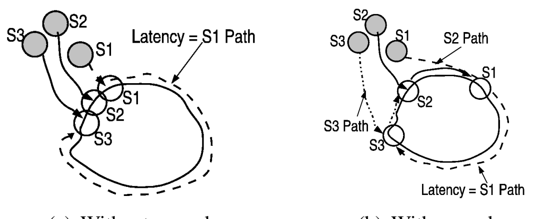
The coverage phase of a sensor begins when the sensor arrives at the contour. As and when the sensors converge on to the contour, the sensors trace the contour in a distributed fashion such that the path traced by them do not overlap. DD uses the traditional wall following approach (used in solving mazes) to trace the contour. As long as the contour is connected, the wall following algorithm traces any arbitrary shape of the contour. Every sensor that lands on the contour begins tracing the contour in the clock-wise direction. The sensors follow the right-hand rule where in the sensor aligns itself such that the contour is on the right of the sensor. The sensor halts when it traces a point already traced by a different sensor on the contour. When all of the sensors on the contour have halted, the coverage phase ends and the sensors send their estimated points to the user[1].

[1]In the absence of energy constraints, each sensor has unbounded number of steps and the contour is said to be completely estimated when all the tracing sensors halt. In the presence of energy constraints, the sensors trace the contour until they exhaust their allotted number of steps to guarantee maximum coverage

Next we discuss some of the shortcomings of DD to motivate the need for spreading of sensors.

3.1.1 Critique of DD

Consider the case where the sensors are collocated in the field and use DD. The sensors end up being collocated on the contour as well as shown in Figure 1(a). Sensors $S1$, $S2$ and $S3$ arrive close to each other on the contour and a major portion of the contour is traced only by a single sensor $S1$, resulting in a higher latency as shown by the dotted line in Figure 1(a). When the sensors spread out as shown in Figure 1(b), the task of tracing the contour is shared by $S1$, $S2$ and $S3$ minimizing the overall latency. Now, we describe our next approach, *Spread Always* algorithm that spreads out the sensors in the field.



(a) Without spread (b) With spread

Figure 1. Effect of Spread on Latency

3.2 Spread Always Algorithm (SA)

Spreading of sensors can be achieved in several ways. One way is to move the sensors such that each node tries to move as far away as possible from its neighbors after deployment [14]. However, the basic disavantage of this method is that at every step, the sensor needs to communicate its location to its neighbors (at least) or all other sensors (at most).

In SA, the spread is modeled based on *angular distribution* of sensors around a *centroid*. The *centroid* is defined as a point equidistant from all the points on the contour. When there are no estimated points on the contour, the coordinates of the center of the field (equidistant from the boundaries of the field) is chosen as the initial approximation of centroid. The centroid is useful in many ways — It acts as the center of the angular distribution, it helps sensors to move towards the contour when stuck in local minima and it also helps in determining how much to spread by estimating the area of the contour.

If N sensors are deployed in the field then the field is divided into $(\frac{2\pi}{N})$ angles centered around the centroid. Each sensor is assigned an angle $\theta^{'}$, called the *target angle*, such that the difference between the sensor's current angle (with respect to the centroid) and the target angle, which is a measure of the angular distance is minimized. In Figure 2, five sensors ($S1...S5$) are deployed in the field and the field is divided into five sectors and the target angles are $0, 72, 144, 216, 288$ degrees. The current angle θ, and target angle $\theta^{'}$ of S3 are as shown in Figure 2. S3 is assigned a target angle such that the angular distance travelled, θ_d, is minimized.

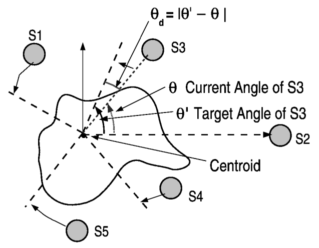


Figure 2. Angular spread around the centroid.

In order to model spread, we define a parameter called the *spread function* that captures how far a given sensor is from its target angle.

The **spread function** s_f at location (x, y) is defined as the difference between the current angle[2] (θ) and the target angle $\theta^{'}$ and is given by

$$s_f(x,y) = (\frac{\theta_d}{2\pi})^2 \text{ where } \theta_d = |\theta^{'} - \theta| \tag{3}$$

Note, $s_f(x, y) = 0$ when $\theta = \theta^{'}$ at (x, y). [3]

In converge phase of SA, at every step, the sensor computes $s_f(x, y)$ for different locations in its neighborhood. The sensor then moves to that neighboring location which minimizes the spread function. Just like in DD, if the sensor is trapped in a local minimum, it stops moving. If the sensor arrives at the contour, then it begins the coverage phase.

In coverage phase, SA performs wall moving to cover the contour like DD.

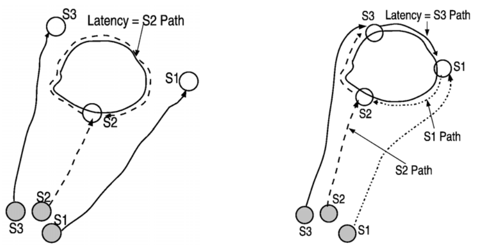
3.2.1 Critique of SA

Next we argue the need to spread judiciously by showing that spreading always is not beneficial all the time. Consider the case sensors are very far from the contour as shown in Figure 3(a). Spreading out always may result in a sensors moving away from the contour and not converging on the contour (sensors $S1$ and $S3$ failed to converge on the contour) or taking a large distance to arrive at the contour resulting in high latency. It is beneficial if the sensors begin to

[2]If (x_c, y_c) represents the current centroid estimate, then $\theta = tan^{-1}\frac{y-y_c}{x-x_c}$ (all angles are measured in radians).

[3]In the case where the sensor's current angle is in the first quadrant and the target angle is in the fourth quadrant, we use the smaller angular difference $(2\pi - \theta_d)$. Also, we divide by 2π in order to normalize

spread out only when they are close to the contour in order to minimize latency. In Figure 3(b), the sensors converge on to the contour minimizing the latency in coverage phase.



(a) Spread Always (b) Adaptive Spread

Figure 3. Effect of Adaptive Spread on Latency

The decision of how much to spread also depends on the size of the contour. For very small contours it may be sufficient for just a few sensors to trace it and as a result the sensors should arrive at the contour as quickly as possible without spreading out since spreading out always increases the converge phase latency for very small gain in coverage phase latency.

Hence spreading always is not judicious and there is a need for balancing out the movement towards the contour and spreading out. Next, we present a robust approach that intelligently combines movement towards the contour as well as spreading out called *Adaptive Contour Estimation*.

3.3 Adaptive Contour Estimation (ACE)

From the previous section we see that the decision to spread or not, when and how much to spread depends upon the following parameters (i) *extent of spread* of the sensors in the field, (ii) *distance* of the sensor from the contour and (iii) *size* of the contour. Next, we describe how ACE makes the choice between judiciously by dynamically estimating these parameters.

At each step, every sensor needs to choose between moving towards the contour or spreading out. In Section 4.1, we described how DD uses the *distance function* to arrive at the contour and in Section 4.2 how SA uses the *spread function* to achieve spread. In ACE, these functions are combined so that ACE can dynamically choose between minimizing the distance function or the spread function. We define adaptive spread function at location (x,y), $as_f(x,y)$ as a weighted combination of distance function $d_f(x,y)$ and spread function $s_f(x,y)$ and is given by:

$$as_f(x,y) = \alpha \times d_f(x,y) + (1-\alpha) \times s_f(x,y) \qquad (4)$$

where, $(0 \leq \alpha \leq 1)$ is the *biasing factor*

Just like the previous two approaches, at every step, the sensor chooses to move to that neighboring location which minimizes the $as_f(x,y)$. Unlike the previous two approaches, the sensor does not halt when it is trapped in a local minimum. It moves to the neighboring point that is closest to the estimated centroid if the sensor is outside $(f(x,y) < \tau)$ or to the point farthest from the estimated centroid if the sensor is inside $(f(x,y) > \tau)$ the contour respectively.

From Equation 3, we see that, by choosing $\alpha = 1.0$, the sensor only moves towards the contour (DD) and if $\alpha = 0$, the sensor only spreads (SA). From the previous section we see that α needs to lie between $(0,1)$ depending on the distance from the contour, the size of the contour and extent of spread of sensors in the field as discussed previously. ACE estimates these parameters as described below.

3.3.1 Estimating Distance From Contour

In ACE, the sensors estimate the normalized distance from the contour denoted as δ, by analyzing the rate of change of field values at all the locations visited so far by the sensor and based on this information, estimates where the field value becomes τ. At each step, every sensor records its current position (x_i, y_i) and the corresponding observed field value z_i. At the end of the k^{th} iteration, the sensor fits $\{(x_0, y_0, z_0), \ldots (x_k, y_k, z_k)\}$ using a standard non-linear regression function used to model physical processes [17].

$$z_i = p_0 + p_1 e^{-p_2 * x_i} + p_3 * e^{-p_4 * y_i} \qquad (5)$$

where the coefficients $p_0 \ldots p_4$ are estimated using Nelder Mead technique [10] and the sum of squares error between the observed z_i and the calculated z_i is minimized. Once the non-linear functional form is estimated, we need to estimate $(\hat{x}, \hat{y})$ where $\hat{z} = \tau$. For this, we find y_i for fixed values of x_i in the field , such that y_i also lies in the field (if $\Re^2$ is represented as a square field of dimensions $[l \times l]$, then for $x_i \in [0,l]$ we find y_i such that $y_i \in [0,l]$) . We choose the closest (x_i, y_i) pair to be an approximation of $(\hat{x}, \hat{y})$. The Euclidean distance between the current position of the sensor (x_k, y_k) and $(\hat{x}, \hat{y})$ is the estimated distance from the contour and is denoted by δ.

3.3.2 Estimating the Size of Contour

In ACE, the sensors coordinate to estimate the size of the contour. The size of the contour is characterized by the area of the envelope bounding the already estimated points on the contour if sensor(s) have landed on the contour or the estimated convergence points on the contour if none of the sensors have converged on to the contour.

$$\rho = \frac{\text{Area of envelope bounding estimated points on contour}}{\text{Area of the field}} \qquad (6)$$

where the area of the field is used as a normalizing factor.

3.3.3 Estimating the Extent of Spread of Sensors

We define the extent of spread of sensors in the field, S to be the ratio of the area of the convex hull formed by the sensors' locations to the field area.

$$S = \frac{\text{Area of convex hull of current positions}}{\text{Area of field}} \quad (7)$$

Finally we combine the three parameters to compute the bias factor α.

3.3.4 Computing Bias Factor

The bias factor α balances between moving towards the contour and spreading out as seen in Section 4.3. We now show that the value of α depends on parameters described in the previous sections.

- *α decreases with extent of spread, S*: If the sensors are are clustered in the field (as seen in Section 4.1.1.), the sensor's movement should be biased towards the spread function s_f in Equation 3.
- *α increases with the distance from the contour, δ*: If the sensors are deployed far away from the contour, they need to approach the contour before spreading (as seen in Section 4.2.1) and the sensor's movement should be biased towards the distance function d_f in Equation 3.
- *α increases with decreasing size of contour, ρ*: If the contour is large, the sensor's movement is biased towards the spread function s_f else if the contour is small, it is biased towards the distance function d_f.

Since the field is nonlinear in nature, we model α as a positive hyperbolic *tanh* function of the distance δ. The rate at which α varies with δ is controlled by a multiplicative factor k. The higher the value of k, sooner does α approaches 1. The value of $k = g(\rho, S)$ is chosen based on the degree of spread as well as the size of the contour.

$$\alpha = tanh(k\frac{\delta}{\delta_{max}}) \quad (8)$$

where δ_{max} is maximum predicted distance from the contour. Figure 4 shows the variation of α with δ for different values of k. We define $k = \frac{S}{\rho}$ and make α completely adaptive to the distance from the contour, size of the contour and the degree of spread of sensors.

Apart from deciding whether to move towards the contour or spread towards the target angle, it is also important to assign target angles to the sensors such that, the nonconverged sensors do not arrive at those locations on the contour which have already been traced by other converged sensors. ACE keeps track of covered sectors and maneuvers the nonconverged sensors to move towards the uncovered sectors as described in the next section.

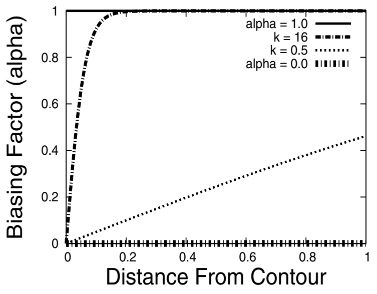


Figure 4. Modeling Bias Factor

3.3.5 Moving Sensors Towards Uncovered Sectors

After computing the bias factor, ACE computes the spans of covered sectors (angle subtended at the centroid by a tracing sensor's path on the contour) as shown in Figure 5. This information is used to compute the number of uncovered sectors and their spans. If the number of uncovered

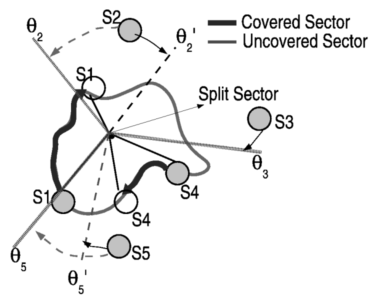


Figure 5. Target Angle Assignment

sectors is lesser than the number of nonconverged sensors, then the largest uncovered sector is split successively until the number of uncovered sectors is equal the number of nonconverged sensors. The angular bisector of the uncovered sector is the target angle corresponding to the sector. This is done so as to encourage the nonconverged sensor to move away from the converged sensor tracing the neighboring sector. In Figure 5, sensor S3 is assigned θ_3 and S2, S5 are assigned $\theta_2^{'}$ and $\theta_5^{'}$ which are the angular bisectors of the uncovered sectors. If the covered sector information is not used, S2 and S5 would have been assigned angles θ_2 and θ_5 that are already covered by sensor S1. By keeping track of the uncovered sectors, these sensors are now re-assigned $\theta_2^{'}$ and $\theta_5^{'}$ thereby directing them to the uncovered portions of the contour. The target angles are assigned to the nonconverged sensors such that the angular distance traversed by the sensors is minimized. In the next section we describe the control flow in the ACE algorithm.

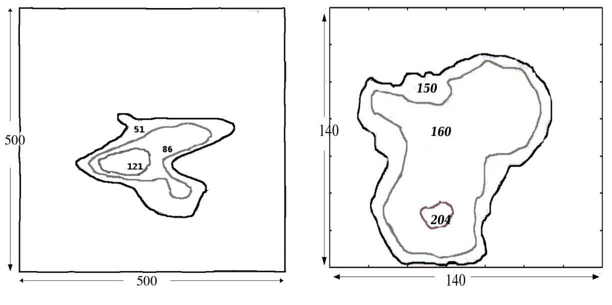
3.3.6 The ACE Algorithm

Once the sensors are deployed, ACE determines the degree of spread of sensors. If the sensors are clustered, it spreads out using SA otherwise uses DD for n_{steps}. At the end of n_{steps}, with the data measured (location and field values), ACE computes for each sensor, extent of spread of sensors S, the distance δ from the contour, the centroid and area of the contour ρ. The bias factor α and is computed using Equation 4 and the value of k is computed with the estimates of the S and ρ. ACE keeps track of the centroid computed previously. If the position of the centroid has changed (determined by the distance between the newly computed centroid and the previous centroid is greater than a threshold), the target angles are recomputed based on the new centroid position and are reassigned to the sensors depending on the number of uncovered sectors as described in Section 4.3.5. Next, ACE determines the neighboring position with minimum as_f value using α. If the sensor has visited the location previously (local minimum) and is located outside the contour ($f(x, y) < \tau$) then it moves to that neighboring position which is nearest to the centroid. Otherwise, if the sensor is located inside the contour ($f(x, y) > \tau$) then it moves to that neighboring location that is farthest from the centroid. When the sensor arrives at the contour, it begins the coverage phase which is similar to DD as explained in Section 3.1.

4. Experimental Evaluation

In this section we study the performance of ACE, DD and SA using latency and precision as evaluation metrics (Section 2). We consider two scenarios (i) without energy constraints (unbounded n_{steps}) and (ii) with energy constraints (bounded n_{steps}) to evaluate performance. The results (along with 95% confidence interval) shown are the average values over $n_{sim} = 1000$ simulation runs. For the unbounded case, a single sensor landing on the contour guarantees 100% precision and therefore we use latency as the performance metric. In addition we also use another metric *Convergence Percentage (CP)*, that captures the number of times where at least one sensor landed on to the contour in our experiments. In the bounded case, one or more sensors landing on the contour does not guarantee coverage since the sensors may not have enough energy to cover the contour and we use precision as metric of comparison. We compare the performance of our algorithms by varying the following parameters — field, contour shape and size, type of deployment and energy level of the sensors.

For sake of simplicity, the field is approximated to be



(a) Pollutant Contours (b) Light Contours

Figure 6. Contours for Evaluation

a two dimensional grid of size $[l \times l]$. Sensors can sense the attribute value f, at each of the grid locations. At any given point, the sensing radius is said to encompass its eight neighboring points in a square tessellation. The sensors can move to any one of its eight neighboring positions.

We use two different types of scalar fields to evaluate performance. First, we used a pollutant dispersion modeling tool WQMAP[4] to generate a pollutant concentration field. We ran the simulation with three pollutant load sites and used built-in hydrodynamics for simulating the effect of the spread the pollutants in water. The simulation ran for 120 time steps and a two dimensional pollutant concentration field of dimensions $[500 \times 500]$ was generated. Figure 6(a) represents pollutant field contours used in our experiments. Next, we created a light field of varying light intensity in the lab, on a grid of dimensions 15×15. We measured the light intensity at each of the grid positions using a single Crossbow Mote. This field was scaled and interpolated to a grid of length $[140 \times 140]$. Figure 6(b) depicts different contours in the light field used for evaluation.

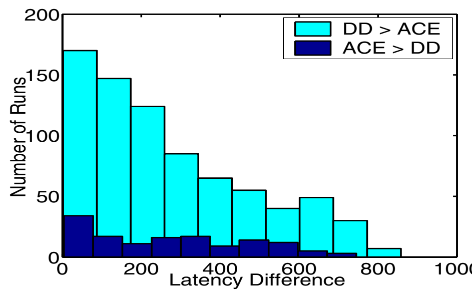
A contour is considered *large* if the area of the contour is greater than 50% of the area of the field (L_{1L} and L_{2L} correspond to level sets $\tau = 150$ and $\tau = 160$ in Figure 6(b)), *medium* if the area of the contour is 30 -50% of the field area (M_{1P} and M_{2P} correspond to level sets $\tau = 51$ and $\tau = 86$ in Figure 6(a)) and *small* if it is lesser than 30% of the field area (S_{1L} and S_{2P} correspond to level sets $\tau = 121$ and $\tau = 204$ in Figure 6).

In order to observe the effect of type of deployment on the algorithms, we use two different types of deployment - (i) Non-clustered — the sensors are distributed uniformly randomly in the field $(x, y) \in U(0, l)$ and (ii) Clustered — the sensors are distributed uniformly randomly within a small circle of radius $\sqrt{N}$ where, N is the number of sensors and the center of the cluster itself is placed uniformly randomly in the field Each simulation consists of a initial deployment of sensors and the sensors move a maximum

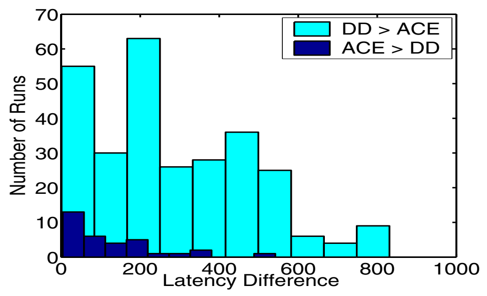
[4]Applied Science Associates Inc., http://www.appsci.com/

Contours	Nonclustered					Clustered				
	ACE	DD		SA		ACE	DD		SA	
		Latency	CP	Latency	CP		Latency	CP	Latency	CP
S_{2L}	46 ± 1.2	46 ± 1.2	99	59 ± 1	20	135 ± 30	285 ± 71	100	47 ± 4	33
S_{1P}	241.7 ± 11	243 ± 9	63	276 ± 8	29	225 ± 21	258 ± 5	14.5	267 ± 8	4
M_{2P}	398 ± 10	514 ± 14	53	623 ± 14	43	654 ± 50	721 ± 12	31	733 ± 44	11
M_{1P}	429 ± 8	550 ± 11	71	693 ± 16	78	806.5 ± 52	950.5 ± 5	35	994 ± 22	22
L_{2L}	134.7 ± 2	140.8 ± 2	100	218.1 ± 4	99	345 ± 4	409 ± 3	100	433 ± 5	78
L_{1L}	128 ± 1	133 ± 2	100	213 ± 4	100	311 ± 4	383 ± 3	100	408 ± 4	83

Table 1. Effect of adaptive spread on latency and convergence percentage[CP].



(a) Non-clustered deployment



(b) Clustered deployment

Figure 7. Distribution of latency difference for ACE and DD for M_{1P} contour.

of n_{max} steps (1000 for non-clustered and 2000 for clustered deployments) before terminating the simulation. In ACE, the sensors perform re-estimation of parameters every $n_{steps} = 5$.

4.1 Latency Comparison

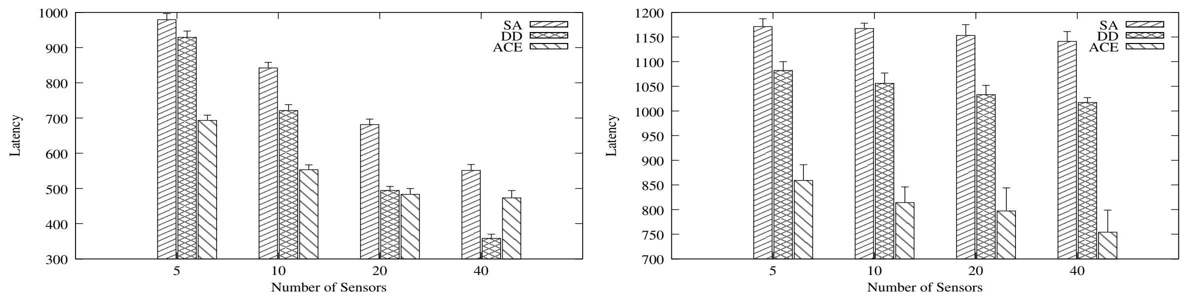
We compare the latency of ACE, DD and SA when the precision is 100% for different contours in both light and pollutant fields and for non-clustered and clustered deployments. Table 1 depicts the average latency values and convergence percentage (CP) of DD and SA. For small contours and non-clustered deployment, ACE performs similar to DD. For small (S_{2L}) contour in the clustered deployment case, even though SA performed the best, CP of SA was much lower (33%). For the large contours in the light field (L_{1L} and L_{2L}), ACE and DD performed similarly as the sensors are deployed close to the contour (light field is smaller than the pollutant field) and there is not much of a chance to spread out before landing on the contour. However for medium (M_{1P}, M_{2P}) sized contours and non-clustered deployment, ACE has 22% and 38% performance gain when compared to DD and SA respectively. For medium and large contours (M_{1P}, M_{2P}, L_{1L} and L_{2L}) the performance gain using ACE is 9-18% over DD and 11-23% over SA in the clustered deployment case.

The CP of DD and SA when the precision of ACE is 100% is as shown in Table 1. We see that the CP for DD and SA was better in the light field than in the pollutant field. The intuition behind this behavior is that the size of the field is smaller in the light field than the pollutant field, increasing the possibility of sensors landing on the contour. As contour size increases, CP increases and it is higher for non-clustered deployments when compared to cluster deployments.

Figure 7 depicts the distribution of the latency difference between ACE and DD for cases where the latency of ACE is more than DD and less than DD. The results are shown for M_{1P} contour in the pollutant field only due to lack of space. For non-clustered deployment, we see that number of cases where the latency of ACE is more than DD much smaller than the cases where DD was less than ACE. The number of cases where latency of ACE was less than DD was a factor of 2-12 times more than the case when the latency of DD was less than ACE. In the clustered deployment case, ACE performs significantly better than DD.

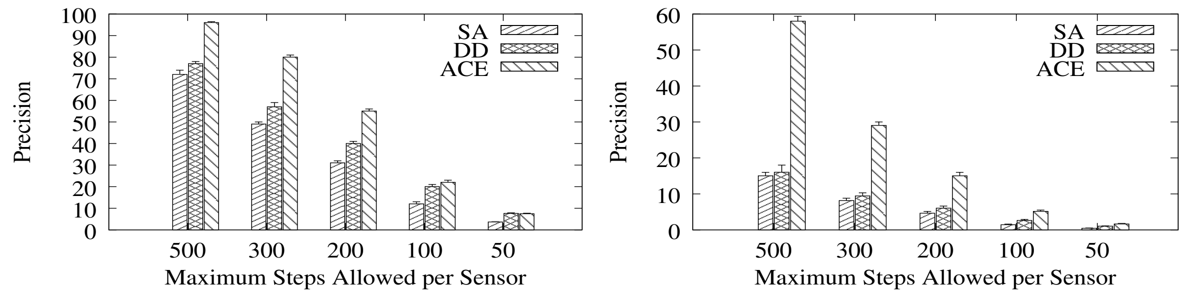
4.1.1 Sensitivity to Varying Number of Sensors

When the number of sensors (N) deployed was increased, we observe that the latency of ACE is 20-25% less than DD and SA for $N = 5$ and the latency of ACE and DD are similar for $N = 20$. Surprisingly, for $N > 20$, we observe that the latency of ACE is higher than DD by 15%. While



(a) Non-clustered deployment (b) Clustered deployment

Figure 8. Sensitivity to number of sensors (Contour M_{1P}).



(a) Non-clustered deployment (b) Clustered deployment

Figure 9. Effect of uniform energy bound on precision.

with clustered deployment, latency of ACE is consistently better— 23% and 33% less than DD and SA respectively. This result indicates that adaptive spread also depends on an additional parameter N. We intend to explore the impact of N on the biasing factor as part of future work.

4.2 Precision Comparison

In this section, we study the performance of ACE and DD in the presence of energy constraints based on the precision metric. In the bounded energy case, in order to maximize coverage the sensors do not stop when it traces a point already traced by another sensor but continues until all of the conour is traced or until it exhausts its battery power.

4.2.1 Uniform Energy Bound for all Sensors

In this experiment, we set all the sensors to have the same number of steps and observe the degradation in precision for ACE and DD. The maximum number of steps, n_{max} allowed per sensor is varied from 50 - 500 uniformly for all sensors. Figure 9 depicts the precision for non-clustered and clustered deployments for a large contour in the pollutant field. In Figure 9(a), we see that ACE has a 25-20% and 30-25% higher precision than DD and SA respectively for $n_{max} > 100$. In Figure 9(b), for clustered deployment, ACE has 45-10% higher precision than DD and SA for $n_{max} > 100$. For $n_{max} \leq 100$, ACE and DD has similar precision irrespective of the deployment.

4.2.2 Random Energy Bound for each Sensor

Next, we assigned different bounds for each sensor, chosen uniformly randomly in the range $[5 - 50\%]$ of n_{max} and we compared the precision and convergence percentage of ACE and DD for different deployments and contours. The results are tabulated in Table 2. We observe that ACE has a better precision, $4 - 26\%$ than DD for non-clustered and $19 - 30\%$ than DD for clustered deployments. Also, ACE exhibits better precision than SA, $15-90\%$ for nonclustered and $16 - 72\%$ for clustered deployments. This implies that when energy is limited, ACE delivers more precision than DD and SA. CP for ACE is uniformly high in the range 91-100%, while DD varies between 62-100% and SA has a higher variation, 19-100%.

4.3 Evaluating Feasibility

In order to validate whether the advantages identified by the simulation translates to practical benefits, it is important to understand (a) some of the assumptions made such as the mobility cost is indeed greater than communication cost and

Contours	**Nonclustered**						**Clustered**					
	ACE		**DD**		**SA**		**ACE**		**DD**		**SA**	
	Precision	CP	Precision	CP	Precision	CP	Precision	CP	Precision	CP	Precision	CP
S_{1P}	74 ± 6	91	48 ± 4	62	22 ± 1	30	30 ± 3	44	11 ± 1	10	3 ± 0.1	3
S_{2L}	98 ± 1	100	94 ± 1	100	9 ± 0.5	19	73 ± 6	97	43 ± 3	81	1 ± 0.1	3
M_{1P}	70 ± 6	100	48 ± 4	96	45 ± 2	93	31 ± 2	79	19 ± 1	32	20 ± 1	22
M_{2p}	72 ± 6	99	53 ± 4	95	38 ± 2	79	32 ± 3	71	19 ± 2	29	12 ± 0.7	12
L_{1L}	98 ± 8	100	98 ± 8	100	83 ± 5	100	97 ± 0.6	100	83 ± 4	99	81 ± 5	83
L_{2L}	96 ± 8	100	96 ± 8	100	79 ± 4	99	99 ± 0.51	100	96 ± 0.5	100	71 ± 4	77

Table 2. Effect of random energy bounds on precision and convergence percentage[CP].

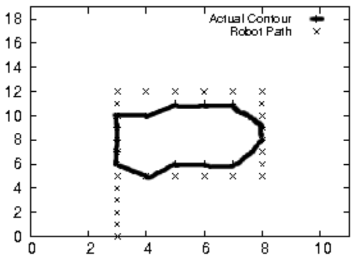


Figure 10. Mobile sensor tracing the contour.

(b) assess the feasibility of implementing the algorithms. As a first step in this direction we (a) implement DD (gradient descent and wall moving), a component of ACE on a robot and (b) perform a simplified energy characterization of the algorithms.

The test-bed comprises of a single mobile robot moving along a grid of dimension 11×18 with granularity 8cm. in a light field of varying light intensity. We determine the ground truth by measuring the light intensity at each grid point. The robot is equipped with a three white line sensors, two shaft encoders and two ultra low power DC motors to move along the grid. The robot comprises of a rotating arm on which a light sensor capable of measuring light intensity is mounted. The rotating arm is powered by two servo motors. The length of the rotating arm is equal to one grid cell length and the light intensity value is measured at the eight neighboring grid intersections. The robot is equipped with ATMEGA128, 11.06 MHz processor and the radio module comprises of 2.4 GHz CDMA, Infrared with 50m range.

4.3.1 Contour Estimation with a Mobile Sensor

First, we verify the feasibility of estimating a contour using a mobile sensor. Figure 10 shows the exact contour of value $\tau = 200 \pm 20$ for a single light source and the path taken by the robot to estimate the contour. The mobile sensor was programmed to use the DD algorithm for contour estimation. As can be seen from the figure, the estimated contour faithfully covers the exact contour. The error in estimation is due to the granularity of the grid used by the sensor to navigate.

4.3.2 Simplified Energy Characterization

In this experiment, we measure the energy consumed by the robot to move one grid cell length and to communicate 5 bytes of information — (i) sensor id (1 byte), (ii) x and y coordinates (2 bytes) and (iii) relative light intensity (2 bytes). The robot was powered by 8.4V DC. A multimeter is connected in series with the robot and the power supply and the current drawn and the time taken to travel one grid length is measured. The energy consumed for moving one step (m_c) along the grid was measured to be 4.96J. This was calculated by measuring the voltage supplied to the robot (7.7V), current drawn for moving one step (205mA) and time taken to move on grid step (3.4s). Next, we measure the energy consumed for transmitting (t_c) and receiving a message (r_c). Even though the time taken to transmit a byte is 0.5ms, we introduce a delay of 1 ms to include radio initialization time. The current drawn for transmission and reception of a byte was derived from the CDMA[5] data sheet. The energy consumed for transmission of a single packet is computed by measuring the voltage supplied to the radio (8.4V), current drawn to transmit a single byte (69mA) and time taken for transmission including the delay for radio initialization (1ms) was 2.9mJ. Similarly energy consumed for receiving a packet of information was computed similarly and found to be 2.4mJ. If n_t and n_r represents number of messages sent and received, then the total energy spent is given by Energy consumed $= Latency \times m_c + n_t \times t_c + n_r \times r_c$

In our simulations, we assume a centralized single hop communication model. The sensors send and receive messages to and from the sink. For contours in the pollutant field, we measured the number of messages exchanged and the energy consumed by ACE and DD for different deployments as shown in Table 3.

[5] http://www.alldatasheet.com/datasheet-pdf/pdf/144993/CYPRESS/CYWUSB6935.html

Contour	Deployment	ACE				DD			
		Latency	Sent (n_t)	Recd (n_r)	Energy	Latency	Sent (n_t)	Recd (n_r)	Energy
M_{1P}	Non-clustered	429	2471	2471	2140J	550	2471	2471	2741J
	Cluster	806	5871	5871	4028J	950	1209	1209	4712J
M_{2P}	Non-clustered	398	3862	3862	1994J	514	1975	1975	2559J
	Cluster	654	7752	7752	3284J	721	1043	1043	3581J
S_{1P}	Non-clustered	241	3324	3324	1212J	243	979	979	1210J
	Cluster	225	8650	8650	1161J	258	728	728	1283J

Table 3. Comparison of total energy consumed by ACE and DD.

From Table 3, ACE has a (8-22%) lower energy consumption than DD for all contours except in the case of S_{1P}, non-clustered deployment where both ACE and DD have similar energy consumption. This is not surprising since ACE performs DD for small and nonclustered deployments. The latency values used in Table 3 are for the cases when both DD and ACE converge (see Table 1).

4.3.3 Summary of Results

We have demonstrated that ACE estimates the contour with 100% precision and $4 - 22\%$ lower latency for medium and large contours for non-clustered deployment, $15 - 52\%$ for clustered deployment. For small contours in the non-clustered case, the performance of DD and ACE was similar. When compared to SA, ACE has $22 - 40\%$ lesser latency in nonclustered deployments and $15 - 23\%$ in clustered deployments. In the presence of energy constraints we demonstrated that ACE delivers better precision $4 - 35\%$ for small and medium contours in the nonclustered deployment and $3 - 63\%$ in the clustered deployment than DD. We demonstrated the feasibility of DD on testbed with a single mobile sensor. With our simplified energy characterization we show that ACE consumed $8.3 - 22\%$ lesser energy than DD. Next, we describe related work and follow it up with conclusions and directions to future work.

5. Related Work

Our work bears a strong resemblance to boundary estimation even though we perform level set estimation. In our case, we have an added knowledge of the value of the level set we are looking for and thereby, we use this information to do the path planning for the mobile nodes. The basis of boundary estimation algorithms is the detection of the edge that separates two different regions [2]. Boundary detection and estimation using a network of static sensors has been studied extensively in the recent past [2]. The authors in [15] derive a theoretical bound on the number of sensors needed in a lattice network of static sensors to achieve a certain accuracy.

In [1], the authors use mobile nodes to adaptively scan the entire region to determine the boundary. Their strategy consists of placing the mobile sensors uniformly along the boundary of the region and scan the entire field in a raster scan interleaved manner. This approach does not assume or make use of any local gradient information to converge on to the boundary. In contrast our approach assumes that the field is continuous and bounded (the field value is bounded so as to define the existence of the level set) and the mobile nodes use the gradient at their current position with respect to the target field value to converge on to the contour. While the adaptive sampling method guarantees convergence (because the whole field is scanned) it does not exploit information available to the sensor. When the size of the field is large, sampling the whole field might be prohibitively expensive. In our approach, we exploit local gradient and previous history information to arrive at the contour with smaller latency. Another interesting contrast is in the case of dynamic boundaries. If the position of the level set is changing in time, the sensors can adapt since their motion is based on the current measurements. However, in adaptive sampling, the measurements made at the previous pass may not be valid at the current instant. It is our belief that we should exploit what ever field information we have to do the path planning and resort to scanning only in the case where such information is unavailable.

In [5] the authors propose the active contour algorithm used in image segmentation [12] to arrive at the boundary and trace the boundary by moving the nodes in and out of the boundary. They directly adapt the snake algorithm (sensors are deployed as a virtual snake) to estimate the level set. Our approach is similar to theirs in the use gradient information to approach the contour but does not require to deploy the sensors in a specific manner. Spread is achieved in their approach by defining a repulsion potential between neighboring sensors. This requires pair-wise communication at every step for movement. We model spread using angular distribution and sensors are not attached to their neighbors. In [3], the agents are allowed to converge and in the trace phase the agents move in and out of the boundary to trace the boundary. The agents interlace the traversal such that they sample different points. This results in a higher latency since all of them traverse the entire boundary. In contrast, we follow the wall moving algorithm and the task of tracing is shared by the sensors. Also, in [5], no study is reported regarding how the algorithm behaves in the presence of energy constraints.

Our previous work [18] proposed an algorithm MCD for estimating the contour based on the knowledge of the centroid of the contour. In the current work, we estimate the centroid based on history of movement, points already traced on the contour and sensor's current locations. We also overlap movement in converge and coverage phases and use wall moving algorithm to trace contours of arbitrary shape.

The work in [4] uses a static sensor network to guide a single mobile sensor to trace a contour. The mobile node computes the gradient using the readings from the static sensors and moves towards the contour. In our work, the estimation is done by autonomous nodes and no overhead of deploying a static network for the purpose is necessary.

In [8], the authors use unconstrained mobility to estimate level set and the agents. Their main assumption is that mobility comes for free. In scenarios like mapping of level sets in oil spills using autonomous robots, one cannot assume that mobility is free of cost. Our approach aims at reducing the mobility cost by intelligently estimating the shape and the location of the contour as the sensors move.

6. Conclusion

While it is intuitive for sensors to spread in a sensor field to estimate a contour, it is hard to determine, in the absence of the knowledge of the contour's exact location and extent, *when* and *how much* to spread. In this paper, we propose a novel algorithm ACE which intelligently decides to spread out or move towards the contour and demonstrate that ACE reduces latency of estimation irrespective of types of deployments, sizes and shapes of contour and improves precision of estimation when compared to the case where no adaptive spread is used even in the presence of energy constraints.

As part of future work, we intend to extend ACE to handle limited transmission ranges. Additionally, instead of performing re-estimation of parameters at every few steps, we plan to dynamically determine when any given sensor needs to reestimate, based on events such as when a sensor lands on the contour or when a sensor's estimated convergence point changes beyond a given threshold. We also plan to extend ACE to handle discontinuous contours and implement ACE on multiple robots to validate our simulation results.

Acknowledgement — We are thankful to Parmesh Ramanathan, Sachidanand Malewar and Amey Apte for their support during the course of this work.

References

[1] R. A. Singh and P. Ramanathan. Active learning for adaptive mobile sensor networks. In *IPSN*, pages 60–68, Nashville, TN, April 2006.

[2] K. Chintalapudi and R. Govindan. Localized edge detection in a sensor field. In *IEEE WSNPA*, pages 59–70, May 2003.

[3] C.Hsieh, Z.Jhin, and D. et al. Experimental validation of an algorithm for cooperative boundary tracking. In *American Control Conference*, volume 2, pages 1078–1083, Portland, June 2005.

[4] K. Dantu and G. S. Sukhatme. Detecting and tracking level sets of scalar fields using robotic sensor network. In *ICRA*, pages 3665–3672, Roma, Italy, April 2007.

[5] D.Marthaler and A.L.Bertozzi. *Recent Developments in Cooperative Control and Optimization*, volume 3, chapter 17, pages 317–330. Kluwer Academic Publishers, 2004.

[6] M. Fingas and C. Brow. Review of oil spill remote sensing. In *Spill Science and Technology Bulletin*, volume 4, pages 199–208, 1997.

[7] F. Galland, P.Refriegier, and O. Germain. Synthetic aperture radar oil spill segmentation by stochastic complexity minimization. In *IEEE Geoscience and Remote Sensing Letters*, pages 265–299, October 2004.

[8] G. Gupta and P. Ramanathan. A distributed algorithm for level set estimation using uncoordinated mobile sensors. In *GLOBECOM 2007*, pages 1180–1184, November, 2007.

[9] J.Cloern, A. Alpine, B.Cole, and T.Heller. Seasonal changes in the spatial distribution of phytoplankton in small, temperate-zone lakes. *Journal of Plankton Research*, 14(7):1017–1024, December 1992.

[10] J.Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[11] K.Martinez, P.Padhy, and A. et al. Glacial environment monitoring using sensor networks. In *Real-World Wireless Sensor Networks*, Stockholm, June 2005.

[12] L.Cohen. On active contour models and balloons. *Computer Vision, Graphics and Image Processing: Image Understanding*, 53(2):211–218, 1991.

[13] Y. Liu and M. Li. Iso-map: Energy-efficient contour mapping in wireless sensor networks. In *ICDCS*, pages 36–44, Toronto, Canada, June 2007.

[14] N.Heo and P.K.Varshney. A distributed self spreading algorithms for mobile wireless sensor netwoks. In *Wireless Communications and Networking*, volume 3, pages 1597–1602, New Orleans, Louisiana, March 2003.

[15] R. Nowak and U. Mitra. Boundary estimation in sensor networks: Theory and methods. In *IPSN*, pages 80–95, Palo Alto, CA, April 2003.

[16] J. Paek. A programmable wireless sensing system for structural monitoring. In *4th World Conf. on Structural Control and Monitoring*, pages 13–24, San Diego, July 2006.

[17] G. A. F. Seber and C. J. Wild. *Nonlinear Regression*. Wiley-IEEE, New York, 1989.

[18] S. Srinivasan and K. Ramamritham. Contour estimation using mobile sensors. In *DIWANS 2006, Los Angeles, USA*, pages 73–81, September 2006.

[19] X.Meng, T.Nandagopal, L.Li, and S.Lu. Contour maps: Monitoring and diagnosis in sensor networks. *Computer Networks*, 50(15):2820–2838, 2006.