An Adaptive Refresh Distributed Model for Estimation and Efficient Tracking of Dynamic Boundaries

Nagarathna
Dept. of Comp. Sc. and Engg
PES College of Engg.
Mandya, INDIA
Email:nagu_cjg@yahoo.com

Valli. S
Dept. of Comp. Sc. and Engg.
Anna University,
Chennai, INDIA
Email:valli@annauniv.edu

***Abstract*—We propose a distributed algorithm for tracking dynamic boundaries in a ranging sensor network. The main aim here is to minimize the number of data pushes to the sink by the observing sensors. Contour is modeled as correlated Brownian motion with drift. Sensors continuously sample the data. Neighboring sensors communicate and exploit the spatio-temporal correlation and using the parameters of the contour the time to push the sample data to the sink is predicted. A multihop path is established between sensor to sink to route the data. To get the global view of the contour sink apply non parametric regression on the sensor data. Along with the sample data sensors push the mean value so that the sink can estimate the sample point till the next push of data from sensor. The sensors push the data when the confidence in the estimate by the sink is below a specified thereshold. The performance of this model is compared with centralized model with respect to energy consumption for routing samples to sink.**

I. Introduction

Tracking dynamic fronts is one of the challenging applications of sensor network. Different aspects of this application are been observed in the literature e.g., [1], [2], [5]–[8]. Some of the application of tracking dynamic boundaries is like oil and chemical spills, gas leaks, forest fires and other such phenomena. Our main interest in this work is to reduce the communication cost in a ranging sensor network monitoring dynamic fonts by adaptively refreshing the measurements.

To have a global view of the contour dynamics the measuring sensors need to provide the measurements to the sink. If the sensors communicate the sample data at regular interval of time or if the reporting rate by the sensors is determined by the part of the boundary that change fastest, then there will be significant wastage of resource. This is because the boundary may change at different rates at different sections. So the main challenge in this dynamic contour tracking is to determine the time to communicate the data from the observing sensors to sink.

In [5], [6] a query based adaptive refresh rate model is proposed for tracking dynamic contour for a centralized system. In a centralized system all the sensors need to communicate the sampled data directly to the sink when queried. Most of the energy is consumed by the sensors to communicate with the sink. The nodes far away from the sink will have short life time compared to nearby sensors. As the number of sensor nodes increases, the sink will not be able to schedule the sensors, which leads to loss of packets. In a distributed system sensor nodes communicate with the neighbors and all scheduling and routing is taken care by the individual node. To forward the data to sink a multihop routing can be adopted.

In our framework we consider ranging sensors randomly deployed in a two dimensional plane and sense the phenomenon in y direction only. One of the example for such a category is radar sensors [9] which can range up to 60 feet. Sensors know their location and measure the distance to a contour point and compute the location of a sample point on the contour. An illustration of the measurement is as follows. If a sensor S_i is at location $(\check{x}_i, \check{y}_i)$ and at time t its ranging measurement in Y direction is $r_i(t)$ then the location on the contour point is computed as $(x_i, y_i) := (\check{x}_i, \check{y}_i + r_i(t))$. Fig 1 shows the scenario that we have considered.

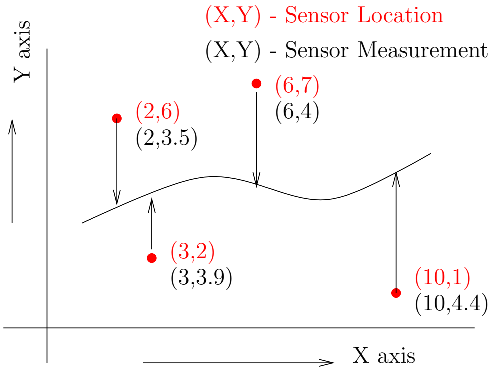


Fig. 1. Illustration for the sensor observation.

We have developed a model based distributed push algorithm for the sensors tracking the dynamic boundaries. The dynamics of a boundary points are modeled to be evolving as a correlated Brownian motion with drift. Sensors continuously measure the boundary points, communicates with the neighbor, but push the measurements to the sink based on the statistical predictions. A spatio-temporal model exploiting the correlation between the adjacent sensor is constructed to predict the time to push. At regular interval of time using the model parameters sensors predicts the time to push. We compare the performance of our distributed model with that of a centralized model with respect to energy consumption for routing the sample and other parameter to the sink.

A. Related Work

We now discuss some relevant literature on the tracking of dynamic boundaries by a sensor network, e.g., [1], [3]–[8]. In [8] subset of mobile sensors communicates the information to the central information processor via multihop routing. Sensors are selected based on the proximity of the mobile nodes to other nodes inside the boundary. The nodes follow a distributed coordination algorithm to re-position themselves along the boundary after selection. The subset of the mobile nodes selected for boundary estimation is not determined by the rate of the boundary change as observed by the sensor but is determined by the proximity of the mobile nodes.

In [3] the boundary is modeled as a hidden Markov model and Cumulative sum algorithm is proposed which can quickly detect small drifts in the parameters of the model. Observations are collected from multiple sensing vehicles which are autonomously tracking the boundary by estimating its approximate shape. With a test bed implementation an improved path planning algorithm for [3] is proposed in [4] for environmental boundary tracking and estimation without external positioning information. Even when a single node detects the change in the location of the boundary in both these schemes measurements are collected from all the sensors.

The main aim of [1], [5]–[7] is to minimizing the communication cost in dynamic boundary estimation. A cluster based boundary estimation algorithm is proposed in [7]. A tree-based boundary estimation scheme with hierarchical cluster heads is used to reduce the number of transmissions. Each cluster has a cluster head which reports the estimate to the sink at regular time intervals and hence reporting rate is independent of the rate of boundary change.

A combined spatial and temporal estimations is described in [1]. Initially the contour is estimated and stored in the sink. A Kalman filter based temporal estimation techniques is used to obtain upper and lower bounds for the boundary and communicated to the sink regularly. Sensors measure the boundary points only when it is outside the bounds previously obtained.

In [5], a query based refresh rate adaptation algorithm is proposed. Contour is modeled as Brownian motion model and the parameters of this model are continuously estimated by the sensors. The sink uses the model and continuously update the estimate of the sample point. Sink queries a sensor when the confidence in the estimate of the sample point falls below a threshold. An extension of this work is [6], where a spatio-temporal model is proposed for tracking dynamic boundaries. The contour evolves as correlated Brownian motion model. The prediction model at sink uses the stastistical parameters to pretict the time to query the senor pairs. Most recent measurements and the other parameters are communicated to sink directly in both [5] and [6].

In this paper we propose an adaptive refresh distributed push model for dynamic contour tracking. The contour model is same as in [6]. The prediction model is for the sensors to predict the push time. Sensors push the data to sink using the prediction model which exploits temporal correlation between the samples of the sensor at different time and spatial correlation between the neighboring sensors. Sensor forward the data using multihop path to th sink.

B. Organization of the paper

The organization of the paper is as follows. In Section II the distributed push model algorithm is explained along with the contour modeled as correlated Brownian motion model. In addition, the routing algorithm used in the distributed system and the contour estimation is also explained. A detail algorithm for sensor side and sink side is also given. In Section III we provide the details of the simulation model of the contour and the node deployment and also define the different performance measures used. In addition we have one set of simulation results discussed. In Section IV we have conclusion.

II. Distributed Push Model

Brownian motion is a well known model which captures the local variations and is a widely used model. We now construct a model for the contour dynamics. Let $Y(t) = [Y_1(t), Y_2(t), \ldots, Y_i(t), Y_{i+1}(t), \ldots Y_N(t)]$ be vector of sensor sample points at time t with x coordinates as -$x_1, x_2, \ldots, x_N$. $Y(t)$ is modeled to evolve as a correlated Brownian motion with drift. Let the corresponding mean of the sample points be $\mu(t) = [\mu_1(t), \mu_2(t), \ldots \mu_N(t)]$. The parameters of this Brownian motion are mean μ and covariance C. This means that conditioned on $Y(t)$, i.e., knowing the value of $Y(t)$, the value of $Y(t+\tau)$ is

$$Y(t+\tau) = Y(t) + N(\mu\tau, C\tau).$$

Here $N(\mu, C)$ is an N dimensional Gaussian random variable. Sink uses this model to estimate sensor measurements. Now we will explain how the sensors predicts the time to push the sample and other parameters to the sink. We consider a cluster of two sensors. If S_i and S_{i+1} are the adjacent sensors in a cluster with a samples $Y_i(t)$ and $Y_{i+1}(t)$ at time t, we will make an assumption that both the sensors communicate via single hop or multihop at regular interval of time and only the adjacent samples are correlated. Let $Y_i(t)$ and $Y_{i+1}(t)$ be the true samples at time t and $\hat{y}_i(t)$ and $\hat{y}_{i+1}(t)$ be corresponding estimates.

Let $Z_i(\tau) = [\hat{y}_i(t+\tau) - Y_i(t+\tau)]$ and $Z_{i+1}(\tau) = [\hat{y}_{i+1}(t+\tau) - Y_{i+1}(t+\tau)]$. Here Z_i and Z_{i+1} are a zero mean Gaussian with variance σ_i^2 and σ_{i+1}^2 respectively. Hence $Z(\tau) := Z_i(\tau) + Z_{i+1}(\tau)$ will also be a zero mean Gaussian with variance $\sigma^2\tau$, where σ^2 is given by

$$\sigma^2 = \sigma_i^2 + \sigma_{i+1}^2 - 2\sigma_i^2\sigma_{i+1}^2\rho_{i,i+1}$$

Here $\rho_{i,i+1}$ is the correlation coefficient between S_i and S_{i+1} at time $t+\tau$.

We will assume that the actual contour may be interpolated by a straight line between the points (x_i, Y_i) and (x_{i+1}, Y_{i+1}) as illustrated in Fig 2, and get the trapezoid $(x_i, Y_i, Y_{i+1}, x_{i+1})$ and call as T. Similarly for the estimated points $(x_i, \hat{y}_i)$ and $(x_{i+1}, \hat{y}_{i+1})$ get the trapezoid $(x_i, \hat{y}_i, \hat{y}_{i+1}, x_{i+1})$ and call as $\hat{T}$.

We measure the time to push the data by the sensor pair with respect to the difference in the area between the two trapezoids T and $\hat{T}$. This is reasonable because, as $(\hat{T}(t) - T(t))$ is minimum it implies that the estimates are close to the actual values and the sensor pair need not have to push the values. For the absolute error between $\hat{T}$ and T, we consider two design parameters one δ the tolerance parameter and the other ϵ the confidence parameter. Sensor pair is confident that they need not push sampled data if

$$Pr(|\hat{T}(t) - T(t)| < \delta) > (1 - \epsilon) \tag{1}$$

Equation (1) can be rewritten as

$$\Pr\left(|(\hat{y}_i(t+\tau) - Y_i(t+\tau)) + (\hat{y}_{i+1}(t+\tau) - Y_{i+1}(t+\tau))| < \frac{2\delta}{x_{i+1} - x_i}\right) > 1 - \epsilon$$

Hence, if t is the previous push time of sensors S_i and S_{i+1} then the next time to push can be at $(t + \tau^*)$ where τ^* is given by

$$\begin{aligned}\tau^* &= \arg\min_{\tau>0} \{\Pr(|Z(\tau)| < a) < 1 - \epsilon\} \\ &= \arg\min_{\tau>0} \{\Pr\left(\mathcal{N}(0, \sqrt{\sigma^2\tau}) \in (-a, +a)\right) < 1 - \epsilon\} \\ &= \arg\min_{\tau>0} \{1 - \Phi(a) + \Phi(-a) > \epsilon\},\end{aligned}$$

with

$$\begin{aligned}a &= \frac{2\delta}{(x_{i+1} - x_i)} \\ \Phi(x) &= \frac{1}{\sqrt{2\pi\sigma^2\tau}} \int_{-\infty}^{x} e^{-y^2/2\sigma^2\tau} dy.\end{aligned}$$

If the sensor pairs are not confident at time t, then they measure the contour point and compute μ and push the measured data and μ to the sink. This check is made after every ten unit of time. The detail algorithm executed at the i^{th} sensor is as in Algorithm (1). For $(i+1)^{th}$ sensor i.e., pair of the i^{th} sensor also it will be the same. This apply to all the sensor pairs.

Mean μ, variance σ^2 are computed as in [5] and correlation coefficient ρ in the interval $(t_i, t_i + \tau_i)$ can be obtained using the following equation.

$$\rho_{i,i+1} = \frac{\sum_{k=1}^{\tau_i}(y_i(t_i+k) - \bar{y}_i(t_i))(y_{i+1}(t_i+k) - \bar{y}_{i+1}(t_i))}{\sqrt{\sum_{k=1}^{\tau_i}(y_i(t_i+k) - \bar{y}_i(t_i))^2 \sum_{k=1}^{\tau_i}(y_{i+1}(t_i+k) - \bar{y}_{i+1}(t_i))^2}}$$

Here $\bar{y}_i$ is computed as $\bar{y}_i(t_i) = \frac{1}{\tau_i}\sum_{k=1}^{\tau_i} y_k(t_i)$. Simple exponential averaging will be used at the sink and neighbouring sensors as in [6] to estimate μ_i and σ_i^2 respectively.

A. Routing for the Distributed Model

We compared the performance of our distributed algorithm with that of centralized algorithm with respect to energy consumption for routing. In a centralized system [5], [6] for every query, the data and other parameters need to be pushed to sink directly from sensors. In a distributed system, sensors communicate with the neighbors and they schedule the routing path. One of the simplest routing is to forward through the shortest path from source to destination. We have opted Dijkstra algorithm to find the shortest route from sensors to sink. So, every sensor before pushing the data computes the shortest route using Dijkstra algorithm and push the data via that route. Power consumption is calculated for a link as distance square.

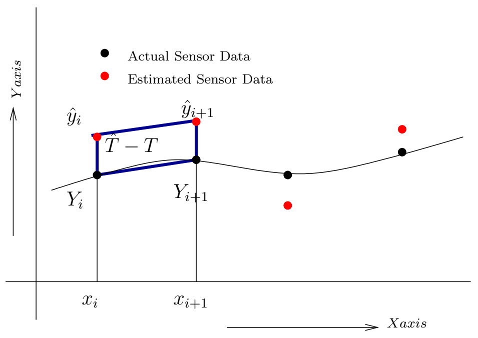


Fig. 2. Illustration for goodness of Estimation for a sensor pair.

Algorithm 1: This algorithm is executed by i^{th} sensor S_i paired with S_{i+1}. Condition to push the data to sink is checked at regular intervals of time . CT is current time and PTP is previous time of push. Sensor S_i at location (x_i, y_i) communicates with its neighbor S_{i+1} at location (x_{i+1}, y_{i+1}). If sensing rate is 1 time unit the time to push is computed for every 10 time unit

```
Input: Given tolerance parameter $\delta$ and the confidence
       parameter $\epsilon$
Initialization;
$CT \leftarrow 1$;
$PTP(i) \leftarrow CT$;
$Sample\ Y_i(CT)$;
$a \leftarrow (\delta * 2)/(x_{i+1} - x_i)$;
$Initialize\ \mu_i,\ \sigma_i^2,\ \sigma_{i+1}^2$ and $\rho_{i,i+1}$ with some
arbitrary values for $CT$
$PUSH(Y_i(CT), \mu_i(CT))$ to the sink;
for $CT \leftarrow 2$ to $\infty$ do
    $Sample\ Y_i(CT)$;
    $Communicate\ Y_i(CT)\ to\ S_{i+1}$;
    $Accept\ Y_{i+1}(CT)\ from\ S_{i+1}$;
    if ($CT \mod 10 = 0$) then
        $Compute\ \rho_{i,i+1}(CT)$;
        $\tau(i) \leftarrow CT - PTP(i)$;
        $\sigma_{i,i+1}^2 \leftarrow \sigma_i^2 + \sigma_{i+1}^2 + 2 * \sqrt{\sigma_i^2} * \sqrt{\sigma_{i+1}^2} * \rho_{i,i+1}$;
        $\Phi(a) \leftarrow \mathbf{N}(a, 0, \sqrt{\sigma_{i+1,i}^2 * \tau(i)})$;
        $\Phi(-a) \leftarrow \mathbf{N}(-a, 0, \sqrt{\sigma_{i+1,i}^2 * \tau(i)})$;
        if ($1 - \Phi(a) + \Phi(-a) > \epsilon$) then
            $Compute\ \mu_i(CT)$ and $\sigma_i^2(CT)$;
            $PUSH(Y_i(CT), \mu_i(CT))$ to sink;
            $Communicate\ \sigma_i^2(CT)$ to $S_{i+1}$;
            $PTP(i) \leftarrow CT$;
        end
    end
end
```

B. Contour Estimation At Sink

Let $\mathbf{x} := [x_1, \ldots, x_N]$ be the vector of the x-coordinates of the N sensor locations and let $\mathbf{Y}(t) := [Y_1(t), \ldots, Y_N(t)]$ be the corresponding Y-coordinates of the contour points at time t. Sink apply non parametric regression as in [2] on these sensors data and compute the contour points.

Non linear, non parametric regression technique is used to estimate the boundary points between the samples assuming that the boundary is smooth. There are two steps involved in applying regression. First, sink finds the nearest neighbors for the sample points using Epanechnikov kernel function and their weights are selected using the following formula.

$$W_T(x_i) = \begin{cases} 0.75[1 - (|x_i - x_0|/h)^2] & \text{if } |x_i - x_0|/h \leq 1 \\ 0 & \text{if } |x_i - x_0|/h > 1 \end{cases}$$

Here h indicates the half bandwidth of the window where all the N sensor nodes are the neighbors, and x_i are the sensor locations.

After selecting and assigning the weights for the nearest neighboring points, the second computation involves the boundary point estimation corresponding to the x-coordinate x_0 which is computed using the Nadaraya-Watson estimator

$$\hat{y}_0(t) = \hat{f}(x_0, \mathbf{x}, \mathbf{y}(t)) = \frac{\sum_{i=1}^{n} W_T(x_i) y_i(t)}{\sum_{i=1}^{n} W_T(x_i)}. \tag{2}$$

The detail steps of communication and computation at sink is as per the Algorithm (2)

Algorithm 2: This algorithm is executed by the sink. Here $Y = [Y_1, Y_2, \ldots Y_N]$ represents the sensor data of N sensors, Similarly PTP and τ are the vectors with the values for all N sensors. Sink estimates the sample points for every 1 time unit and apply regression and check for the sensor data for every 10 time unit

```
Initialization ;
$CT \leftarrow 1$;
for $i \leftarrow 1$ to $N$ do
    $PTP(i) \leftarrow CT$;
    $Y_i(PTP(i)) \leftarrow Y_i(CT)$;
    $\mu_i(PTP(i)) \leftarrow \mu_i(CT)$;
end
Apply regression and get contour points;
for $CT \leftarrow 2$ to $\infty$ do
    for $i \leftarrow 1$ to $N$ do
        $\tau(i) = CT - PTP(i)$;
        $Y_i(CT) \leftarrow Y_i(PTP(i))$ + $\mu_i(PTP(i))$ * $\tau(i)$;
    end
    if ($CT \mod 10 = 0$) then
        for $i \leftarrow 1$ to $N$ do
            if ($S_i(CT)$ available) then
                Accept $Y_i(CT)$;
                Accept $\mu_i(CT)$;
                $PTP(i) \leftarrow CT$ ;
            end
        end
    end
    Apply regression and get contour points;
end
```

III. Numerical Results

In this section we explain the simulation model that we have used to provide some numerical evidence of the efficiency of the distributed model that we have developed.

A. Simulation Model

N sensor nodes are uniformly randomly deployed from [0, M]; x(j) is the x-coordinate of the location of sensor j. The y-coordinate corresponding to these M discrete points are varied to simulate the dynamics of the boundary. The dynamic boundary points $\mathbf{Y}(t) := [Y_k(t)]$ is modeled as correlated Brownian motion with a constant drift with the known start and ending time as $[0, T]$. The constant drift $\boldsymbol{\mu}$ is M-dimensional vector with the value $\mu_k = (Y_k(0) - Y_k(T))/T$. The covariance parameter $\mathbf{C}$ is an $M \times M$ matrix with the following condition. $C_{i,i} = \sigma_k^2$ and for $i \neq j$, $C_{i,j} = d^{|i-j|}$ where d is a small constant. $Y(t)$ evolves as $\mathbf{Y}(t+1) = \mathbf{Y}(t) + \mathbf{N}(\boldsymbol{\mu}, \mathbf{C})$ where the $\mathbf{N}(\cdot, \cdot)$ is the M-dimensional Gaussian variable.

In the simulation process we have two sensors per cluster with say S_i and S_{i+1} where i and $i+1$ is a odd and even adjacent value with respect to its x axis position in the 2 dimensional plane. As explained earlier the sensor pairs predicts the time to push the data and push μ and samples to sink. The sink then uses simple exponential averaging to estimate the values. We discuss the result of a sample run for a sample deployment in Section III-D. The results for other runs are with very similar characteristics.

B. Performance Measures for Estimations

In this section we explain how we measure the performance of our distributed model.

We will first define the notation for the sample points. We have two set of sample points. First, the actual sensor points and the estimated sensor points by the sink. Let $Y_j(t) := Y(x_j, t)$, i.e., the point on the contour that will be sampled by S_j located at a position with x coordinate x_j. Here the actual sample points for N sensors are represented as $\{y^{(j)}(t)\}_{j=1,\ldots,N}$. Since $y^{(j)}(t)$ is not available at the sink at all times and it uses $\hat{y}^{(j)}(t)$ represented as $\{\hat{y}^{(j)}(t)\}_{j=1,\ldots,N}$ for N sensors. Note that some of these estimates may be the actual values pushed by the sensors at time t and others are estimated values by the sink. We measure the accuracy of estimation by computing the time average of mean square error between $y^j(t)$ and $\hat{y}^j(t)$ for the simulation time T,

$$ES := \frac{\sum_{t=1}^{T} \sum_{j=1}^{N} \left[y^{(j)}(t) - \hat{y}^{(j)}(t)\right]^2}{NT}$$

We now give the notation for actual contour and estimated contour which is discretised into M number of points. We represent the actual contour as $y_k(t)_{k=1,\ldots,M}$. In Fig. 3 the sample of the contour points Y_k is shown as C_1 and the contour obtained after applying regression on $y^j(t)$ is shown as C_2. The notation used for contour obtained after regression on $\hat{y}^j(t)$ is $\tilde{y}_k(t)_{k=1,\ldots,M}$. In Fig. 3 the sample of the contour points $\tilde{Y}_k$ is shown as C_3. Observe that $C3$ is reasonably close to C_2 which will be used by the sink as the estimated contour.

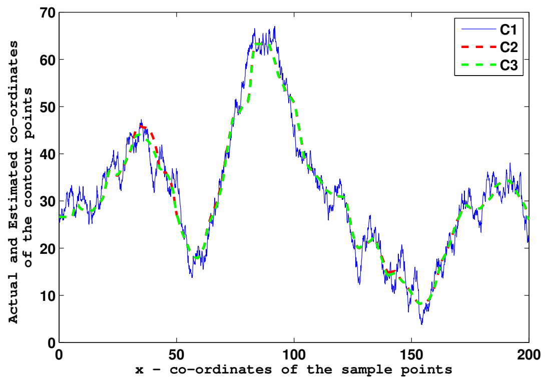


Fig. 3. C_1 is the actual contour points; C_2 is the a contour obtain from regression through $y^j(t)_{j=1,,N}$, and C_3 is the contour obtained from regression through $\hat{y}^j(t)_{j=1,,N}$. The result is for the value of $d = 0.98$ and $h = 6$. and after 2000 iterations with $\delta = 0.15$ and $\epsilon = 0.2$.

We measure the accuracy of estimation by computing the time average of mean square error between $y_k(t)$ and $\tilde{y}_k(t)$ for the simulation time T,

$$EC := \frac{\sum_{t=1}^{T}\sum_{k=1}^{M}\left[y_k(t) - \tilde{y}_k(t)\right]^2}{MT}$$

To prove the efficiency of our distributed model in reducing the communication cost we have measured the reduction in the total number of push made by the sensors compared to

the maximum. Thus if Q is the total number of push from all the sensors in the simulation interval T, then

$$P_Q := \frac{Q}{NT}$$

All these measures are along the lines of [6].

C. Energy Consumption for Routing

As mentioned earlier we compare the performance of our distributed model with the centralized model with respect to energy consumed for routing. First we will explain how we compute the energy consumption between a link connecting two sensor nodes. Let (x_i, y_i) and (x_j, y_j) be location of two sensor S_i and S_j on a two dimensional plane. Knowing the location we can compute the distance between the 2 sensors as $d = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$. The energy consumed for data transfer between S_i and S_j is defined as square of the distance i.e., d^2 and call it as D.

- Energy Consumption in Centralized Model: Let $S_Q = [s_1, s_2, \ldots, s_k]$ be set of sensors queried by the sink. The calculation of energy consumption is as follows. If k sensors are queried at time t, then all will communicate to the sink directly. Knowing the location of the sink and the sensors, the energy consumed to communicate data from each queried sensors to sink can be computed as explained above. If $D_{QC} = [D_1, D_2, \ldots, D_K]$ represents the energy consumed for the respective sensors of S_P to communicate directly to sink, then the energy consumed over time average for the simulation time T is

$$Eng_{CC} := \frac{\sum_{t=1}^{T}\sum_{k=1}^{K} D_k}{T} \tag{3}$$

- Energy Consumption in Distributed Model: Assume the network as a fully connected graph. Knowing the location of all the sensors and the sink, compute and get a weighted graph as explained above. For each sensor compute the shortest path to the sink. For simplicity we have considered Dijkstra algorithm to compute the shortest path i.e., the minimum energy path from source to destination. If k sensors need to push data to sink at time t then the sum of the energy consumed over time T is computed using (3) and call as Eng_{CD}.

D. Simulation Results

100 ranging sensors (N) deployed uniformaly randomly in a two dimensional plane of $200X200$ dimension. Sink is located at (100,100). Discrete M points i.e., 2000 points with respect to x-axis are the contour points. Simulation time T is discrete time from $[1, 2000]$. we compare the performance of our distributed model with that of centralized model [6]. We discuss the result with variance value $\sigma^2 = 0.01$, with $d = 0.98$ for the correlation matrix and h= 0.6 for non parametric regression. We have observed a quailtatively similar result for other values.

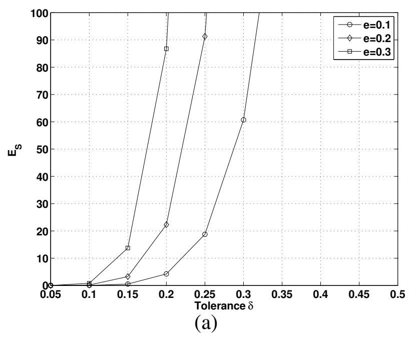
The performance of our distributed model measured with respect to mean square error in estimating sample points E_S, mean square error in estimating contour points E_C and the percentage of query P_Q is presented. The different performance measures as a function of δ (tolerance) for different ϵ (confidence) are shown in Fig. 4. Figs. 4(a)–4(c) are the results for our distributed model which is same as that of the centralized model. Figs. 5(a) and 5(b) are the results of energy consumption as a function of δ for different values of ϵ for a centralized and distributed models respectively.

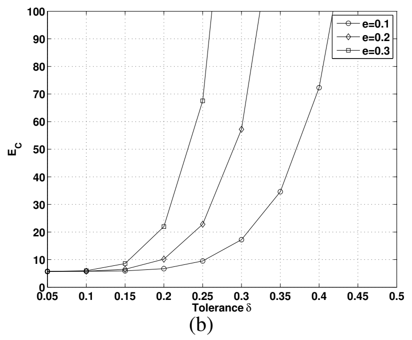
From Figs. 4(a)–4(b) we can see that as the tolerance δ increase beyond 0.2 contour tracking by sink start to worsen and this is because of the reduction in the number of data pushes from the observing sensors as seen in Fig. 4(c). We can also observe that as the δ increases the number of pushes from the sensor drastically reduces and hence the quality of tracking. This is because the value of $\hat{T} - T$ starts increasing with increasing value of δ and intern the probability of the estimate at the sensor pair being valid increases. Hence the tolerance parameter should not be too large.

Figs. 5(a) and 5(b) are the plots for average energy consumption for routing by the sensors to sink in the centralized system [6] and for our distributed model as a function of δ for different values of ϵ. Plots clearly show that the average energy consumed by centralized system where all the sensors communicate directly to sink is thrice the amount of average energy consumed by our distributed model.

IV. Conclusions

In this paper we propose a distributed push model to estimate and track dynamic boundaries. Contour evolves as





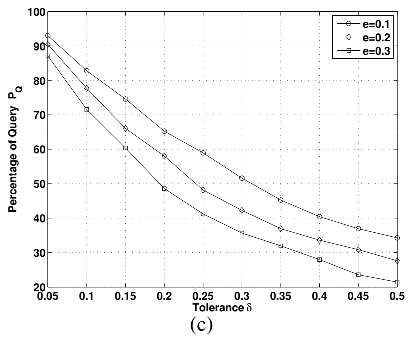
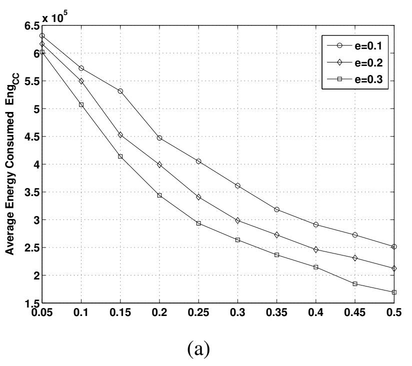


Fig. 4. Performance measures from a sample simulation as a function of the tolerance parameter (δ) for different values of the confidence parameter (ϵ) for centralized system and for adaptive refresh distributed model which are same. 4(a) Mean square error between $\hat{y}^{(j)}(t)$ and $y^{(j)}(t)$,4(b) Mean square error between $y_k(t)$ and $\tilde{y}_k(t)$ and 4(c) The number of pushes made by the sensors as a percentage of the maximum possible. The result is for the value of $d = 0.98$ and $h = 6$. and after 2000 iterations with $\delta = 0.15$ and $\epsilon = 0.2$.



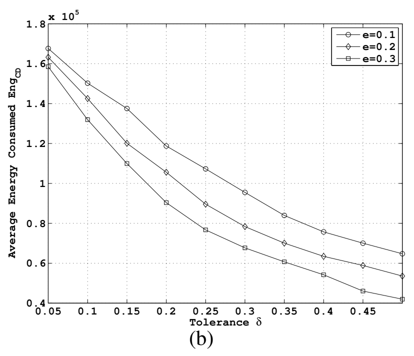


Fig. 5. Average energy consumed for routing data to sink as a function of δ for different values of ϵ. 5(a) Average energy E_{CC}, consumed for direction communication to sink by sensors in a centralized system, 5(b) Average energy E_{CD}, consumed for routing in distributed model. The result is for the value of $d = 0.98$ and $h = 6$. and after 2000 iterations with $\delta = 0.15$ and $\epsilon = 0.2$.

correlated Brownian motion with drift. The sensor pair in a cluster use the parameters of the evolving contour along with their spatio-temporal correlation to predict the time to push the sample data and μ values to the sink. This model parameter μ is used by the sink to estimate the contour location. The performance of our model in estimating the sample points, estimating the contour and in reducing the communication cost is measured. Distributed model is more energy efficient than the centralized model with respect to routing. Simulation study with different values show similar results.

V. Acknowledgement

The authors would like to thank and express deep gratitude to Professor D. Manjunath, IIT Bombay, for his valuable suggestions and guidance in carrying out this work.

References

[1] S. Duttagupta, K. Ramamritham, and P. Kulkarni. Tracking dynamic boundaries using sensor network. *IEEE Transactions on Parallel and Distributed Systems*, 22:286–290, October 2011.

[2] S. Duttagupta, K. Ramamritham, and P. Ramanathan. Distributed boundary estimation using sensor networks. In *Proc. of Third IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, pages 316–325, October 2006.

[3] Z. Jin and A. L. Bertozzi. Environmental boundary tracking and estimation using multiple autonomous vehicles. In *Proc. of the 46th IEEE Conference on Decision and Control*, New Orleans LA USA, 2007.

[4] A. Joshi, T. Ashley, Y. R. Huang, and A. L. Bertozzi. Experimental validation of cooperative environmental boundary tracking with on-board sensors. In *Proc. of American Control Conference*, pages 2630–2635, St. Louis MO USA, 2009.

[5] Nagarathna, Valli.S, and D. Manjunath. Using an evolution model for efficient estimation and tracking of dynamic boundaries. In *Proc. of TENCON 2012*, Cebu, Philippines, November 2012.

[6] Nagarathna, Valli.S, and D. Manjunath. A spatio-temporal model for estimation and efficient tracking of dynamic boundaries. In *Proc. of NCC 2014*, Kanpur, India, February-March 2014.

[7] R. Nowak and U. Mitra. Boundary estimation in sensor networks: Theory and methods. In *LNCS: Information Processing in Sensor Networks*, pages 80–95, 2003.

[8] A. Savvides, J. Fang, and D. Lymberopoulos. Using mobile sensing nodes for dynamic boundary estimation. In *Proc. of WAMES*, pages 2630–2635, Boston MA USA, 06 June 2004.

[9] A.Arora *et. al.* A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46(5), December 2004.