TRACKING MOBILE TARGETS USING SENSOR NETWORKS

Ahmed M. Khedr* and Walid Osamy
Department of Mathematics, Faculty of Science
Zagazig University, Zagazig, Egypt

الخلاصة:

تحتوي شبكة المستشعرات اللاسلكية على العديد من الوحدات ذات الطاقة المنخفضة و المزودة بوصلة لاسلكية للاتصال فيما بينها ، حيث إن تلك المستشعرات يمكن استخدامها في العديد من التطبيقات. وقد تم في هذا البحث توزيع المستشعرات بشكل عشوائي على المنطقة المراد مراقبتها و تقسيمها بشكل ما إلى مجموعات لتعمل كقواعد بيانات متفرقة، حيث تحتوي كل واحدة من تلك المستشعرات على بيانات مكانية عن الجسم المتحرك داخل المنطقة. وبعد ذلك تم بناء خوارزيم يعمل على توجيه و إدارة الشبكة لتتبع مسار حركة أي جسم يمر خلال الشبكة.

يتمثل الهدف الأساسي من البحث في جعل كل مستشعر من المستشعرات يقوم باستخلاص البيانات ذات الدلالة محلياً، و من تلك البيانات المستخلصة محلياً يتم اكتشاف الشكل العام لمسار حركة الجسم سواء كان مسار حركته خطياً أو غير خطي مع الأخذ في الاعتبار أن يكون عدد الرسائل المتبادلة بين كل مستشعر و آخر أقل ما يمكن، و ذلك لتوفير الطاقة المستهلكة داخل كل مستشعر أو داخل الشبكة بشكل عام. و في النهاية تم تحليل الخوارزيم بناء على عدد الرسائل المتبادلة بين المستشعرات و بعضها بعضاً.

* To whom correspondence should be addressed
E-mail: amkhedr@yahoo.com

Paper Received 1 March 2005; Revised 23 July 2006; Accepted 8 November 2006

ABSTRACT

Wireless sensor networks consist of a large number of low power devices equipped with *RF* links for communication that have numerous military, civil, and environmental monitoring applications. The energy constraints due to limited battery power present several design challenges. We have considered a random sensor network , where the entire network of these sensors acts as a set of distributed datasets. Each of these sensors has its local temporal dataset along with spatial data and the geographical coordinates of a given object or target. In this paper, a cluster based algorithm is proposed for managing and coordinating a sensor network for tracking moving targets by mining global temporal patterns from these datasets and results in the discovery of nonlinear trajectories of moving objects under supervision. The main objective here is to perform in-network aggregation between the data contained in the various datasets to discover global spatio-temporal patterns; the main constraint is that there should be minimal communication among the participating nodes. We present the algorithm and analyze it in terms of the communication costs.

Key words: wireless sensor networks, in-network aggregation, spatio-temporal patterns, distributed datasets, data mining.

TRACKING MOBILE TARGETS USING SENSOR NETWORKS

1. INTRODUCTION

Sensor networks are related to wireless ad hoc networks, which are self-organizing systems consisting of hosts that do not rely on the presence of any fixed network infrastructure. After deployment, the wireless sensors self-configure to gather acoustic, magnetic, or seismic information and send the information to a cluster head or a base station; here end-users can retrieve the data. Important messages (*e.g.*, an intruder detected by a sensor) can be broadcast to all nodes. Sensor networks are characterized by sheer numbers of nodes, data-centric applications, and constrained resources (bandwidth and energy).

Sensor networks are an important area of applications, where each sensor collects data in its vicinity, and within its interest, to determine global patterns in a geographically distributed collection of sensors. In this paper, we present a methodology for mining geographically distributed datasets for spatio–temporal patterns.

In our work, we assume a cluster based architecture for the sensor nodes that are randomly spread in a region to be supervised. They are capable of capturing and storing data. Our main aim is to perform data mining using our algorithm on the data stored in these sensor nodes. Data mining in sensor networks has a host of real time applications that can yield very useful, and profitable, results. For example, the following questions can be answered by implementing Data Mining techniques on the vast data available to us from a set of distributed databases in a state. Is there a pattern of increasing crime rate in the different counties of the state? How is this pattern spreading over the state and in which direction? Which were the most affected areas in the state by Hurricane Isabel? Which areas are most likely to be affected by a power outage?

For the purpose of our paper, we refer to sensor networks as a set of geographically distributed stand alone sensors, that are spread over a region, and are capable of sensing predefined parameters such as temperature, motion, and geographic coordinates of objects. Every sensor is capable of collecting information in its vicinity and exchanging it to its Cluster Head. We assume that the sensors we work with are capable of running self-decomposable algorithms that extract useful information from the data stored in the sensors. Upon collection of data, the main interest to us is to determine global patterns from these sensors. Mobility of the sensors is dependent on the application and its use. We have restricted our work in this paper to stationary sensors.

2. RELATED RESEARCH

There are many technical challenges associated with sensor networks, such as self-organizing algorithms, energy-efficient routing protocols, data analysis/mining technology and network lifetime improvements [1–4]. The source of energy for a wireless sensor is most often an attached battery. The power in sensor nodes can be used up simply by computations and transmissions. Furthermore, it is infeasible to replace thousands of nodes in hostile or remote regions. Therefore, conserving energy so as to prolong the network lifetime is becoming one of the key challenges for such power constrained networks. Recent research has addressed this topic, for example [5, 6].

One of the most important areas where the advantages of sensor networks can be exploited is for tracking mobile targets. Scenarios where such network may be deployed can be both military (tracking enemy vehicles, detecting illegal border crossings), and civilian (tracking the movement of wild animals in wildlife preserves). Typically, for accuracy, two or more sensors are simultaneously required for tracking a single target, leading to coordination issues. Additionally, given the requirements to minimize the power consumption due to communication or other factors, we would like to select the bare essential number of sensors dedicated for the task, while all other sensors should preferably be in the hibernation or off state. In order to simultaneously satisfy such requirements as power saving and improving overall efficiency, we need large scale coordination and other management operations. These tasks become even more challenging when one considers the random mobility of the targets and the resulting need to coordinate the assignment of the sensors best suited for tracking the target as a function of time. In this paper, we propose an algorithm for managing and coordinating a sensor network for tracking moving targets.

The problem of tracking targets with sensor networks has received attention from various angles. In [20], Galstyan *et al.* proposed a distributed online algorithm in which sensor nodes use geometric constraints induced by both radio connectivity and sensing to reduce the uncertainty of their position. In that algorithm, sensor nodes use online observation of a moving target to simultaneously improve both the moving target and their own positions. A small fraction of reference nodes are pre-planned or GPS placements into the network. In [21], Kirill *et al.* presented a two-level cooperative tracking algorithm using binary-detection sensors to track the object with more precision and accuracy.

In the first level phase, the local target position estimation is computed. Initially, the target is estimated to equal to the position of the sensor node. As more information from the other sensor nodes is available, the estimated position is recomputed as a weighted average of the sensor locations. Sensor nodes that lie closer to the path of the target receive more weight. These estimations are then aggregated to compute the path of the object, which produces a more precise estimate for the target location. In the second level, a piecewise linear approximation of the path is computed using a line-fitting algorithm on the positions obtained in the first level. In [24], Fang *et al*. proposed a collaborative computation approach where they count the number of targets in the sensing terrain. They equate the computation of the number of targets to the computation of leader nodes in the network with maximum signal strength. They aggregate this information to get a final report of the total count. The work described in [22, 23, 25] provides similar distributed collaborative algorithms for target localization, classification, and tracking.

The topographically addressed sensor nodes are similar to the way we have dealt with the placement of nodes in our work [7]. A distributed algorithm, which estimates the gradient of an environmental scalar variable such as temperature, intensity of light, atmospheric pressure, *etc*. using a random sensor network, is discussed by [8].

The work in [9, 10] has examined ways to provide in-network aggregation for the internet in a traditional way. The authors used the concepts of routers and centralized node. But in the practical world of sensors and other similar devices, it is very expensive for a central node to communicate with all other nodes in the system. Hence a hop by hop network is built where each node can talk with its neighboring nodes [11]. In this aspect, the central idea of our paper is similar to [11], but we do not have the concept of router or centralized node. We assume that our algorithm would be primarily used for applications where the bandwidth is limited and the communication costs between nodes are very high, but the computation cost at the local nodes is very less.

In [12], an algorithm has been designed to carry out in-network processing for data aggregation using a dense deployment of nodes. The main aim of our paper is similar to [12], where our proposed algorithm is used to compute data aggregations over a system of nodes by using in-network processing of the data stored in the nodes as in [12] but involving a temporal attribute, and also, the adopted methodology is different in that they assume two-way flow of data between nodes.

In [13] an algorithm is designed that exploits the characteristics of ad hoc wireless sensor networks to discover position information of the sensor nodes even when they have been sprinkled all over the earth. Our work differs from [13], in that the sensor nodes in our work do not have to know the global knowledge of the topology of their physical locations, but they do need to know their global coordinates to start the in-network data analysis, and aggregation.

Knowledge discovery and data mining are emerging fields, whose goals are to make sense out of large amounts of data collected, by discovering hitherto unknown patterns. Many interesting and efficient data mining algorithms have been proposed [14–16]. These database-oriented mining algorithms can be classified into two categories: concept generalization-based discovery and discovery at the primitive concept level. Generalization of attribute values (or concepts) is the main idea in the former, whereas the latter discovers strong regularities or association rules from databases without concept generalization.

There has not been much work in the area of mining temporal concepts. Most of the existing work is based on time series analysis of temporal sequences. Reference [17] discusses some of the challenges posed by the temporal data. There has been some work done in [11] in the field of parallel algorithms for temporal aggregation. In this paper, we discuss an algorithm which discovers temporal patterns among distributed datasets.

The rest of the paper is organized as follows: In the following section, we give an introduction to the basic concepts. A step by step outline of our algorithm is described in Section 4. We describe the trajectory of tracking in Section 5. In Section 6, we give an example scenario. Section 7 analyzes the complexity of our algorithm. In Section 8, we present our simulation results. In Section 9, we discuss the practical matter. We conclude our work in Section 10.

3. PROBLEM FORMULATION TERMS

We consider sensor nodes that randomly spread across a geographical area and collaborate among themselves to establish a random sensor network. Also we assume a cluster based architecture for the sensor network. Each cluster has only one leader called Cluster Head (*CH*) and some nodes that can communicate with other neighbor clusters called Gateways (*GW*), and the remaining nodes in the cluster called Cluster Members (*CM*). Our development is in the context of temporal data being recorded by a number of sensors. We outline some of the ideas used in the development of our algorithm. The dataset used in our algorithm consists of the x and y coordinates of the points at which the light event is sensed, along with the timestamp

[26]. We refer to the term point to a combined set of information about a moving object which includes *x, y* coordinates and the timestamp. Each sensor may have a number of such data points recorded in its local memory.

3.1. Local Hypothesis (*LH*)

Forming of Local Hypotheses is the first step of our algorithm that is implemented in each of the sensor nodes. A Local Hypothesis is a set of three or more points, satisfying the following criteria.

1. Taking sets of three points or more, they should lie on the same line in the same direction;
2. The points in the *LH* should be in ascending timestamp manner.

The angle between two points p_1, and p_2 can be computed by the following equation:

$$Angle(p_1, p_2) = \arctan\left[\frac{Ycord(p_2) - Ycord(p_1)}{Xcord(p_2) - Xcord(p_1)}\right] \quad (1)$$

We define the points p_1, p_2, p_3, and p_4 to be lying in the same direction if the angle between any pair of points is the same as the angle between the other pair, *e.g.*, in Figure 1, the points p_1, p_2, p_3, and p_4 lie in the same direction. Hence they make a Local Hypothesis as is indicated by a rectangular box in Figure 1.

LH is considered as a straight line that starts at the first point (*FP*), and ends with the last point (*LP*), with a specific angle of slope. For that, we consider the structure of *LH* as: *LH* = (*FP, LP*, angle). In Figure 1, LH_1 has p_1 as the first point , and p_4 as the last point .

3.2. Global Hypothesis (*GH*)

Forming *GH* is the second step of our algorithm. A *GH* is considered as a structure that contains set of *LHs.* A *GH* is formed when *LHs* from different sensor nodes are merged in ascending manner according to their timestamps. We keep track of *GH* direction by updating the *PointChangeList* which includes the points at which the *GH* changes its direction.

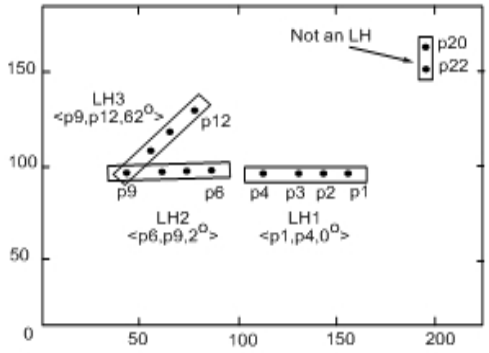

Figure 1. Example of LHs at one node

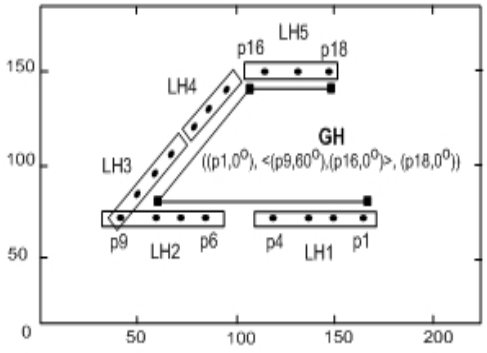

Figure 2. A GH Example at a cluster head

The *GH* starts at the first point of the first attached *LH* and ends at the last point of the last attached *LH.* We define the length of *GH* to be the number of *LHs* considered during forming *GH.* We define the *GH* structure as *GH* = (Start Point (SP), <*PointChangeList* >, End Point (*EP*))**.** In Figure 2, the solid black line is the *GH,* and the small boxes represent the points at which the *GH* changes its direction.

4. ALGORITHMS OUTLINES

There are two stages of our algorithm: the first stage will be executed at every sensor node in the system to generate the *LHs* (**Local Computation**), while the second stage will be executed at every cluster head that receives *LHs* from its cluster members to generate the *GH* (**Global Computation**).

4.1. Algorithm Assumption

We assume a cluster-based architecture for the sensor network and the choice was motivated by the need to ensure the sensor network's scalability and energy efficiency. A number of clustering algorithms have been proposed in the literature [18, 19]. In this paper, we are not concerned with details of clustering mechanisms. Also, we assume that the cluster head knows the sensor identities and locations for all sensors belonging to its cluster. The assumptions about the sensor network

are the following:

1. All sensors have the same characteristics;
2. Sensors are randomly distributed across the whole sensing area;
3. All sensors have the capability to capture the information of any moving object in their sensing range. The information includes the approximate *x, y* coordinates, and the timestamp.

No specific assumptions are made about the movement pattern of the target. However, we assume that the targets originate outside the sensing region and then move inside. Also, the aggregated data are reported to the end user.

4.2. Local Computation

Each node in the system collects all the required information about the moving target in its range, and form the *LHs* by executing the following procedure.

1. Output**:** Set of Local Hypotheses
2. Each node in the system, initially reads the data (say *N* points) from its database, and sorts it in ascending order according to their timestamps.
3. $k = 1, i = 1$ // k is the current *LH* number and $i = 1$ is the current point number
4. while($i < N$)

(a) Find the angles $Angle1 = angle(p_i, p_{i+1})$ and $Angle2 = angle(p_{i+1}, p_{i+2})$

(b) if ($Angle1 = Angle2$)

i. $LH_k.FP = p_i$; $LH_k.LP = p_{i+2}$;

$LH_k.angle = Angle2$. ii. for $j = i+3$ to N

A. if $angle(p_{j-1}, p_j) = angle(p_{i+1}, p_{i+2})$

- $LH_k.LP = p_j$;

B. else

- $i = j, k = k+1$
- GOTO Step (4)

C. End if

iii. End for

(c) else $i = i + 1$

(d) End if

5. End while
6. End Local Computations

Initially, each node arranges the recorded points in ascending manner according to their timestamps line 2. Take the first three points and find the angle between the first and the second point, and the angle between the second and the third point (line 4 (*a*)). In line 4 (*b*), if the computed angles in 4 (*a*) are the same, the *LH* is established by setting up the first point as the *FP* of the *LH*, the third point as the *LP* of the *LH* and *LH* angle will be the angle between the second and the third point. For further points that may be added to the current *LH,* we take the next point p_j and determine its angle with the last added point to the current *LH*. If (p_{j-1}, p_j) angle equal to the *LH* angle, we update *LH.LP* to p_j (line 4 (*A*)). In line 4 *B*, if the first three points are not at the same angle, we skip the first point of the three points, and start from the second one (line 4 (*c*)).

4.3. Global Computation

In the second stage of our algorithm, every cluster member sends its *LH*s to its cluster head to form the *GH* component, then each cluster head will find the next suitable cluster head to send its *GH* component for merging. The formed *GH* after merging includes the main points of the trajectory along current cluster head and its previous one in the network. Also, in this stage *CH* communicates only with cluster heads that are in the direction of the event through the selected *GW*. The length of the *GH* component increased from *CH* to next *CH* till we obtain the summarized *GH* that contains all the useful points about the moving object trajectory. Finally, the end user will receive the summarized *GH*

that includes the start point, the end point, and the main points of the object trajectory through the network.

In order to describe Global Computation step, we first introduce the notation of "*Event*". The *Event* occurs at *CH,* if it receives any message contains an *LH* from its cluster members. The default status for each *CH* is no *Event* at *CH* (*Event= false*).

We now describe how Global Computation step is executed at every cluster head CH_i. After receiving the first *LH,* the *Event* is set to true then CH_i waits a period of time to receive any more *LH* messages. After the time has expired a sequence of tasks in the following order are carried out at every cluster head:

- Execute **GenerateGH** procedure to generate your *GH* component,
- Search for the *GW* which has minimum distance from the last point in your *GH* component,
- If the received message contains *GH* component , merge the received *GH* into your *GH,*
- Report the *GH* components to the next *CH* (CH_{i+1}), which has true *Event* or to the end user,
- Reply to query message that asks for *Event* status.

After receiving *LH*s, CH_i executes the **GenerateGH** procedure to generate its *GH* component, and CH_i searches in its database for the *GW* which has minimum distance from the last point in its *GH* component. Then CH_i communicates with the selected *GW,* and sends through it a query asking the next *CH* (CH_{i+1}) for *Event* and waits for response. If no response back or no *Event* message, CH_i marks the selected *GW* as already selected before and repeats searching for *GW*. If CHi_{+1} responses back that *Event* occurs in its direction, CH_i will send its *GH* component to CH_{i+1} through the selected *GW*. However, if CH_i could not find any *GW* (*i.e.* the object moves out from the network), in this case, CH_i has responsibility to report the *GH* to the end user.

If CH_i receives a message from different *CH* contains the *GH* component, CH_i will merge the received *GH* with its *GH* component as follows: Initially CH_i checks if $GH_{i-1}.EP$ timestamp (the received *GH* component) is less than $GH_i.EP$ timestamp (the current *GH* component of CH_i), the new GH_i starts at the start point of GH_{i-1} ($GH_{i-1}.SP$) and ends at the end point of GH_i ($GH_i.EP$), and the new *PointChangeList* ($GH_i.PointChangeList$) will be the union of the two sets $GH_{i-1}.PointChangeList$, and $GH_i.PointChangeList$. Finally, if CH_i receives any query message, it will *response* back by yes if there is *Event* otherwise, by no.

4.3.1. Procedure GenerateGH

This procedure generates *GH* from a list of *LHs*.

1. **Input:** List of *LHs*
2. **Output:** *GH*
3. Arrange the received *LHs* in ascending manner according to their timestamps.
4. set $i = 1$, // i is the current *LH* number
5. $GH.PointChangeList = \Phi$
6. $GH.SP = LH_1.FP$,
7. $angle = LH_1.angle$.
8. for every LH_i in the arranged *LHs* list ($i > 1$)
9. if ($angle \neq LH_i.angle$)
10. $GH.PointChangeList = GH.PointChangeList \bigcup < LH_i.FP, LH_i.angle >$
11. $angle = LH_i.angle$
12. end if
13. end for
14. $GH.EP = LH_i.LP$
15. End GenerateGH procedure

In line 3, the *LHs* will be arranged in ascending manner according to their timestamps. The *PointChangeList,* the start point *(GH.SP),* and the angle *of GH* will be *initialized* to *Φ,* the first point of LH_1, and $LH_1.angle$ respectively, in lines 5,6, and 7. In lines 8 to 12 if the current *LH* angle does not match the previous *LH* angle, then the first point and the angle of the current *LH* will be added to the *PointChangeList* of *GH* and then continue with the next *LH*. Otherwise continue with next *LH*. Finally the end point of the *GH* (*GH.EP*) will be the last point of the last *LH* in the list (line 14).

5. TRAJECTORY DESCRIPTION

The end user will have the summarized *GH* as a set of points (<*x, y, time*> in 3–dimensional *location-time*), and the objective is to represent the trajectory of the moving object at the end user.

A trajectory can be represented by a sequence of connected segments each of which joins two consecutive reported points: *i.e.* the start point of the reported summarized *GH* is connected by line segment to the first point in the *PointChangeList,* and the first point in the *PointChangeList* is connected by line segment to the next point till the last point in the *PointChangeList*, which is connected with the end point of the reported summarized *GH*.

To produce these segments, one way is to use the interpolation schemes (Line Based Models). These schemes create trajectories that have angles at reported locations which do not represent well the smooth trajectories of moving objects.

The second representation used is the curve-based trajectory representation model using Catmull–Rom spline [27] which provides much more accurate trajectories than line-based models when we have the same amount of data. One of the features of using Catmull–Rom spline is that the specified curve will pass through all of the control points and this is not true of all types of splines.

In the curve-based trajectory representation model, we represent the trajectory by a sequence of curve segments, rather than line segments, each of which connects two consecutive points, where as most natural moving objects, such as airplanes, vessels, and vehicles, draw a smooth trajectory with no angles.

The parametric form of a third-order polynomial to obtain a spline is given by the following equation.

$$P(t)= a_0+a_1 t+a_2 t^2+a_3 t^3 \tag{2}$$

where a_0, a_1, a_2, and a_3 are constant coefficients. Theses coefficients are determined from several equations that reflect the properties of the cubic spline. To calculate a point on the Catmull–Rom spline curve, two points on either side of the desired point are required. The point is specified by a value *t* that signifies the portion of the distance between the two nearest control points.

Given a *GH* of four points p_0 ,p_1, p_2, *and* p_3. The Catmull–Rom spline in matrix form will be (for point *q* on the curve at time *t*):

$$q(t) = 0.5 * \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \cdot \begin{bmatrix} -1 & 3 & -3 & 1 \\ 2 & -5 & 4 & -1 \\ -1 & 0 & 1 & 0 \\ 0 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} p_0 \\ p_1 \\ p_2 \\ p_3 \end{bmatrix} \tag{3}$$

This becomes: $q(t) = 0.5 * ((-p_0 + 3*p_1 -3*p_2 + p_3)*t^3$

$$+ (2*p_0 -5*p_1 + 4*p_2 - p_3)*t^2 + (-p_0+p_2)* t + 2*p_1).$$

t takes values between 0 and 1, and the curve passes through p_1 at $t = 0$ and p_2 at $t = 1$. To do more than two points, we can step through the array of points using the previous point, the current point, and the next two points as the four points for the spline. For each segment, we can draw a curve for $0 < t < 1$. This curve will be between the current point and the next point.

6. EXAMPLE SCENARIO

In this section, we provide a simple example with a step by step explanation to describe our algorithm. Assume that we have a random sensor network as shown in Figure 3. The arrows show how the object goes thought the network. In the first phase, data is collected and each sensor arranges its data according to their timestamps.

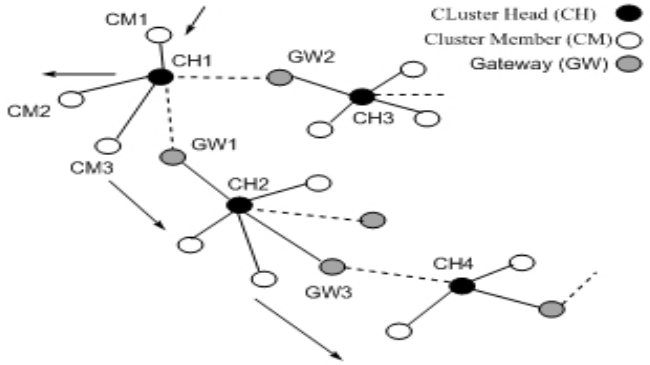

Figure 3. Random Sensor Network

We will limit our description to CH_1 and its cluster members. The CM_1 has the points $p_1, p_2, , p_3, p_4$, with angle 56.4°, and p_5, p_6, p_7, with angle 0°, CM_2 has the points $p_8, p_9, p_{10}, p_{11}, p_{12}$, with angle *0°*, and *CM3* has the points $p_{14}, p_{15}, p_{16}, p_{17}, p_{18}, p_{19}, p_{20}$ with angle 290°. According to our algorithm we will have $LH_{1=} < p_1, p_4,$ 56.4° >, $LH_2 = < p_4, p_7,$ 0° > at CM_1 , $LH_3 = <p_8, p_{12}, 0° >$ at CM_2, and $LH_4 = < p_{14}, p_{20}, 290° >$ at CM_3 as shown in Figure 4. Figure 5 shows the created *GH* component at CH_1 (GH_1).

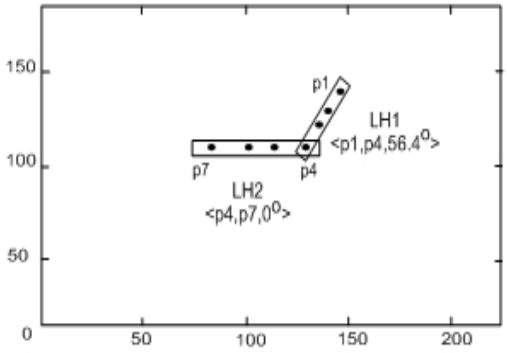

Figure 4. LHs at CM_1

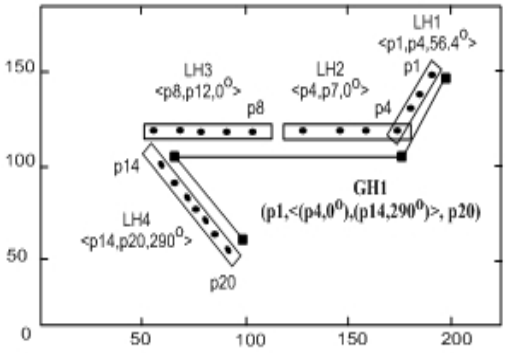

Figure 5. GH_1 at CH_1

From Figure 5. the start point of GH_1 is p_1, and the end point is p_{20}. The two points $< p_4, 0° >$, and $< p_{14}, 290° >$ will be added to the *PointChangeList* of GH_1. The next step is to send the formed GH_1 at CH_1 to the next suitable cluster head under the criteria we mentioned before. CH_1 will choose GW_1 as the nearest *GW* to $GH_1.EP$. But if an error occurred and CH_1 selected GW_2 *simply*, GW_2 will response back with no event, since CH_3 has no event. CH_1 will mark GW_2 as selected before and select GW_1. CH_2 responses back by *yes* confirmation and it is ready to receive the *GH* component from CH_1. CH_2 merges GH_1 and GH_2 as shown in Figure 7, the result of merging the GH_s in Figures 5 and 6. CH_2 will select GW_3 and send query to CH_4. CH_4 responses back by *yes* confirmation. Finally, CH_4 updates its GH_4 by merging it with the received GH_2. The end user will receive the characteristic of the trajectory curve from the received *GH*.

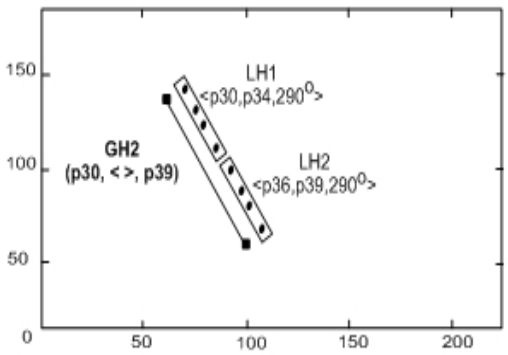

Figure 6. GH_2 at CH_2

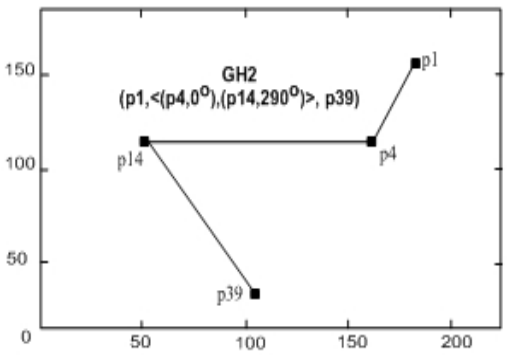

Figure 7. The updated GH_2 at CH_2

7. COMPLEXITY ANALYSIS

In this section, we present the complexity analysis in terms of number of exchanged messages between the nodes in the network. If we assume that:

- k is the number of cluster heads,
- g is the number of gateways,
- c is the number of cluster members.

7.1. Local Computation Cost

Each node in the system analyzes the collected data to extract the local patterns, since the extraction of the local patterns or forming the *LH* occurs locally, therefore the number of exchanged messages for this step will be zero.

7.2. Global Computation Cost

In the second stage of our algorithm,

- Each sensor node forwards its *LHs* to its *CH,* which takes communication cost equal to the number of cluster members at each cluster that detects the event, *i.e.*, the number of messages will be at most $g + c$.
- Each cluster head in the system analyzes the collected *LHs* to form its *GH*. Since the forming of the *GH* occurs locally, therefore the number of exchanged messages for this step will be zero.
- Each *CH* searches in its list of database for the *GW,* that has minimum distance from the last point in its *GH.* To reduce the amount of useless data traffic through the network and so the consumed energy in transmitting useless data we use a *yes/no* query before sending any data that contains the *GH.* Where a *yes/no* message requires only one bit of data size, while the *GH* component message requires many bits of data size. Here we consider two cases:
- ***Best Case* scenario**, the best case will be occurred when each cluster head that has event, selects the correct gateway that is in its event direction as the first selection to report its *GH*, in this case the number of messages will be $2*k + k = 3*k$ (k *yes/no* messages , k responses for *yes/no* messages, and k messages for sending the *GH*s). Therefore the total number of messages of the algorithm will be $3*k + g + c = 2*k + n$, where n is the total number of sensor nodes.
- ***Worst Case scenario,*** the worst case will occur when each cluster head selects all of its gateways before selecting the correct one to report its *GH.* In each selection, every *CH* sends *yes/no* query to the selected *GW,* and receiving response for *yes/no* query, and after selecting the desired *GW,* only one message contains the *GH* component will be sent, *i.e.* $2*g + k$ messages will be exchanged by all cluster heads. Therefore, the total number of messages of the algorithm will be $2*g + k + g + c = 3*g + k + c = 2*g + n$, where n is the total number of sensor nodes.

8. SIMULATION RESULTS

The simulation program is an event-driven simulator developed using Visual C#.Net. The simulation area is configure as a square region where sensor nodes, varying in number from 100 to 400 in increments of 100, were uniformly randomly distributed. The area of the square region varied with the number of sensors so as to keep the average density of nodes, *i.e.* nodes per square area, constant. The sensor nodes communicate with each other by broadcasting messages across channels which are assumed to be symmetric. The sensing range of the sensor nodes is set to 20 *m* and can be tuned in the program. The communication range of each sensor node is set to approximately twice of sensing range and that is to ensuring that two sensor nodes with overlapping sensing area are capable of communicating directly with each other. In context of our simulations, all the sensors in the network have the same characteristics, can act as *CH*s or *GW*s, and capable of storing data and performing computations on this stored data.

For clustering and gateway formation in our simulation, each node performs neighbor discovery by exchanging advertisements. It then waits for a time inversely proportional to its neighbor density and energy, before declaring itself as a leader (cluster head). Each cluster head sends out declaration messages with a list of the potential members and waits for some time to hear from them. The nodes that receive this declaration give up the election process and join the cluster by sending confirmation to the cluster head. The next step in the creation of a connected network is the formation of gateways between various stand alone clusters. Gateway selection is made conditional to the availability of an appropriate schedule for communication between the two clusters. The clusters go either in wait mode or search mode. During the search mode, the cluster head, CH_i requests all its non-gateway members to search for neighboring cluster

heads. Each cluster member sends periodic broadcast messages in the search mode to discover a gateway. On being contacted by a neighboring cluster head CH_j (in wait mode), it confirms its gateway status to CH_j, and notifies its parent CH_i about the link formed between itself and CH_j. During the wait mode, CH_j waits to hear from cluster members in search mode. Upon listening to a broadcast from *CM*, CH_j checks if it does not already have a gateway to the cluster CH_i. If CH_j does not have such a gateway then add CH_i to its neighbor list and *CM* becomes a gateway node between CH_i and CH_j [18]. The output of the algorithms for discovering the nonlinear trajectory or the global patterns of a moving object is shown in Figure 9 and Figure 10 with Figure 8 shows the actual trajectory of moving object through the network of 300 nodes that are capable of collecting and recording approximate coordinate information about the moving object.

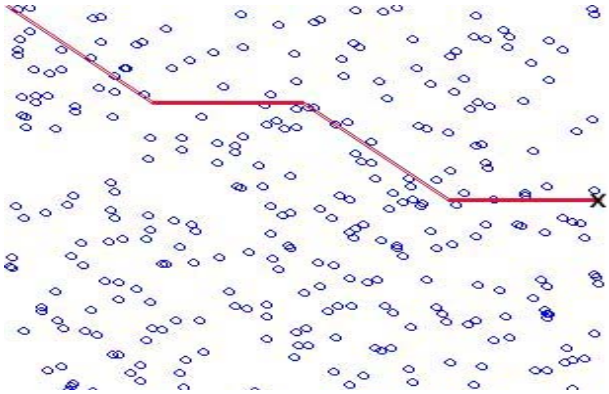

Figure 8. The actual trajectory of moving object approach

The main points of the global summarized *GH* are shown in both Figure 9 and Figure 10 by small red rectangles which are the start point, the *PointChangeList* (P_0, P_1, P_2), and the end point of the *GH*.

Figure 9 shows the estimated trajectory which obtained by our algorithm. This figure presents the trajectory representation of the reported summarized *GH* as sent to the end user, using a curve-based model, *i.e.* Catmull-Rom spline curve; the *PointChangeList* (P_0, P_1, P_2) contains the control points which the Catmull–Rom spline curve passes through to give a smooth representation for the object trajectory.

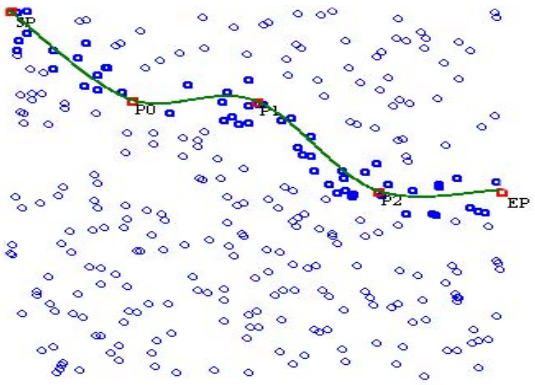

Figure 9. The Global Summarized GH Trajectory using Curve Model Representation

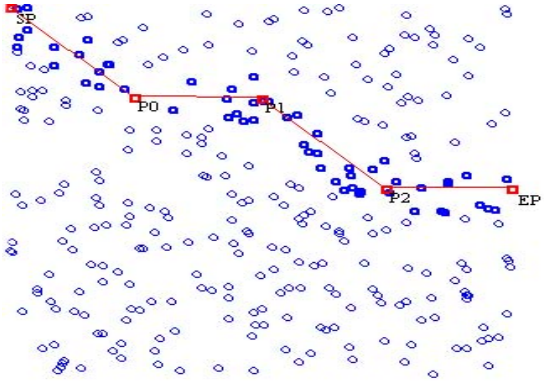

Figure 10. The Global Summarized GH Trajectory using Line Segment Representation

The thin red line in Figure 10 shows the representation of the estimated trajectory using line segment representation, where the start point (*SP*) in the reported summarized *GH* is connected by line segment to the first point (P_0) in the *PointChangeList*, the first point in the *PointChangeList* is connected by line segment to the next point (P_1), P_1 is connected by line segment to *P2,* and *P2* is connected by line segment to the end point (*EP*) of the reported summarized *GH*.

Simulations were separately conducted to analyze the communication complexity of our mechanism. The aim of these simulations is to show the proper working of the algorithm and that the algorithm is scalable to varying number of sensor nodes. At the simulation experiment using 100 of sensor nodes, the number of messages noted till the global summarized *GH* is obtained. This was repeated for different scenarios of 200, 300 and 400 node networks. In each case the number of exchanged messages for the different scenarios was computed. We noted that the best case will be occurred when each cluster head that has event, selects the correct gateway that is in its event direction as the first selection to report its *GH*, and the worst case will be occurred when each cluster head selects all of its gateways before selecting the correct one to report its *GH* . Figure 11 shows the total number of exchanged messages in the worst and best cases. The results were taken for different network sizes at which the event occurs in all of its clusters.

Our complexity analyses show that, there are two cases in our algorithm. The best case in which the total number of exchanged messages approximately equals twice of the total number of cluster heads plus the total number of nodes in the network, as shown in Figure 11, the number of exchanged messages is greater than the total number of sensor nodes. The second one is the worst case in which, the number of exchanged messages equals twice of the total number of gateways plus the total number of node in the network. It is obvious, when the size of the clusters in the network becomes large, the total number of cluster heads decreases which leads to the total number of exchanged messages in the network decreases to close to the total number of sensor nodes in the network. Our simulation results show that, the performance of our algorithm is satisfactory overall and objective. In addition, our results prove that the algorithm works similarly irrespective of the number of nodes in the network. In other words, the algorithm is scalable for increasing the number of sensors.

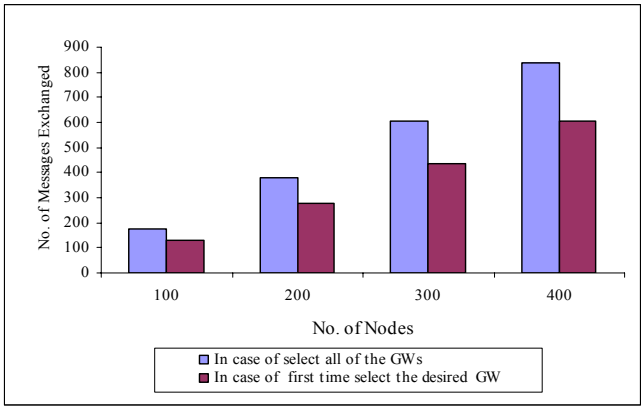

Figure 11. Total number of messages with increasing number of nodes

9. COMMENTS ON PRACTICAL MATTERS

Our algorithm described above works well in ideal situations. But in practical scenarios, wireless networks are prone to message losses, node failures *etc*. due to various environmental disturbances, and hardware restrictions at the sensors. This section considers such situations, and describes how these issues can be dealt with to enable proper working of the algorithm.

- **Node Failure.** Node failure can occur due exhaustion of battery life at the sensors. Such a situation may occur during exchanging *LH* messages or/and after the *GH* has been generated. In the former case, it would not be a problem because the neighbors of the current node might have the target information over this area. In the latter case, the *CH* needs to exchange its *GH* to its next *CH* through a *GW* so if the failure occurs in the *GW* this leads to a problem in reporting the *GH*. The *CH* should infer the death of its gateway so that the *CH* can retrace a different *GW,* which connect the next *CH* in this direction before sending its *GH, i.e.*, the *CH* should make sure that its cluster is in valid situation all the time and this could be by running a cluster maintenance algorithm.
- **Message Loss / Collision.** Message losses/collisions are very common in applications involving wireless communication because of the frequent environmental disturbances in the channel. Such situations are very difficult to detect at the sending sensor as it is not sure whether the message has reached its neighboring or intended sensor. Thus, to enable detection of messages losses at sending sensor, an acknowledgment signal can be used from the neighboring sensor to indicate proper reception of the message. If the sensor dose not receives

such an acknowledgement signal after a time limit, then it can retransmit the message to enable proper working of the algorithm.

- **Communication Reliability.** In our setting, we assume that there is communication reliability in the network.
- **Localization.** We would like the sensors to determine their own positions after placement. This is known as localization, and is typically achieved by having each sensor compute range measurements to its neighboring sensors, then algorithmically embedding the graph formed by these ranges into a coordinate system. This coordinate system is then used to perform location-dependent tasks such as geographic packet routing or target tracking. Localization is often complicated by the difficulty of obtaining enough accurate pair-wise range measurements between sensors. Inter-sensor ranges can be corrupted by noise or lost entirely due to occluded line-of-sight. Thus, consistently accurate localization requires robustness in the face of missing or low quality measurements. Nevertheless, localization is rarely if ever the purpose of a network. Sensor networks are typically deployed to observe active phenomena in the environment, and require accurate localization as a means to that end. As a result, there is pressure in localization research to achieve accuracy and robustness using as little hardware as possible. Target tracking is one of the motivating localization-dependent applications of sensor networks. In tracking applications, sensors jointly observe phenomena, which may be people or objects passing through the network or physical effects such as bullet shock-waves or anomalous sounds. Once a phenomenon is detected, the sensors collaborate to determine its spatial location. This estimate is reported to a computer or person monitoring the network. In our algorithm, we consider that each sensor node is aware of its own location. Where given the locations of the sensors and accurate range information to the target, it is straightforward to determine the target's position based on the discovered sensor positions. Consequently, a localization error in our case depends on the specialized localized algorithm which handle localization problem.
- **Calibration.** In an ideal world, sensors would arrive from the factory fully calibrated and able to begin taking accurate measurements of their surroundings. However, this ideal situation is rarely achieved. For instance, deployment conditions such as temperature affect the accuracy of ranging algorithms based on acoustic time-of-flight by altering the speed of sound. The differences between sensors can also result in mis-calibrations that are difficult to correct before deployment. Calibration in the field can therefore offer meaningful improvement in both localization and target tracking accuracy. As with localization, there is considerable economic incentive to develop auto-calibration algorithms that allow sensors to self-calibrate in the field without external intervention.

10. CONCLUSION

In this paper, we proposed a feasible solution for distributed tracking of mobile targets using random sensor networks; we have demonstrated a new algorithm for discovering temporal patterns from a set of distributed databases containing temporal data. The concept of maximizing the computations at the local sites and minimizing the exchange of information between the nodes helps reduce the load on the overall network, this formed the crux of our algorithm. In our work, we considered the problem of mining temporal data in distributed datasets. We worked with sensor nodes, which were capable of capturing and storing approximate coordinate information about a moving object or a target object. These nodes were placed in a random-like manner and our algorithm was used to predict the nonlinear trajectory of the moving object. We considered this equivalent to mining of global spatiotemporal patterns from geographically distributed datasets. Since there is only one database scanning required in the beginning, the complexity of our algorithm reduced.

Our simulation results show that the tracking performance is satisfactory overall, with particularly good performance at higher tracking resolutions. In addition, this algorithm is specifically aimed at minimizing the energy consumption of the network, a very important consideration for sensor networks.

REFERENCES

[1] M. Bhardwaj, T. Garnett, and A. P. Chandrakasan, "Upper Bounds on the Lifetime of Sensor Networks", *Proceedings of IEEE ICC'01*, *Helsinki, Finland*, 2001, p. 785.

[2] C. Intanagonwiwat, R. Govindan, and D. Estrin., "Directed Diffusion: a Scalable and Robust Communication Paradigm for Sensor Networks", *Proceedings of ACM MobiCom'00. Boston, MA, USA*, 2000, p. 56.

[3] A. Manjeshwar and D. P. Agrawal., "TEEN: A Routing Protocol for Enhanced Efficiency in Wireless Sensor Networks", *Proceedings of the IPDPS Workshop on Issues in Wireless Networks and Mobile Computing. San Francisco, CA*, 2001, p. 2009.

[4] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie,"Protocols for Self-Organization of a Wireless Sensor Network", *IEEE Personal Communication*, 2000, p. 16.

[5] J. H. Chang and L. Tassiulas, "Energy Conserving Routing in Wireless Ad Hoc Networks", *Proceedings of IEEE INFOCOM'00, Tel Aviv, Israel*, 2000, p. 22.

[6] S. Lindsey and C.S. Raghavendra, "Energy Efficient Broadcasting for Situation Awareness in Ad Hoc Networks", *Proceedings of ICPP'01*, *Valencia, Spain*, 2001, p. 149.

[7] Ahmed M Khedr and Raj Bhatnagar, "A Decomposable Algorithm for Minimum Spanning Tree", *Distributed Computing–Lecture Notes in Computer Science*. Heidelberg: Springer-Verlag, **2918**(2003), p. 33.

[8] S. N. Slobodan and S. Shankar, "Distributed Gradient Estimation Using Random Sensor Networks", *in Proceedings of the First ACM International Workshop on Wireless Sensor Networks and Applications*, *Atlanta, GA*, 2002.

[9] E. Amir, S. McCanne, and R. H. Katz,"An Active Service Framework and its Application to Real-Time Multimedia Transcoding", *in Proceedings of the ACM SIGCOMM conference*, *Vancouver, Canada*, 1998, p. 178.

[10] D. L. Tennenhouse, J. M. Smith, W. D. Sincoskie, D. J. Wetherall and G. J. Minden, "A Survey of Active Network Research", *IEEE Communications Magazine*, 1997, p. 80.

[11] R. Nagpal, H. Shrobe, and J. Bachrach, "Organizing a Global Coordinate System from Local Information on an Amorphous Computer", *Technical Report AI Memo No. 1666*, *MIT Artificial Intelligence Laboratory*, 1999.

[12] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, "Building Efficient Wireless Sensor Networks with Low Level Naming", *in Proc. ACM Symposium on Operating Systems Principles*, *Chateau Lake Louise, Canada*, 2001, p. 146.

[13] W. R. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-Efficient Communication Protocols for Wireless Microsensor Networks", *in Proceedings of the Hawaii International Conference on Systems Sciences*, *Wailea Maui, HI*, U.S.A; 2000, p. 4.

[14] J. S. Park, M. S. Chen and P. S. Yu "An Effective Hash-Based Algorithm for Mining Association Rules", *in Proceedings of the ACM/SIGMOD International Conference on Management of data*, *San Jose, CA*, U.S.A., 1995, p. 175.

[15] Rajeev Rastogi and Kyuseok Shim., "Mining Optimized Association Rules with Categorical and Numeric Attributes", *IEEE Transactions on Knowledge and Data Engineering*, **14**(2002), p.29.

[16] Rakesh Agrawal and John C. Shafer, "Parallel Mining of Association Rules", *in IEEE Trans. on Knowledge and Data Engineering*, 1996.

[17] Bongki Moon, Ines Fernando Vega Lopez, and Vijaykumar Immanuel.,"Scalable Algorithms for Large-Scale Temporal Aggregation", *Technical Report TR 98-11*. Tucson, AZ 85721, 1998.

[18] Chugh Shrvti and Dhrama P. Agrawal, "An Energy Efficient Collaborative Framework for Event Notification and Data Aggregation in Wireless Sensor Networks", *MSc thesis, University of Cincinnati, Electrical, Engineering*, *and Computer Science*, 2004.

http://rave.ohiolink.edu/etdc/view?acc_num=ucin1077661982

[19] G. Pei, M. Gerla, X. Hong, and C. Chiang,"A Wireless Hierarchical Routing Protocol with Group Mobility", *in Proceeding of Wireless Communication and Networking Conference*, **3**(1999), p. 1538.

[20] Aram Galstyan, Bhaskar Krishnamachari, Kristina Lerman, and Sundeep Pattern, "Distributed Online Localization in Sensor Networks Using a Moving Target", *IPSN'04* , 2004.

[21] Kirill Mechitov and Sameer Sundresh, "Cooperative Tracking with Binary-Detection Sensor Networks," *ACM Sensys' 03*, 2003.

[22] Jaewon Shin, Lenidas Guibas and Feng Zhao, "A Distributed Algorithm for Managing Multi-target Identities in Wireless Ad hoc Sensor Networks", *2nd workshop on Information Processing in Sensor Network (IPSB'03)*, *Palo Alto, California*, *U.S.A.*, 2003.

[23] J.J. Liu, J. Liu, J. Reich, P. Cheung, and F. Zhao., "Distributed Group Management for Track Initiation and Maintenance in Target Localization Applications", *in Proc. 2nd Workshop on Information Processing in Sensor Networks (IPSN)*, Palo Alto, CA, April 2003, Lecture Notes in Computer Science (LNCS 2634), Springer-Verlag Heidelberg, pp. 113–128.

[24] Qing Fang, Feng Zhao, and Leonidas Guibas, "Counting Targets: Building and Managing Aggregates in Wireless Sensor Networks", *Palo Alto Research Center (PARC) Technical Report*, 2002.

[25] Javed Aslem, Zack Butler, Florin Constantin, Valentino Crespi, George Cybenko, and Daniela Rus, " Tracking a Moving Object with a Binary Sensor Network", *ACM Sensys 03* , 2003.

[26] Chin-Lung Yang, Saurabh Bagchi, and William J. Chappell, "Location Tracking with Directional Antennas in Wireless Sensor Networks", *IEEE MTT-S International Microwave Symposium (IMS 2005)*, *Long Beach, California*, *U.S.A.*, June 11–17, 2005.

[27] Yu. Byunggu, Ho Kim Seon, Bailery Thomas, and Gamboa Ruben, "Curve-Based Representation of Moving Object Trajectories", *Proc. Internal Database Engineering and Applications Symposium (IDEAS'04)*, 2004.