This full text paper was peer reviewed at the direction of IEEE Communications Society subject matter experts for publication in the IEEE INFOCOM 2009 proceedings.

Tracking with Unreliable Node Sequences

Ziguo Zhong, Ting Zhu, Dan Wang and Tian He
Computer Science and Engineering, University of Minnesota
200 Union Street SE, Minneapolis, MN, USA
Email: {zhong, tzhu, tianhe}@cs.umn.edu

***Abstract*—Tracking mobile targets using sensor networks is a challenging task because of the impacts of in-the-filed factors such as environment noise, sensing irregularity and etc. This paper proposes a robust tracking framework using node sequences, an ordered list extracted from unreliable sensor readings. Instead of estimating each position point separately in a movement trace, we convert the original tracking problem to the problem of finding the shortest path in a graph, which is equivalent to optimal matching of a series of node sequences. In addition to the basic design, multi-dimensional smoothing is developed to enhance tracking accuracy. Practical system deployment related issues are discussed in the paper, and the design is evaluated with both simulation and a system implementation using Pioneer *III* Robot and MICAz sensor nodes. In fact, tracking with node sequences provides a useful layer of abstraction, making the design framework generic and compatible with different physical sensing modalities.**

I. Introduction

Wireless sensor networks (WSN) have been considered as a promising system for area surveillance applications. Low-cost sensor nodes are deployed randomly or deliberately in an area to accomplish the task of monitoring, including event/target detection, localization and tracking with or without cooperation from the target itself. Due to extremely limited resource constrains for each sensor node, accurate large scale mobile target tracking still remains to be one of the challenging issues in the WSN community.

Many excellent ideas have been proposed for target tracking with WSN [1][2][4][5][6][7][8][11][16]. In most systems, tracking is achieved through sequential localization [3][16][2] or moving velocity measurement [5][7], cooperating with target movement modelling, estimation and filtering [1][6][11] (e.g., Kalman filter [10], Particle filters [9], Bayesian networks [33]). However, model based methods not only bring about a complex system design, but also require some maneuver-related assumptions about the mobile target. For sensor networks deployed in a wide area to detect unexpected events, it could be difficult and unrealistic to predetermine the movement parameters for possibly occurring mobile targets. On the other hand, tracking and localizing targets using ranging-based methods, in which point-to-point distance or angle measurements are required, could make the system costly for adding per-node additional hardware [3][16][12][13], or requiring careful environment profiling [3][14], both of which are highly unfavorable.

This paper investigates a new approach for mobile target tracking with sensor networks. Without assumptions of target movement models and without accurate range-based localization, mobile tracking is accomplished by processing *node sequences*, which can be easily obtained by ordering related sensor nodes according to their sensing results of the mobile target. As a range-free approach, tracking by processing node sequences provides a useful layer of abstraction: as long as the node sequences obtained reflect the relative distance relationships among the target and the sensor nodes with known positions, specific format of the physical sensing modality (e.g., heat/RF radiation, acoustic/sematic wave) is irrelevant to the tracking algorithm. Thus, the design is very generic, flexible, and compatible with different sensing modalities.

The major challenge in this system is that node sequences are *unreliable* due to combined factors such as irregular signal patterns emitted from the target, environment noise, sensing irregularity [15] and so on. By applying the space and time domain constrains that are universally appropriate for any moving object, we demonstrate that our design owns better tracking accuracy than the base-line method using sequential-based localization, especially in the scenarios where considerable noise exists.

The rest of the paper is organized as follows. Section II gives an overview about the design. Section III and IV detail the system design. Section V discusses several issues concerning practical system deployment. Section VI and VII evaluate the design with extensive simulation and a real system implementation. Section VIII briefly discusses related work, and section IX concludes the paper.

II. System Overview

This section gives an overview of the tracking system, which is composed of three parts, as shown in Fig.1.

In Fig.1(a), after the deployment of sensor nodes, the map of the area under surveillance can be divided into lots of small regions, named *faces*, according to the positions of the sensor nodes, obtained during the network deployment and initialization period [19][20]. Using the center of gravity of each face as vertices, a *neighborhood graph* can be built for the purpose of preventing biased movement estimation caused by sensing noise. In practice, both map dividing and neighborhood graph building can be pre-computed as soon as the network has been deployed.

When a mobile target enters the monitored area, sensor nodes detect certain forms of physical signals emitted from the target. Due to different geographic distances between each sensor node and the target, the sensing results at each sensor node vary, e.g., different signal strength, time-of-arrival. This naturally gives an ordering of the sensor nodes called a *detection node sequence*,

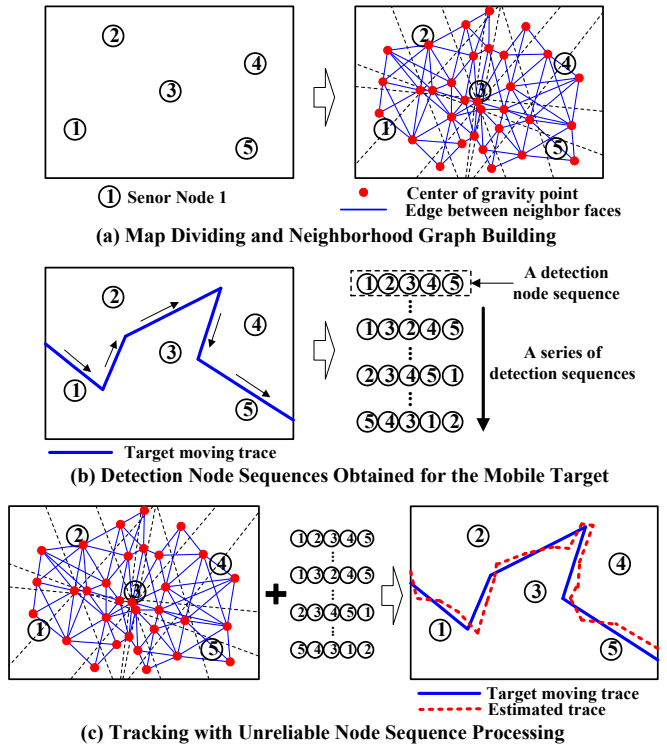


Fig. 1. Tracking System Overview

or for short, a *detection sequence*. Along the moving trace of the target, periodic sensing results from related sensor nodes produce *a series of detection sequences*, as shown in Fig.1(b), which embed relative position relationships among the sensor nodes and mobile target. Then, with pre-computed map division and neighborhood graph, the trace of the mobile target can be estimated by processing a series of detection sequences, as shown in Fig.1(c).

In the following, we concentrate on the abstract layer of node sequence processing rather than a certain type of sensing. For the sake of clarity, we first present the design without considering some practical issues such as system scalability, all of which are addressed later in section V.

III. Basic System Design

After the deployment of the sensor networks, pre-tracking preparation builds the *neighborhood graph* at the sink node with the location information of all sensor nodes, for limiting a continuous estimated trajectory of the target so as to filter out errors brought about by unreliable sensing results. Neighborhood graph building is based on the map division introduced below firstly.

A. Division of the Map

The division of the map is based on the fact that, ideally, the geographic distance between a sensor node and the target has a *monotonic* impact on the sensing readings, such as signal strength, signal time-of-fly, and etc. For example, radio, acoustic, and heat radiation signals attenuate monotonically with increasing distance in free space [21][22], and so does the time of fly for signal propagation.

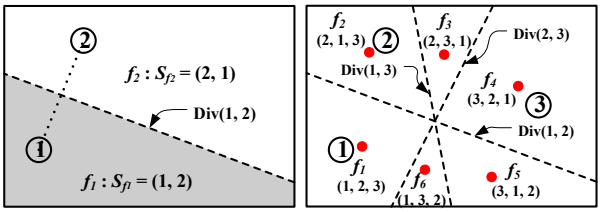


Fig. 2. Map Divding Examples

As shown in Fig.2(a), given node 1 and 2 with known positions, the whole area can be divided into two parts by the perpendicular bisector $Div(1,2)$ for the dotted segment connecting two nodes. By geometry, every position point in the gray area under $Div(1,2)$ is closer to node 1 than to node 2. So if node 1 and node 2 are ordered by increasing distance at each position point, all the position points in the gray area have a common node sequence: $(1,2)$. We call such an area composed of position points with identical node sequences as a *face* f, and the corresponding node sequence as the *signature node sequence* of face f, or for short, *signature sequence* S_f. For example, in Fig.2(a), there are two faces: f_1 and f_2 with signature sequences $S_{f_1} = (1,2)$ and $S_{f_2} = (2,1)$, respectively.

With n sensor nodes, there are $C_n^2 = \frac{n(n-1)}{2}$ perpendicular bisector lines, which divide the whole map into $O(n^4)$ faces, according to geometry study [23]. So, with an increasing number of sensor nodes, the whole map will be divided into more faces with smaller sizes. Fig.2(b) shows an example with three sensor nodes. The whole map is divided into six faces with distinct signature sequences. One fact about map division is that *each face has a unique signature sequence*.

Proof: Going from any f_i to f_j $(i \neq j)$ along a straight line, we need to across the boundary of f_i to reach f_j. The boundary is a perpendicular bisector for a pair of nodes in the map, say node u and node v. Since we follow a straight line, we can only cross this perpendicular bisector once. Therefore, when we arrive at f_j, the order of node u and v must get flipped in S_{f_j} from S_{f_i}, namely $S_{f_j} \neq S_{f_i}$.

In ideal case, the geographic distance between a sensor node and the target has a monotonic impact on the sensing readings. So, the signature sequence S_f of face f reflects the perspective ranking of in-the-field sensor nodes according to their sensing results, if the target locates in f. Therefore, with map division results, a simple localization system works as follows. Given a detection sequence S_d, target can be localized by matching S_d with each S_{f_i}. The face making the best match shows the estimated position area of the target.

B. Unreliable Detection Node Sequence

In ideal case, a detection sequences S_d should be identical with one of the face signatures. However, in a real system, sensing at each sensor node could be irregular and affected by many factors including environment noise, obstacles and etc. S_d is *unreliable*, which could be either a *full detection sequence* including all the related sensor nodes, or a *partial detection*

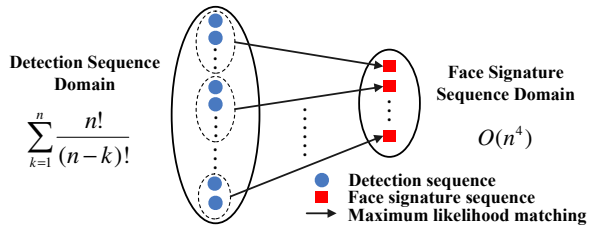


Fig. 3. Detection Sequences v.s. Face Sequences

sequence, in which some of the nodes supposed to appear are missing. In addition, nodes in S_d could get flipped due to noisy sensing.

A node sequence with k sensor node elements has $P(k,k) = k!$ possible permutations. In addition, S_d could be a partial sequence, so the total number of possible unreliable detection sequences in a system with n sensor nodes is:

$$\sum_{k=1}^{n} P(n,k) = \sum_{k=1}^{n} \frac{n!}{(n-k)!} \quad (1)$$

On the other hand, for n sensor nodes, there are $O(n^4)$ faces with distinct signature sequences. This is a much smaller space than that of the detection sequences in Equation 1. Therefore, as shown in Fig.3, given a detection sequence S_d, during most of the time there is no direct face matching; instead, we need to search for a face with the maximum likelihood.

Before going further, there are two questions need to answer firstly: (i) how to evaluate the likelihood between a detection sequence S_d and a signature sequence S_{f_i}? (ii) is there a monotonic relationship between the likelihood and the geographical distance? Next subsection addresses both questions by defining a metric, explaining the insight, and proposing computing algorithms.

C. Sequence Distance

Given two node sequences S_1 and S_2, the *Sequence Distance* $SD(S_1, S_2)$ between them is defined as *the number of flipped node pairs between* S_1 *and* S_2.

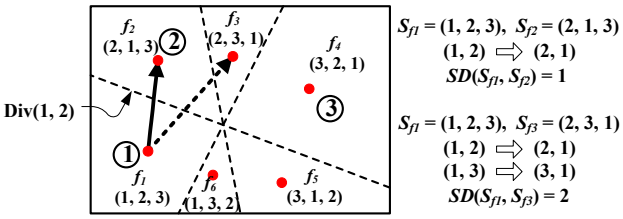


Fig. 4. Sequence Distance Example

Fig.4 shows an example, for f_1 and f_2, $S_{f_1} = (1,2,3)$ and $S_{f_2} = (2,1,3)$. Only one pair of nodes gets flipped from S_{f_1} to S_{f_2}, namely $(1,2) \Longrightarrow (2,1)$. So the sequence distance $SD(S_{f_1}, S_{f_2}) = 1$. Similarly, two pairs of nodes get flipped from S_{f_1} to S_{f_3}, so $SD(S_{f_1}, S_{f_3}) = 2$.

1) The Insight of Sequence Distance: The insight of node pair flips is crossing the bisector lines in the map. For example, in Fig.4, if we go from f_1 to f_2 along a straight line, we need to cross the perpendicular bisector $Div(1,2)$, which causes the distance relationships for node 1 and node 2 get reversed. Similarly, going from f_1 to f_3 needs to cross two bisector lines, shown with the dashed arrow, so two node pairs get flipped.

For two faces f_i and f_j $(i \neq j)$, now there are two types of distance: (i) the *geographical distance* $GD(f_i, f_j)$ between the center points of f_i and f_j, and (ii) the *sequence distance* $SD(S_{f_i}, S_{f_j})$. For these two distances, we have the following *observation I* :

$$SD(S_{f_i}, S_{f_j}) \propto GD(f_i, f_j) \quad (2)$$

Equation 2 indicates that the sequence distance between two faces is approximately proportional with their geographical distance. This is because longer geographical distance creates chances for crossing more bisectors, resulting in more flipped node pairs. Fig.5 depicts a simple example for this observation: faces with identical sequence distances to f_1 are filled with identical pattern.

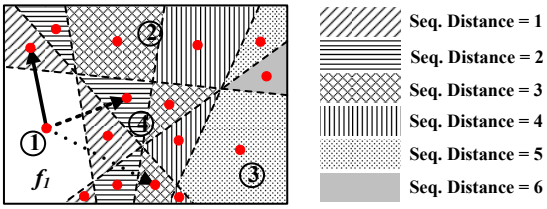


Fig. 5. Sequence Distance vs. Geographic Distance

2) Extended KT Distance Algorithm: The total number of discordant node pairs on ordering between two sequences is called the *Kendall Tau Distance* [24], or *KT distance*. Traditional KT distance addresses only the situation that two sequences have identical length and digit sets. As analyzed before, the detection sequence in our system could be partial, so traditional KT distance can not be applied directly.

In our system, nodes from any detection sequence is always a subset of the node set of signature sequences, since a signature sequence includes all nodes that can possibly detect the targets. So, given a detection sequence, it can only be (i) a full sequence with the same length as any signature sequence, or (ii) a shorter sequence composed of a subset of the nodes in the map. We define an *Extended KT Distance* (EKT Distance) to address both cases above. For case (i), EKT distance degrades to the traditional KT distance.

Definition: *The EKT distance between a detection sequence* S_d *and a signature sequence* S_{f_i} *equals the total number of flipped node pairs, considering the missing nodes in* S_d.

The basic idea for calculating the EKT distance is to use *wildcard characters*, shown in Fig.6. In Fig.6(a), if the detection sequence S_d is shorter than the signature sequence S_{f_i}, add $*$ at the end of the S_d to make S'_d with the same length as S_{f_i}. Then, for every node pair in S_{f_i}, search in S'_d to check if the ordering of this pair gets flipped. There are three cases: (i) if two nodes u and v, appear in both S'_d and S_{f_i}, simply compare their ordering; (ii) if one node, saying u, is missing from S'_d, call it a flip if the order is (u, v) in S_{f_i}, otherwise, it is a match; (iii) if both u and v are missing in S'_d, consider no flip.

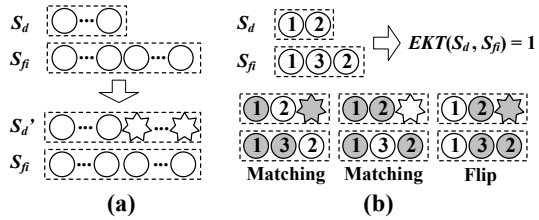


Fig. 6. EKT Distance Example

Fig.6(b) shows a simple example. There are three ordered node pairs in S_{f_i}: $(1,3)$, $(1,2)$, and $(3,2)$. Correspondingly, three ordered node pairs in S'_d: $(1,*)$, $(1,2)$, and $(2,*)$. Clearly, we can match the first two ordered pairs from two sequences, and we can find a flip between $(3,2)$ and $(2,*)$. Counting the number of flips, we can conclude that the $EKT(S_d, S_{f_i}) = 1$.

The rationale behind adding wildcards at the end of a detection sequence is that if a node is missing in S_d, it is likely to be further from the target than those nodes appearing in S_d, thus it is assumed to be at the end of the sequence.

D. Neighborhood Graph

In this subsection, the *neighborhood graph* is introduced for filtering out errors brought about by the unreliable detection sequence.

Most of the mobile targets follow the *observation II* :

$$\triangle X_{max} = V_{max} \cdot \triangle T \tag{3}$$

meaning the maximum moving offset of the target $\triangle X_{max}$, within the time interval $\triangle T$ between two sensing operations of a sensor node, is bounded by its maximum speed V_{max} times $\triangle T$. This is because that with limited maximum speed, a target moves from one position to another following a continuous trace rather than performing a "hyper-space jump".

According to Equation 3, we can conclude that if the target currently locates in one face in the map, at the next sampling instance, the target is either still in the current face or at most moves into a *neighbor face* which is defined as follows:

Definition: *If the geographic distance between the closest points of two faces are shorter than $\triangle X_{max}$, these two faces are neighbor faces to each other.*

As shown in Fig.7(a), the gray area is f_1. If a target is currently in f_1, after $\triangle T$, its location is bounded by the offset boundary depicted with a thick gray curve. Therefore, the dashed faces illustrated in Fig.7(b) are the neighbor faces of f_1. Connect the center of gravity points of neighbor faces with

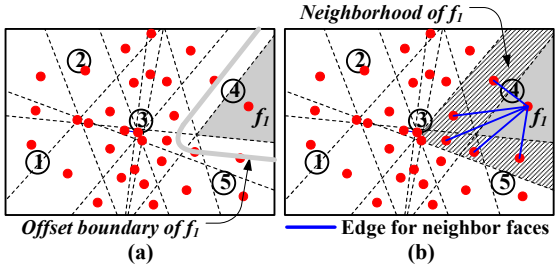


Fig. 7. Neighborhood Graph Building

that of f_1, as shown in Fig.7(b), indicating that it is possible for the mobile target to move from f_1 to those linked faces during $\triangle T$, and vice versa. Connecting all the faces with their neighbor faces builds a *neighborhood graph* G, shown in Fig.8. The vertex set $V(G)$ is composed of center of gravity points of all faces, and the link set $E(G)$ limits possible inter-face movements within $\triangle T$.

E. Tracking as Optimal Path Matching

Given a series of detection sequences $S_d(k), k = 0, 1 \cdots, M$, instead of performing per-sequence face matching, a path composed of faces $f(k)$ with minimal accumulated EKT distance to $S_d(k)$ owns maximal overall likelihood. Now, the tracking problem turns into an optimal path matching issue:

$$\begin{aligned} \text{minimize} \quad & \sum_{k=0}^{M} EKT(S_d(k), S_{f(k)}) \\ \text{subject to} \quad & f(k) \in V(G) \\ & \forall k, edge(f(k), f(k+1)) \in E(G) \end{aligned} \tag{4}$$

Section III-A mentioned that with n sensor nodes, the map can be divided into $O(n^4)$ faces. The tracking accuracy gets significantly enhanced with increasing number of sensor nodes, verified by Fig.8 showing example G with increasing n. However, G becomes extremely complicated with larger n. Finding the optimal path becomes so challenging that a naive algorithm would lead to exponential complexity growth with n.

1) Optimal Path Matching: This subsection presents a forward dynamic programming based method for solving the optimization problem shown by Equation 4.

Fig.9(a) shows a simple example for a neighborhood graph G. In the figure, each vertex stands for a face, namely face f_1, f_2, f_3, f_4 and f_5. For the sake of clarity, temporarily assume that at time $k = 0$, the starting face s of the target is f_1. Later, we will address the issue of unknown starting face s.

Starting with $s = f_1$ at $k = 0$ in G, at $k = 1$, the target can only either remain at f_1 or move into f_2 or f_3. So, f_1, f_2, and

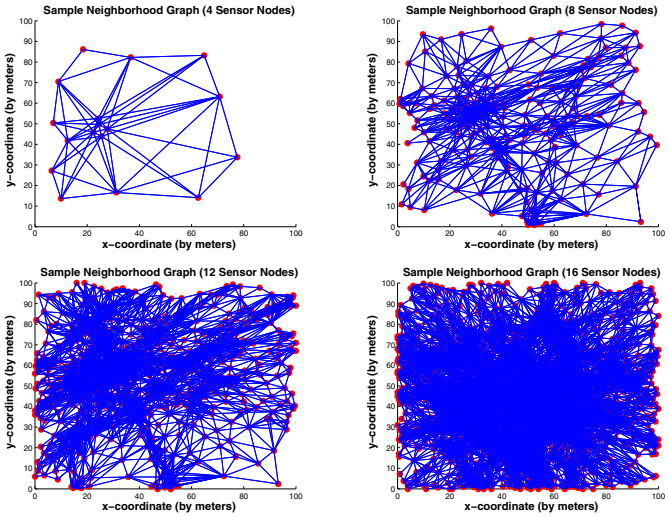


Fig. 8. G Examples with Randomly Deployed 4, 8, 12 and 16 Sensor Nodes

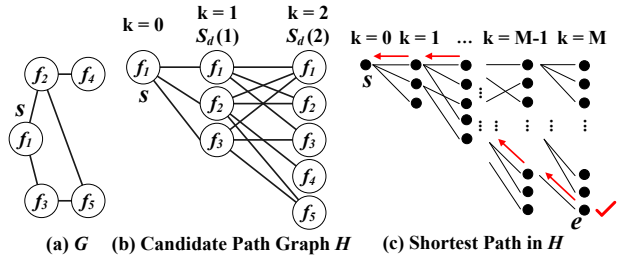


Fig. 9. From Optimal Path Matching to Shortest Path Searching

f_3 are candidate faces for time $k = 1$, listed under $k = 1$ in Fig.9(b). Connecting f_1 under $k = 0$ with its candidate faces under $k = 1$ adds edges to the graph in Fig.9(b). Repeating this process, we can list candidate faces for $k = 2$ and add corresponding edges wiring to vertices under $k = 1$. Given two detection sequences $S_d(1)$ and $S_d(2)$ for time instance $k = 1$ and $k = 2$, compute the EKT distance from the signature sequence of each face under $k = 1$ and $k = 2$ to $S_d(1)$ and $S_d(2)$, respectively. We can obtain a candidate path graph H, in which each vertex in column k is a possible face to reach at time k, and carries a weight of EKT distance to $S_d(k)$.

As shown in Fig.9(c), finding an optimal path in G with overall maximum likelihood to a series of detection sequences $(S_d(1), S_d(2) \cdots S_d(M))$ is equivalent to searching for a path in H from $k = 0$ to $k = M$ with minimum accumulated vertex weight. Defining the accumulated vertex weight of a path in H as the length of the path, now the problem turns into a *shortest path* problem in graph H, which can be solved with forward dynamic programming with polynomial complexity [25]. The basic idea is to keep only the best path to each vertex under each k in H. Then choose the vertex under $k = M$ with the minimum path length as the terminal vertex e, and recursively trace back to build the whole shortest path.

In practice, without knowing s at $k = 0$, the matching algorithm starts from $k = 1$ by choosing c faces, where $1 \leq c \ll O(n^4)$, with the smallest EKT distance to $S_d(1)$. Later simulation reveals that a small number c (e.g., $c = 3$) is sufficient to get sound results.

2) Algorithm and Complexity: The grid-like structure of graph H conveniently supports both off-line tracking, which computes an overall optimal path after collecting all detection sequences, and on-line tracking, which processes a new detection sequence immediately by adding another column in H and outputs an optimal path so far. Algorithm 1 illustrates the computation structure applicable for both systems.

Line 1 initializes the faces for $H(k = 1)$ by selecting c faces in G with the shortest EKT distance to $S_d(1)$. The H graph is built between line 2 and 10. Line 3 prepares the faces for $H(k)$, each of which is processed between line 4 and 9 by computing EKT distance, finding its single optimal $preface$, and accumulating the path cost. Finally, an overall optimal path P can be obtained by recursively tracing back from $H(M)$ to $H(1)$ at line 11.

Each $H(k)$ contains at most $O(n^4)$ faces for a system with n nodes. EKT distance calculation costs a complexity

Algorithm 1: Optimal Path Matching

input : Detection sequences $S_d(k), k = 1, \cdots, M$
Neighborhood graph G
output: Optimal path P

1 $H(1).faces$ = Initialization $(S_d(1), G)$;
2 **repeat**
3 $H(k).faces$ = Neighbor $(H(k-1).faces, G)$;
4 **repeat**
5 f = Unprocessed $(H(k).faces)$;
6 $f.dis$ = EKT$(S_f, S_d(k))$;
7 $f.preface$ = Min $(H(k-1).faces.cost)$;
8 $f.cost$= $f.preface.cost + f.dis$;
9 **until** *all faces in* $H(k)$ *are processed* ;
10 **until** $k = M$;
11 P = TraceBack $(minimum\{H(M).faces.cost\})$;

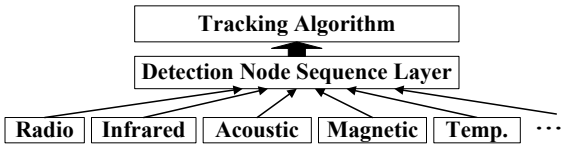


Fig. 10. Multi-Modality Integration

of $O(n \log(n))$ with a bubble-sort algorithm [26]. Preface searching at line 7 costs $O(1)$ since the optimal preface can be obtained at line 3 when listing all neighbor faces. So, the time complexity for Algorithm 1 is $O(M \cdot n^5 \log(n))$ for off-line systems and $O(n^5 \log(n))$ for updating in on-line systems. The storage complexity for both systems is $O(M \cdot n^4)$. For the moment, the complexity seems high due to $O(n^4)$ faces. In later section, we will explain how the complexity can be significantly reduced to a feasible level for large-scale networks.

IV. Multi-Dimensional Smoothing

This section introduces multi-dimensional smoothing in the modality domain, time domain, and space domain, working together to contribute to the accuracy and generality of the whole system design.

A. Modality Domain Smoothing

If a sensor node is capable of sensing the environment with multiple modalities (e.g., acoustic, infrared and etc), it could be hard to merge those sensing results at the physical layer. For example, comparing a $10dB$ acoustic signal with a $-60dBm$ RF strength is meaningless. In our design shown in Fig.10, by converting the sensing results from each modality into node sequence at the sink, an abstract layer is provided for integrating sensing results from diverse modalities. Given detection sequences from Q modalities for time k, denoted as $S_d^i(k)$ $(i = 1, 2 \cdots, Q)$, final EKT distance to face f, $D_f(k)$, can be obtained by smoothing over individual EKT distance for each modality i with different weight w_i according to its precision and reliability:

$$D_f(k) = \sum_{i=1}^{Q} \left(w_i \cdot EKT(S_d^i(k), S_f) \right) \quad (5)$$

For the tracking algorithms on top of the sequence layer, specific physical modalities become invisible.

B. Time Domain Smoothing

Time domain smoothing over continuous detection results is commonly used for filtering out random noise in many systems. Unlike most of the other systems conducting time domain averaging at the physical modality layer, in our design, smoothing can be performed conveniently at the node sequence layer. The basic idea is to average the EKT distance to each face f along the timeline over a *smoothing window* with odd length L:

$$\overline{D}_f(k) = \frac{\sum_{i=-(L-1)/2}^{(L-1)/2} D_f(k+i)}{L} \tag{6}$$

The length of the averaging window can vary in specific applications. The EKT distance to a face roughly reflects the geographic distance, so averaging EKT distances has an effect similar to averaging sensing results directly.

C. Space Domain Smoothing

The design presented so far maps the position of the mobile target at each time instance to the center of gravity point of a face in the map. This results in two phenomena: (i) many positions in the true trace are projected to the same estimated position, and (ii) estimated positions scatter at both sides of the true trace. It is not a good idea to plot the estimated trace by simply connecting those estimated positions without space domain smoothing, because it could give a curve oscillating around the true trace. A better trace estimation can be obtained by smoothing over the estimated positions using a smoothing window with odd length L':

$$\widetilde{x}_k = \frac{\sum_{i=-(L'-1)/2}^{(L'-1)/2} \widehat{x}_{k+i}}{L'} \quad \widetilde{y}_k = \frac{\sum_{i=-(L'-1)/2}^{(L'-1)/2} \widehat{y}_{k+i}}{L'} \tag{7}$$

where $(\widehat{x}_k, \widehat{y}_k)$ are estimated position coordinates for time k before space domain smoothing, which actually are the coordinates of the center of a face; $(\widetilde{x}_k, \widetilde{y}_k)$ is the final estimated position after space domain smoothing.

V. Discussion

This section discusses issues for real system implementation, including system scalability, multiple targets tracking, time synchronization, and energy efficiency.

A. System Scalability and Multiple Targets

An observation for large-scale systems is that *only a small portion of sensor nodes close to the mobile target are effective for target detection at any time instance*.

As it is shown in Fig.11, the gray area depicts the signal pattern emitted from the target. So, the length of a detection sequence is much shorter than n (total number of sensor nodes deployed). Setting a range R for an effective area large enough to cover perspective signal pattern with high confidence, shown in Fig.11, map division can be done locally with sensing range R instead of including all the sensors in the map. Now face

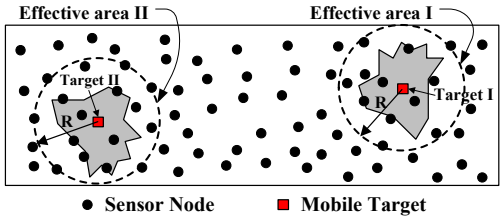


Fig. 11. Scalability and Multiple Targets

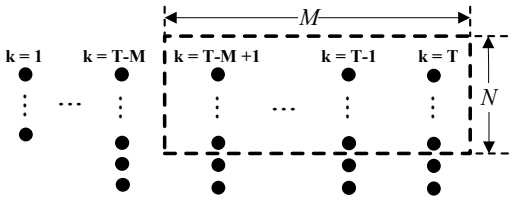


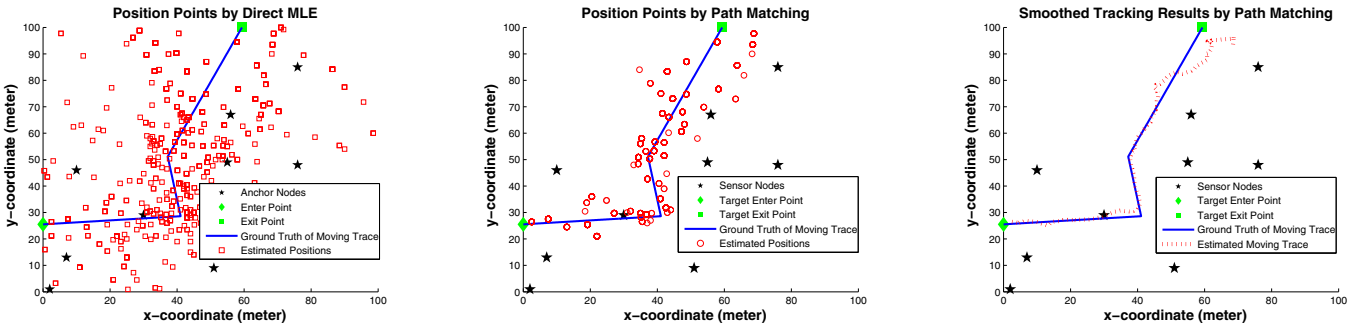
Fig. 12. Reduced Candidate Path Graph H

signature sequences have diverse length but not longer than m which is the maximum number of sensor nodes in an effective area with radius R. The EKT distance can be computed with complexity of $O(m\log(m))$ rather than $O(n\log(n))$, where $m \ll n$.

When the target stays in the monitored area for a long time, the shortest-path-searching graph H illustrated in Fig.9 could be too big to store. Fig.12 illustrates the basic idea for effectively truncating H. A processing window with length M and height N is applied to the original graph H. Horizontally, when a column moves out of the dashed window (e.g., $k = T - M$), the decision for the shortest path until this column is made and stored. Then the system works as if there is a new starting vertex in $k = T - M$ for columns $K = T - M + 1, \cdots$. Vertically, if the faces under $k = T - 1$ have more than N neighbor faces, only let faces under $k = T - 1$ with smaller accumulated path costs post their neighbors (totally no more than N) as candidate faces under $k = T$.

Now, the computation complexity is not directly related to the total number of sensor nodes n, but turns to $O(T \cdot (N\log(N) + N \cdot m\log(m)))$, where $N\log(N)$ is for sorting, and storage complexity becomes $O(M \cdot N)$ for all the vertices within the dashed window.

Multi-target tracking (MTT) [27] is not an easy extension of single target tracking because of inherent data association ambiguity. To disambiguate, MTT should be able to uniquely identify the signature of each target, which is beyond the capability of this paper. In our design, if targets are far apart from each other, the tracking system is able to differentiate them and achieve simultaneous tracking. Fig.11 depicts a simple example. If a distributed tracking system is available, the processing terminal close to effective area I could handle target I, and another processing terminal close to effective area II could handle target II. If only a single sink is used for detection sequence processing, the sink is able to differentiate detections for target I from those for target II, since the set of sensor nodes in area I is geographically distant from that in area II.



(a) Position Points by D-MLE (b) Position Points by PM (c) Smoothed Result by PM

Fig. 13. Visualized Simulation Example

B. Time Synchronization and Energy Efficiency

Current time synchronization techniques can achieve microsecond level accuracy (e.g., FTSP [28]) The sampling rate of each sensor could be from several HZ to hundreds of HZ. The time interval between two samples varies from microseconds to seconds. So if a short timestamp is attached to each sensing result, the sink or the distributed tracking terminals can correctly assemble the detection sequences for different time instance.

Energy efficiency is vital in sensor networks. Most of the time, sensor nodes keep a low duty cycle until some event or target appears in the monitored area. Nodes near the target increase their sampling rate and alert only nodes close to the projected moving trace [29], keeping others remain in sleep. On the other hand, working nodes can dynamically adjust sampling rate according to realtime tracking results. For instance, if the sensing results vary slowly, meaning that the target slows down, sensor nodes can reduce the sampling rate, vice versa. In addition, nodes adjusting their sampling rates may also notify related nodes helping them prepare for the coming target.

VI. Simulation Evaluation

We evaluated the system design with both simulation and testbed implementation. In this section, we compare the tracking performance of the optimal path matching (PM) proposed in this paper with sequential maximum likelihood estimation, or Direct MLE [30].

In the simulation, we model the monitored area as a grid map. A movement trace of the mobile target is generated with the random waypoint mobility model (RWP) [31]. An estimation error at one point in the trace is defined as the geographic offset between the estimated position and corresponding true position. The mean tracking error is defined as averaged error of all the points in the trace. All the statistics are averaged over 50 runs for high confidence. The following table illustrates the default simulation setup:

Parameter	Description
Field Area	100 (meters)×100 (meters)
Noise Model	Logarithmic ($\beta = 4, \sigma_X = 6$)
Number of Sensor Nodes	10, randomly deployed with uniform distribution
Sensing Sampling Rate	10Hz
Target Velocity	Random between 1 $\sim$ 5 (meters/s)
Averaging Window	2.9s (Time Domain), 99 points (Space Domain)

A. Noise Models

A linear delay noise model for time-of-fly based detection, depicted in equation 8, and a logarithmic attenuation noise model for signal-strength based detection [22][21], described by equation 9, are used for simulation evaluation.

$$S_i(k) \propto \frac{1}{(1+\alpha)\cdot d_i(k)}, \quad \alpha \sim N(0, \sigma_\alpha^2) \tag{8}$$

$$S_i(k) \propto -10\beta \log(\frac{d_i(k)}{d_0}) + X_i(k) \tag{9}$$
$$d_0 = 1 \quad \text{and} \quad X_i(k) \sim N(0, \sigma_X^2)$$

$S_i(k)$ stands for the sensing result of senor node i at time instance k. $d_i(k)$ is the physical distance between node i and the target at time k. In the linear model (equation 8), α is a random variable for time delay following a normal distribution with 0 mean and variance σ_α^2. In logarithmic model, β is the signal fading factor and $X_i(k)$ is a random noise at time k for node i following a normal distribution with 0 mean and variance σ_X^2.

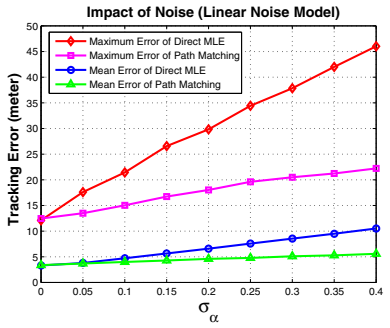
B. An Example by Figures

This subsection gives an intuitive comparison between Direct MLE (D-MLE) and path matching (PM). Detailed analysis is provided in later sections.

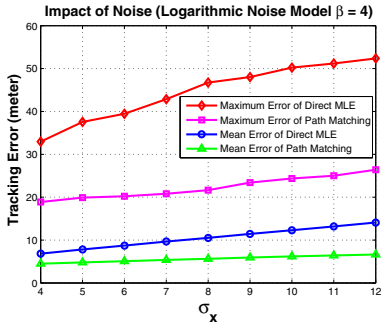
Fig.13(a) and Fig.13(b) show all the position points estimated by D-MLE and PM, respectively. From these two figures, it is clear that position points given by PM are much more closely distributed around the true trace. Fig.13(c) illustrates the smoothed traces for PM.

C. Simulation Results

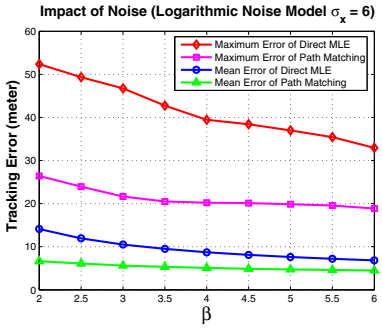
1) Impact of Sensing Noise: Fig.14(a) illustrates the performance of both methods under different σ_α for the linear noise model (Equation 8). Fig.14(a) indicates: (i) noise introduces tracking error; (ii) path matching-based tracking is very robust to noise (error increases slightly with larger noise), while the Direct MLE degrades quickly. For example, when $\sigma_\alpha = 0.4$, the D-MLE has doubled the error rates of PM. (iii) when noise is 0 ($\sigma_\alpha = 0$), the results of two methods converge. This is because if noise is 0, identical faces are chosen by both methods. Similarly, Fig.14(b) shows the performance trend of both methods under different σ_x for the logarithmic noise model (Equation 9). The figure indicates that (i) greater sensing noise



(a) Linear Noise Model



(b) Logarithmic Noise Model: σ_x



(c) Logarithmic Noise Model: β

Fig. 14. Impact of Sensing Noise to Tracking Error

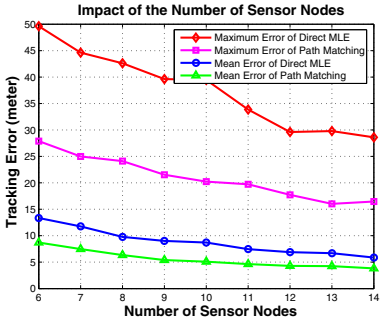


Fig. 15. Number of Sensor Nodes

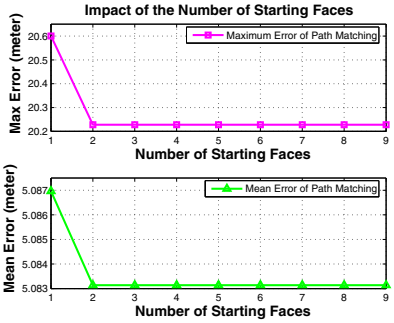


Fig. 16. Number of Starting Faces

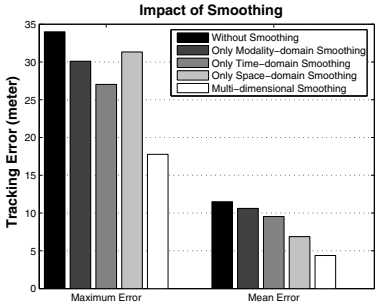


Fig. 17. Effectiveness of Smoothing

brings in larger tracking error, and (ii) path matching-based tracking is more robust to noise compared with the Direct MLE. Fig.14(c) shows the performance trend as β increases for the logarithmic noise model. We can see that bigger β reduces the tracking error. This is because the distance owns more weight with increasing β, and noise has less impact comparatively.

From the above three figures, we can conclude that the path matching based method is superior to the Direct MLE based solution, especially when the noise is considerable.

2) Impact of the Number of Sensor Nodes: We compare the path matching method with Direct MLE under a different number of sensor nodes, ranging from 6 to 14 in steps of 1. Fig.15 depicts that (i) with the increasing number of deployed sensor nodes, the tracking error is reduced, and (ii) the path matching system outperforms the system using Direct MLE.

3) Impact of the Number of Starting Faces: In order to achieve an optimal path, theoretically searching should start from every face in the graph. Simulation results shown in Fig.16 indicate that (i) increasing the number of starting faces from 1 to 2 enhances the system accuracy considerately, while (ii) more than 3 starting faces gets little performance gain. So, we can safely use 5 starting faces as the default system setup.

4) Effectiveness of Smoothing: Fig.17 illustrates the effectiveness of smoothing at each dimension individually and intergraded together. For modality domain smoothing, we assume that a node has two sensors for acoustic and RF signal strength detection, respectively. The figure shows that both smoothing at individual dimension and working together help enhance the system accuracy.

VII. System Evaluation

A system implementation of the design is conducted in the outdoor environment. A Pioneer *III* robot is used as a mobile target. 10 MICAz motes are used and 9 of them are deployed as a cross "+" shape in a lawn as shown in Fig.18. The robot, carrying a MICAz sensor nodes continuously sending out radio packages in every 100 ms, is programmed to move along a "⊓" shape trace in the field at a velocity of 10 cm/s. The robot moves with a relatively slow speed since the grass land is not even and the robot itself is heavy. The sensor nodes deployed in the lawn record the RF signal strength as well as the timestamps of the packages received. The data is processed off-line and Fig.19 illustrates the results.

The true movement trace of the robot is a distorted "⊓" since the grass ground is not flat. We can see from Fig.19 that (i) the system gives a good tracking result; (ii) larger estimation error appears at two corners of "⊓", when the robot was turning and the antenna radiation pattern was changing; (iii) there is a small burr in the estimated trace near $x = 80, y = 80$. We checked the data and found that there were 3 packages only received by the node located at $x = 60, y = 30$ when the robot was close to $x = 80, y = 80$. However, this strong noise is almost filtered out in the estimated trace. From the outdoor system evaluation, we can see that the design proposed in this paper is robust to the strong noise and works in a real system.

VIII. Related Work

Target tracking in sensor networks has been an active research topic recently [17][18][1][27]. Due to space constraints, we can only mention a few directly related works here.

Fig. 18. Outdoor System Evaluation

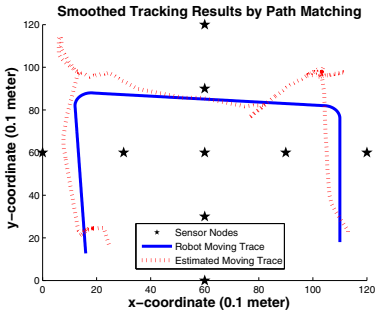


Fig. 19. Robot Tracking Results

Two recent works MSP [32] and Sequence-based [30] localize the node/target by processing node sequences. Both methods address only stationary sensor node localization and the system reliability is largely ignored. The design in this paper differs from them significantly by (i) applying movement constrains for tracking with unreliable sensing, (ii) modeling tracking as a problem of optimal path matching in a graph, and (iii) providing a framework to support multiple sensing modalities.

Another key element in tracking is movement modeling based filtering and estimation. The most widely used techniques are Bayesian networks [33], Particle filters [9], Kalman filter [10] and its extended versions [34]. Most of the existing tracking algorithms in sensor networks assume a certain movement and/or noise model, which might not be available without in-situ noise profiling and sensor calibration. In contrast, our work is shown to be robust to different types of noise models, and we impose maximum speed as the only constraint.

IX. Conclusion

This paper presented the first work for mobile target tracking using unreliable node sequences in wireless sensor networks. Tracking is modeled as an optimal path matching problem in a graph. Besides the basic design, multi-dimensional smoothing is proposed for further enhancing system accuracy. Evaluation results demonstrate that tracking with optimal path matching outperforms per-position maximum likelihood estimation. In addition, the design provides a general platform for different physical modalities with an abstract layer of node sequence.

X. Acknowledgment

This research was supported in part by NSF grants CNS - 0626609 and CNS - 0626614.

References

[1] D. Li, K. Wong, Y.H. Hu, and A. Sayeed, *Detection, Classification and Tracking of Targets in Distributed Sensor Networks*, IEEE Signal Processing Magazine, 2002, 19(2).
[2] A. Smith, H. Balakrishnan, M. Goraczkoet, and N. Priyantha, *Tracking Moving Devices with The Cricket Location System*, MobiSys '04.
[3] P. Bahl and V. N. Padmanabhan, *Radar: An In-building RF-based User Location and Tracking System*, InfoCom '00.
[4] F. Gustaffsson and F. Gunnarsson, *Mobile Positioning Using Wireless Networks*, IEEE Signal Processing Magazine, 2005, 22(4).
[5] B. Kusy, A. Ledeczi, and X. Koutsoukos, *Tracking Mobile Nodes Using RF Doppler Shifts*, Sensys'07.
[6] X.R. Li and V.P. Jilkov, *A Survey of Maneuvering Target Tracking: Approximation Techniques for Nonlinear Filtering*, SPIE '04.
[7] S. Mohanty, *VEPSD: A Novel Velocity Estimation Algorithm for Next-Generation Wireless Systems*, IEEE Trans. on Wireless Com., 2005,4(6).
[8] A. Terzis, A. Anandarajah, K. More, and I-J. Wang, *Slip Surface Localization in Wireless Sensor Networks for Landslide Prediction*, IPSN '06.
[9] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter: Particle Filters for Tracking Applications*, Artech House, 2004.
[10] R.E. Kalman, *A New Approach to Linear Filtering and Prediction Problems*, Trans. ASME, Journal of Basic Engineering, 1960.
[11] P. Zhang and M. Martonosi. *LOCALE: Collaborative Localization Estimation for Sparse Mobile Sensor Networks*, IPSN 08'.
[12] A. Savvides, C-C. Han, and M.B. Strivastava, *Dynamic Fine-grained Localization in Ad-hoc Networks of Sensors*, MobiCom '01.
[13] C.D. Whitehouse, *The Design of Calamari: an Ad-hoc Localization System for Sensor Networks*, Technical Report, UC Berkeley, 2002.
[14] K. Whitehouse, C. Karlof, and D. Culler, *A Practical Evaluation of Radio Signal Strength for Ranging-based Localization*, SIGMOBILE Mob. Comput. Commun. Rev., 2007, 11(1).
[15] J. Hwang, T. He, and Y. Kim, *Exploring In-situ Sensing Irregularity in Wireless Sensor Networks*, SenSys '07.
[16] N. B. Priyantha, A. Chakraborty and H. Balakrishnan, *The Cricket Location-support System*, MobiCom '00.
[17] R.R. Brooks, P. Ramanathan, and A.M. Sayeed, *Distributed Target Classification and Tracking in Sensor Networks*, In Proc. of IEEE, 2003, 91(8).
[18] P. Dutta, M. Grimmer, A. Arora, S. Bibyk, and D. Culler, *Design of A Wireless Sensor Network Platform for Detecting Rare, Random, and Ephemeral Events*, IPSN '05.
[19] R. Stoleru, T. He, and J.A. Stankovic, *Walking GPS: A Practical Solution for Localization in Manually Deployed Wireless Sensor Networks*, LCN '04.
[20] R. Stoleru, T. He, J.A. Stankovic, and D. Luebke, *A High-accuracy, Low-cost Localization System for Wireless Sensor Networks*, Sensys'05.
[21] H. Lord, W.S. Gatley and H.A. Evensen, *Noise Control For Engineers*, McGraw Hill Book Co., 1980.
[22] T.S. Rappaport, Wireless Communications, *Principles and Practice*, Prentice Hall, 1996.
[23] M. de Bery, O. Cheong, M. van Krevald, and et al, *Computational Geometry: Algorithms and Applications*, Springer-Verlag, 2008, 3rd Edition.
[24] M. Kendall, *Rank Correlation Methods*, Charles Griffin Co. Ltd., 1948.
[25] G.D. Forney. *The Viterbi Algorithm*. In Proc. of the IEEE, 1973, 61(3).
[26] C. Dwork, R. Kumar, M. Naor and D. Sivakumar, *Rank Aggregation Methods for the Web*, WWW '01.
[27] J. Liu, M. Chu, and J.E. Reich, *Multitarget Tracking in Distributed Sensor Networks*, IEEE Signal Processing Magazine,2007, 24(3).
[28] M. Maroti, B. Kusy, G. Simon and A. Ledeczi, *The Flooding Time Synchronization Protocol*, SenSys'04.
[29] R. Gupta, *Tracking Moving Targets in A Smart Sensor Network*, IEEE 58th Vehicular Technology Conference, 2003.
[30] K. Yedavalli and B. Krishnamachari, *Sequence-Based Localization in Wireless Sensor Networks*, IEEE Trans. on Mobile Computing, 2008, 7(1).
[31] D.B. Johnson and D.A. Maltz, *Dynamic Source Routing in Ad hoc Wireless Networks*, Mobile Computing, 1996, 353.
[32] Z. Zhong and T. He, *MSP: Multi-Sequence Positioning of Wireless Sensor Nodes*, Sensys'07.
[33] J. Liu, J. Reich and F. Zhao, *Collaborative In-Network Processing for Target Tracking*, J. on Applied Signal Processing, Mar. 2003.
[34] S. Julier and J. Uhlmann, *A New Extension of the Kalman Filter to Nonlinear Systems*, Int. Symp. Aerospace/Defense Sensing, Simul. and Controls, 1997.