Detecting the Direction of Motion in a Binary Sensor Network

Panagiotis Karras
Department of Computer Science
University of Hong Kong
Pokfulam Road, Hong Kong, China
pkarras@cs.hku.hk

Nikos Mamoulis
Department of Computer Science
University of Hong Kong
Pokfulam Road, Hong Kong, China
nikos@cs.hku.hk

Abstract

We examine the problem of detecting the direction of motion in a binary sensor network; in such a network each sensor's value is supplied reliably in a single bit of information: whether the moving object is approaching towards or moving away from the sensor. We demonstrate that the geometric properties of the network itself can be exploited for the detection of movement direction, from a single instance of sensor reading only. Moreover the estimation is performed in a distributed processing fashion, with only a minimal data collection at situation-dependent leading sensors and features a low computational burden on each sensor. In addition, different detection instances drain the resources of different groups of sensors, of a small size compared to the size of the whole network. Our experiments demonstrate high accuracy that increases with sensor density and/or sensing range, while the responsiveness of the detection model is practically instantaneous.

1. Introduction

Recent advances in sensor technology [15] have engendered the integration of physical (e.g. acoustic, seismic, thermal, optical or magnetic) sensing, data processing, memory and communication capabilities into single miniature devices. Such devices are used for the widespread deployment of wireless sensor networks with inherent computation power. Networks of this kind present revolutionary opportunities for a wide range of applications. Energy conservation is a major preoccupation for a sensor network system design, while the limited computing power and memory of sensor nodes impose a constraint on the nature of algorithms they can execute and the size of intermediate results and historical information they can store [7]. A corollary of these constraints is that data transmission to a central node is too expensive for sensor networks of non-trivial size. Therefore at least part of the computation is preferably performed locally. Since sensor networks have very different communication and computation constraints from those of state-of-the-art desktop computers and other dedicated data equipment, data processing in such networks cannot rely on existing techniques.

Several mainstream sensor network applications, like wildlife monitoring and battlefield surveillance, involve the detection and/or tracking of moving targets and the answering of queries about the motion patterns observed. In this paper we specifically investigate a fundamental problem the underlies such applications, namely the detection of the direction of a target's movement. We adopt a pragmatic minimalist model, whereby each sensor provides a single bit of information; this bit indicates whether a tracked object is arriving to or departing from the sensor. This model is realistic, as its sensing element can be implemented with basic, established, co-locatable and passively operating sensing hardware, such as acoustic, thermal or magnetic sensors [1]. Mathematically, the bit obtained by such a *binary* sensor is the sign of the derivative $\frac{\mathrm{d}|\vec{d}|}{\mathrm{d}t}$, where $|\vec{d}|$ is the Euclidean distance of the moving object from the sensing device. We call this a *departure/arrival* bit, to distinguish from a *proximity* bit, which indicates whether a target lies within a threshold distance. Notice that the former, as opposed to the latter, provides information of a *dynamic* nature; in order to derive this information it relies on minimal signal processing that can be reasonably and reliably performed in situ [1].

A previous exploration of this model for motion tracking [2] suggests a probabilistic method using particle filtering techniques [6]. However, this approach does not fully exploit the geometric characteristics of the network itself. In addition, it rests on the assumption that all data is available in a centralized repository. Such an approach inflicts uneconomical energy consumption on the network and is not scalable with its size. In our methodology we postulate no centralized data collection and we allow any node to request and collect information. Our model makes use of the geometric properties of the network and the computational powers of the sensors themselves.

2. Background and Related Work

The modern ubiquity of sensor networks and its prospective future growth has spawned considerable research attention, encompassing their applicability for mobile target tracking [4, 8, 16, 3, 14, 13, 11]. The binary sensor model was employed in [2] for detecting the direction of motion, when a single *departure/arrival* bit of information is available, as well as for localizing a target, when an additional *proximity* bit of information is available. [10] introduced a distributed tracking algorithm for graphs employing a binary sensor model as well; in this case, the single bit of information is a proximity bit and the focus of the work is the difficulties of sensor localization. In this paper we adopt the departure/arrival bit model, while we assume that the localization problem has been solved in advance. We also follow a computational geometry [5] approach for localized data processing, as it has been done wherever the geometry of the network itself is deemed relevant [9, 12].

3. Problem Formulation

Following the binary model, we assume that each sensor can only supply one bit. However, we do not require that this be transmitted to a base station. We adopt a more pragmatic approach, in which every sensor's bit is communicated only to its communication-range neighbors. Henceforward, we use the term *neighbors* in this specific sense. Each sensor s is able to detect whether a moving target is distancing itself from, or approaching towards s. In the former case s's bit acquires a *minus* $(-)$ value, while in the latter a *plus* $(+)$ value. We assume that each sensor can detect motion within a certain *sensing range* of radius R, and share its bit with its neighbors. Moreover, each sensor can communicate its location to its neighbors. We focus on the problem of estimating an object's direction of movement from the instantaneous readings of the sensors that detect it in their range. Assume that a set of $|\mathcal{S}|$ binary sensors $\mathcal{S} = \{s_1, s_2, \ldots, s_{|\mathcal{S}|}\}$, deployed within a bounded $2D$ area can detect an object W moving inside this area. Let $\vec{V}(t)$ be the velocity vector of W, $\vec{N}(t)$ the normal to $\vec{V}(t)$, and $\vec{X}(t)$ a parametric representation of the object's changing position. Then if λ_t is the slope of $\vec{V}(t)$ and $\hat{\lambda}_t$ the slope of $\vec{N}(t)$, it will be $\hat{\lambda}_t = -\frac{1}{\lambda_t}$. Each sensor $s_i \in \mathcal{S}$ produces a sequence of readings $v_i^j \in \{-, +\}$ indicating whether W is departing from or approaching s_i at time t_j. We devise a protocol that will provide an estimate for the instantaneous direction of movement expressed by λ_{t_j}.

3.1. Geometric Observations

For the purposes of our discussion, we define as *slope interval* an ordered pair $\{\phi, \psi\}, \phi, \psi \in [0, 2\pi)$. The pair denotes an interval of angles around the trigonometric circle in counterclockwise fashion. If $\phi > \psi$ then the interval denotes the union $[\phi, 2\pi) \cup [0, \psi]$, otherwise it denotes the simple interval $[\phi, \psi]$. Given an *ordered* pair of points in $2D$ space, $\{p_1, p_2\}$, we can determine the direction of the vector $\overrightarrow{p_1p_2}$ as an angle in $[0, 2\pi)$ using their coordinates. Notably, the estimate for the direction of a target's movement can be expressed a slope interval.

For the sake of simplicity, in the rest of our exposition we shall use the notation s_i to denote both a physical sensor as well as its abstract geometric position as a point. Let v_q the reading of sensor s_q at a certain time t. Letting $A = \{s_i | v_i = +\}$, $B = \{s_j | v_j = -\}$, and $C(A)$ ($C(B)$) be the convex hull of A (B), it is proved in [2] that $C(A) \cap C(B) = \emptyset$ and $X(t) \notin C(A) \cup C(B)$. In other words, W lies outside the (distinct) *minus* and *plus* sensors' convex hulls. Moreover, these distinct hulls are separated by the normal $\vec{N}(t)$ to the velocity $\vec{V}(t)$, while $\vec{V}(t)$ points to $C(A)$. We extend these basic geometric observations by the following Theorems:

Theorem 1 Given a pair of equi-signed sensors $\{s_i, s_j\}$, $v_i = v_j$, and a third opposite-bit sensor s_k, $v_k = -v_i$, then the slope $\hat{\lambda}$ of $\vec{N}$ lies in the interval between the slopes of the lines s_is_k and s_js_k that is exterior to the convex angle $\widehat{s_is_ks_j}$ and its opposite angle formed by these lines.

Proof. Since $\vec{N}$ separates the convex hull of negative-signed sensors from that of positive-signed sensors, it follows that it separates $\{s_i, s_j\}$ from s_k. Then a line L parallel to $\vec{N}$ anchored on s_k leaves s_i and s_j on the same semi-plane, hence it is exterior to the angle $\widehat{s_is_ksj}$ and its opposite formed by the lines s_is_k and s_js_k. Therefore, the slope $\hat{\lambda}$ of $\vec{N}$ lies in the interval between the slopes of the lines s_is_k and s_js_k that is exterior to the convex angle $\widehat{s_is_ks_j}$ and its opposite formed by these lines. ∎

Based on Theorem 1, we conclude that every triplet of non-equal-signed sensors defines an allowed slope interval for the slope $\hat{\lambda}$ of $\vec{N}$, hence for the slope λ of $\vec{V}$ as well. We now prove a related theorem:

Theorem 2 Assume a network instance including at least two $+$ and two $-$ sensors. Then for each $+$ $(-)$ sensor s_i there exists at least one pair of $-$ $(+)$ sensors $\{s_j^i, s_k^i\}$ such that *all* other same-bit (as the pair) sensors lie within the *convex* region bounded by the semi-lines $s_is_j^i$ and $s_is_k^i$. We call $\{s_j^i, s_k^i\}$ the *covering pair* of s_i.

Proof. For any given node s_i, let there be a random pair of sensor nodes of the opposite sign, $\{s_j, s_k\}$. If there exists at least one sensor s_l, equi-signed to s_j and s_k, lying outside the convex angle $\widehat{s_j s_i s_k}$, then the line $s_i s_l$ leaves s_j and s_k on the same semi-plane (if it did not, then s_l would lie in the convex region bounded by the complementary semi-lines of $s_i s_j$ and $s_i s_k$; hence, the triangle of equi-signed vertices $s_j s_k s_l$ would include the opposite-signed s_i, which is a contradiction). Without loss of generality, assume that s_j and s_k lie on the *left* of the (directed) line $s_i s_l$, hence s_k lies in the convex angle $\widehat{s_j s_i s_l}$. Then let node s_l substitute s_k to get a new pair, $\{s_{j'}, s_{k'}\}$. If there still exists a node $s_{l'}$ of the same sign lying outside the convex angle $\widehat{s_{j'} s_i s_{k'}}$, then let it again substitute the node with which it forms the maximum convex angle headed at s_i. Let this process be repeated until it terminates, given the finite number of nodes. The final pair is the desired covering pair of nodes $\{s^i_j, s^i_k\}$ such that *all* other equi-signed nodes lie within the *convex* angle $\widehat{s^i_j s_i s^i_k}$. There may be more than one such pairs, in case there are more than one nodes lying on the arms of the *convex* angle $\widehat{s^i_j s_i s^i_k}$. The process will definitely terminate at any of these pairs. ∎

A Corollary of Theorem 2 is that, for a given sensor s_i and the associated *covering pair* $\{s^i_j, s^i_k\}$, the convex angle $\widehat{s^i_j s_i s^i_k}$ is the *maximum* angle that can be formed with its vertex in s_i and its arms defined by a pair of opposite-signed sensors. Besides, according to Theorem 1, the triplet $\{s_i, s^i_j, s^i_k\}$ defines a slope interval for $\hat{\lambda}$. This interval is the *tightest* of all slope intervals similarly defined by s_i. Accordingly, we call it *minimal slope interval* of s_i. As a matter of fact, in all possible configurations, there exists a quadruplet or triplet of sensors, consisting of at most a pair of positive-bit sensors and at most a pair of negative-bit sensors, which is sufficient in order to provide the *tightest possible* constraint of the slope $\hat{\lambda}$, as the following theorem shows.

Theorem 3 Let there be a set of sensors $S = \{s_1, s_2, \ldots, s_k\}$, divided into two convex hulls according to their readings. Then, in the general case, there exists a quadruplet of sensors $\{s_{i_1}, s_{i_2}, s_{j_1}, s_{j_2}\}$, such that $v_{i_1} = v_{i_2} = +$ and $v_{j_1} = v_{j_2} = -$, and the two diagonals D_1, D_2 of the convex quadrilateral T, $T = s_{i_1} s_{i_2} s_{j_1} s_{j_2}$ divide the $2D$ plane in four areas, as follows: An area on the side of s_{i_1} and s_{i_2}, where all positive-valued sensors are found, an area on the side of s_{j_1} and s_{j_2}, where all negative-valued sensors are found, and two interloping void areas. Then a line L parallel to $\vec{N}$ anchored on the intersection of D_1 and D_2 will lie on the void areas and hence its slope is bounded by the slopes of D_1 and D_2. We call this group of four sensors *critical quadruplet*, and the pairs that define D_1 and D_2 *critical pairs*, for the given motion. In special cases, one of the two equal-signed sensor pairs within the quadruplet may degenerate into a single sensor, hence the quadruplet degenerates into a *critical triplet*. In all cases there exist *exactly two* lines $D1$, $D2$, that cross (at least) a pair of opposite-signed node points (vertices of the respective convex hulls) and both leave the rest of the two groups of equal-signed node points (i.e. the two convex hulls) in different semi-planes.

Proof. For any $+$ node s^1_i, let $\{s^1_k, s^1_l\}$ be its covering pair. If there exists a $+$ sensor s^2_i within the convex angle $\widehat{s^1_k s^1_i s^1_l}$, then let s^2_i substitute s^1_i. Let $\{s^2_k, s^2_l\}$ be the covering pair of s^2_i. It follows that the convex angle $\widehat{s^2_k s^2_i s^2_l}$ is larger than the antecedent convex angle $\widehat{s^1_k s^1_i s^1_l}$, hence the minimal slope interval of s^2_i is tighter than the respective of s^1_i. If there exists a sensor s^3_i within the convex angle $\widehat{s^2_k s^2_i s^2_l}$, then let the substitution process be repeated, until it terminates, given the finite number of nodes. The resulting triplet $\{s^n_i, s^n_k, s^n_l\}$ will leave all $-$ sensors within the convex angle $\widehat{s^n_k s^n_i s^n_l}$, and all $+$ sensors outside this area. For the sake of simplicity, let us call this triplet $\{s_i, s_k, s_l\}$. If all $+$ sensors lie within the area bounded by the opposite convex angle of $\widehat{s_k s_i s_l}$, then the triplet $\{s_i, s_k, s_l\}$ is a critical triplet. Otherwise, if there exists a $+$ sensor s^1_j lying outside that convex angle, i.e. in one of the other two (interloping) angles formed between the intersecting lines $s_i s_k$ and $s_i s_l$, then, without loss of generality, assume that s^1_j lies in the same semi-plane defined by line $s_i s_l$ as s_k, and let $\{s^1_q, s^1_r\}$ be its covering pair. Also without loss of generality, assume that s_i lies in the same semi-plane defined by the line $s^1_j s^1_q$ as s^1_r. Consider the quadruplet $\{s_i, s^1_j, s_k, s^1_r\}$ and let c be intersection of lines $s_i s_k$ and $s^1_j s^1_r$. If there still exists a $+$ sensor s^2_j in one of the interloping angles $\widehat{s^1_j c s_k}$, $\widehat{s_i c s^1_r}$, then, without loss of generality, assume that s^2_j lies in $\widehat{s^1_j c s_k}$. Let s^2_j substitute s^1_j in the quadruplet (if s^2_j lied in $\widehat{s_i c s^1_r}$ we would let it substitute s_i). Let s^2_r be the member of s^2_j's covering pair that leaves s_i and s_l on different semi-planes. Then let s^2_r substitute s^1_r in the quadruplet (in the opposite case, if s^2_j substituted s_i, then s_k would be substituted by the appropriate member of s^2_j's covering pair). Let this substitution process be repeated, until it finds the interloping angles empty. Note that the iterative process is guaranteed to terminate, given the finite number of nodes and the fact that the interloping angles can only decrease after each substitution. For the sake of simplicity, call the resulting quadruplet $\{s_i, s_j, s_k, s_r\}$. Then all $+(-)$ sensors lie within a single area bounded by the diagonals D_1, D_2 of the convex quadrilateral defined by the quadruplet and we have established a critical quadruplet. ∎

Figure 1 shows the critical quadruplet derived in a random network. A corollary of Theorem 3 is that the members of the critical quadruplet are lying on the border of their respective convex hulls; if they lied in a hull's interior then diagonals D_1 and D_2 would not leave all equal-signed sensors on the same semi-plane. Let s_i the location of a node and s'_i be its projection to the line L parallel to $\vec{N}$ anchored on the moving object. If (i) s is in either $+/-$ convex hulls, (ii) the line segment $s_i s'_i$ does not cross the interior of the respective signed convex hull, and (iii) s is a neighbor of an opposite-signed node, then s_i is called a *facing node*. We call property (ii) the *projection property*. The ordered polyline formed by facing nodes on the same (either $+$ or $-$) convex hull is called the *facing line* of the corresponding convex hull. An example is illustrated in Figure 2.

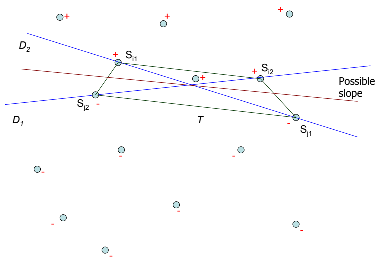


Figure 1. A *critical quadruplet* of sensors within a $2D$ sensor field for a given set of bit values

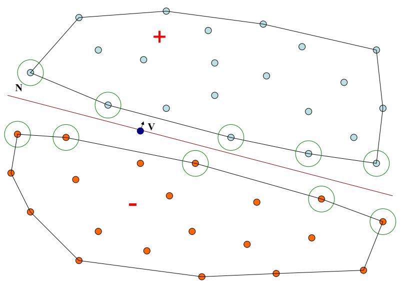


Figure 2. *Facing lines* of two convex hulls

3.2. Problem Statement

Given the above theoretical background, the following problem poses itself: Given a set of $|\mathcal{S}|$ binary sensors $\mathcal{S} = \{s_1, s_2, \ldots, s_{|\mathcal{S}|}\}$, deployed within a bounded $2D$ sensor field and an object W moving inside this field, we are interested in formulating a power-efficient and computationally effective model of intra-sensor communication and computation that can determine the *critical quadruplet*, and hence a valid slope interval for the slope λ_t of $\vec{V}(t)$, at a given time t_j.

4. A Protocol for Direction Detection

4.1. Assumptions

We assume that the sensor nodes are randomly distributed across the whole sensing area with uniform density and isotropic disk connectivity. The reading of each sensor is derived continuously and passively, i.e. the sensor does not need to transmit a signal in order to make a measurement [1]. Whenever the measured value of a sensor changes, it broadcasts an update signal to its neighbors. Each node is equipped with adequate memory to store the readings and locations of its neighbors' set. The exchange of location information is made once during the network deployment and initialization stage. Thereafter an update operation is executed by sending the measured bit along with a sensor id. Hence all nodes continuously keep track of their neighbors' readings. Given the isotropic disk connectivity, it follows that the set of $+$ $(-)$ facing nodes forms a contiguous sequence of $+$ $(-)$ convex hull vertices. If not, then there would be a convex hull vertex, whose two adjacent vertices were facing nodes, that would satisfy the *projection property* but have no (opposite-signed) neighbors in an angle of 180 degrees - a contradiction [1]. Let $\mathcal{S}_o \subseteq \mathcal{S}$ be the set of all sensors that detect an opposite-signed neighbor, and left $\mathcal{F}$ be the set of all *facing* sensor nodes. Then it follows that $\mathcal{F} \subseteq \mathcal{S}_o$, i.e. $\mathcal{S}_o$ includes all *facing* nodes as well other nodes, which can detect an opposite-signed neighbor but are not vertices of one of the two signed convex hulls. Notice that, in a network $\mathcal{S}$ of uniformly distributed sensors with balanced extent in two dimensions, if $|\mathcal{S}|$ is the total number of its nodes, then the size $|\mathcal{S}_o|$ of the linearly distributed set (and the corresponding number $|\mathcal{F}|$ of facing nodes) is expected to be $O(\sqrt{|\mathcal{S}|})$. The operation of our protocol rests exclusively on nodes in $\mathcal{S}_o$, while the rest remain asleep.

4.2. Direction Estimation

After a movement update, each node s establishes its whereabouts in the instantaneous geometry of the network as follows. First, s determines whether it has at least one opposite-signed neighbor and gives notice of this information to all its neighbors. For any such node, let the *peerset* $\mathcal{PS}(s)$ of s be the set of nodes $p \in \mathcal{S}_o$ which are neighbors of s and read the same sign as s. In effect, s becomes aware of its peerset by their transmissions. Furthermore, s establishes its closest opposite-signed neighbor, s_-. Afterwards, each node $s \in \mathcal{S}_o$ enters a state of communication alertness (during which a sequence of communications and

[1]The case when the target moves at the verge of the workspace notwithstanding.

computations eventually derives the direction of the object movement), while the remaining network stays asleep.

Linking of Peers After it has established its property as such, each node $s \in \mathcal{S}_o$ determines its *left* and *right links* among its peerset, denoted as L_s and R_s respectively, $\mathsf{L}_s, \mathsf{R}_s \in \mathcal{PS}(s)$. This pair of sensors is selected so that the angle $\widehat{\mathsf{L}_s s \mathsf{R}_s}$ divides the plane into two regions, one containing all opposite-signed nodes seen by s, and another containing all equi-signed nodes seen by s. In order to facilitate the *links*' determination, the directed counterclockwise angles that each node $p \in \mathcal{PS}(s)$ forms with s at the head and s_- at the other arm are calculated. Therewith the selected left and right links, L_s and R_s are defined as:

$$\mathsf{L}_s = \left\{ p_l \middle| \widehat{p_l s s_-} = \max_{\{p_j \in \mathcal{PS}(s)\}} \left(\widehat{p_j s s_-}\right) \right\}$$
$$\mathsf{R}_s = \left\{ p_r \middle| \widehat{p_r s s_-} = \min_{\{p_j \in \mathcal{PS}(s)\}} \left(\widehat{p_j s s_-}\right) \right\}$$

The orientations left and right are meant with respect to the direction looking towards the opposite-signed sensors' semi-plane. An illustration of the concept behind these formulae, along with a sketch of the calculated extreme angles, is depicted in Figure 3.

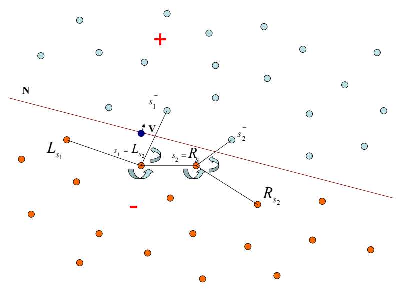
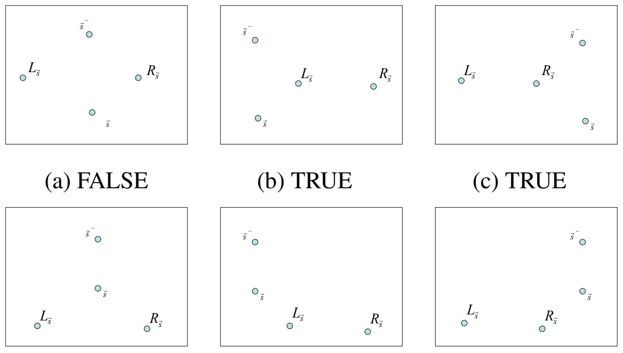


Figure 3. Links and closest opposite nodes for s_1, s_2

Determination of Edges In the same process, each node $s \in \mathcal{S}_o$ determines whether it lies on the edge of a *facing line*. This property is decided by the following rule: A node $s \in \mathcal{S}_o$ is called an *edge* if and only if (i) it has reciprocal left/right relationship with one and *only* one of its links (either L_s or R_s), and (ii) the line segments from s to s_- and from L_s to R_s *do not cross* each other. Symbolically, let $\tau\{a, b, c\}$ denote the direction of the turn formed by the ordered triplet a, b, c, which takes the values $left$, $right$ or $none$, and $\mathsf{NOT\_CROSS}(s)$ denote the second property above. Then this property can be formally be expressed as:

$$\begin{aligned} \mathsf{NOT\_CROSS}(s) = \quad & (\tau\{\mathsf{L}_s, \mathsf{R}_s, s\} = left) \\ \vee \quad & (\tau\{s, s_-, \mathsf{L}_s\} = \tau\{s, s_-, \mathsf{R}_s\}) \end{aligned}$$

The above equation expresses the clause that the line from L_s to R_s leaves both s and s_- on the same (left, by necessity) semi-plane, *or* the line from s to s_- leaves both L_s and R_s on the same semi-plane. All six possible configurations of the four involved nodes are depicted in Figure 4.



(a) FALSE (b) TRUE (c) TRUE

(d) TRUE (e) TRUE (f) TRUE

Figure 4. Six possible configurations of the four involved nodes and respective values of the $\mathsf{NOT\_CROSS}(s)$ function

Then the edge property is defined as:

$$\begin{aligned} \mathsf{EDGE}(s) = \quad & s \in \mathcal{S}_o \\ \wedge \quad & ((\mathsf{R}_{\mathsf{L}_s} = s) \otimes (\mathsf{L}_{\mathsf{R}_s} = s)) \\ \wedge \quad & (\mathsf{NOT\_CROSS}(s)) \end{aligned}$$

Notice that the symbol $\otimes$ denotes the boolean exclusive or. For example, a node $s \in \mathcal{S}_o$ at the left edge of a facing line will naturally be the left link of its own right link, but will not be the right link of the node that it has been untowardly attributed to as a left link. Moreover, this node will satisfy the $\mathsf{NOT\_CROSS}(s)$ property. Notice that there will be exactly four edges for each network instant, two for each facing line.

Let a *non-crossing* (*crossing*) node be a node that satisfies (does not satisfy) the $\mathsf{NOT\_CROSS}(s)$ property. Then a non-crossing node that has a reciprocal relationship only with its *left* link is a *right edge*, while one who has a reciprocal relationship only with its *right* link is a *left edge*, along a facing line. Notice that there potentially exist other nodes $s \in \mathsf{S}_o$ with a single non-reciprocal relationship; however, these are crossing nodes, serving as *auxiliary* nodes that facilitate communication between their neighbors. An example of such non-reciprocal, crossing, auxiliary link nodes is depicted in Figure 5. The overall procedure that performs the linking and edge-related decisions for a node s is shown in Figure 6.

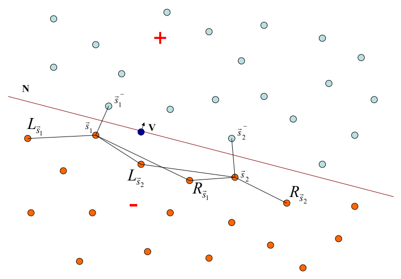


Figure 5. Crossing, auxiliary link nodes between s_1 and s_2

```
Process Linking(Node $s$)
1. $s_-$ := nearest opposite-signed node of $s$;
2. compute $\mathsf{L}_s$, $\mathsf{R}_s$;
3. receive values $\mathsf{R}_{\mathsf{L}_s}$, $\mathsf{L}_{\mathsf{R}_s}$;
4. compute EDGE($s$);
5. if EDGE($s$) then
6.    if ($\mathsf{R}_{\mathsf{L}_s} = s$) then direction($s$) := left;
7.    if ($\mathsf{L}_{\mathsf{R}_s} = s$) then direction($s$) := right;
```

Figure 6. Process for linking nodes in $\mathcal{S}_o$ and determining edges

Message Transmission The above process leads to the establishment of the four edges, positive and negative leftward and rightward edges, respectively. Each of them initializes a message transmission towards the center of its facing line. In effect, two messages are aggregated along each of the two facing lines, from the edges towards their middle parts. These transmissions collect the coordinates of all facing nodes. The nodes $s \in \mathcal{S}_o$ that participate in this message propagation are divided into (i) facing nodes and (ii) *auxiliary* nodes, which are not convex hull vertices, but they must assist in the propagation, due to the limited communication range. The critical quadruplet is composed out of such nodes, therefore the accumulated messages need to include only coordinates of such *facing* nodes. In practice, the nodes which append their coordinates to a propagated message m will be significantly fewer than the total nodes which assist in its propagation, rendering the total size of m manageable. The operation occurring on every node is described in Figure 7.

Specifically, each *edge* node initiates a message m with its own coordinates and forwards it to its appropriate *link* l; in its own turn, l propagates m towards its own link in the same direction. If l is non-crossing, before propagation, it appends its own location information to m and prunes from it any previously entered node-coordinates that are now shown not to be facing nodes. This pruning is performed using the basic step of the convex hull computation algorithm [5]. However, in our distributed in-network convexity computation approach the coordinate-sorting step of the centralized algorithm is avoided.

```
Process Receive/Transmit(Node $s$)
1. Receive compact message and direction information;
2. if NOT_CROSS($s$) then
3.    Append own coordinates on message;
4.    while last three nodes do not make turn of direction
5.       delete middle of last three nodes;
6. Transmit message towards link in direction;
```

Figure 7. Process for receiving/transmitting information along a facing line

Once a node receives messages from both directions, it appends them to each other and broadcasts a signal to its neighbors that it assumes a *leader* role. In effect, one leader emerges on each signed hull's *facing line*, l_+ for the *plus* hull and l_- for the *minus* hull. Then l_- transmits its own message to its closest opposite-signed neighbor. Therefrom it is forwarded along the established path towards l_+. In effect, the aggregate coordinate information of all *facing* nodes, $f_-^1, f_-^2, \cdots, f_-^m, f_+^1, f_+^2, \cdots, f_+^n$, is collected in the memory of a single sensor. Figure 8 illustrates the flow of information within the network. Both *auxiliary* and *facing* nodes are highlighted in the figure.

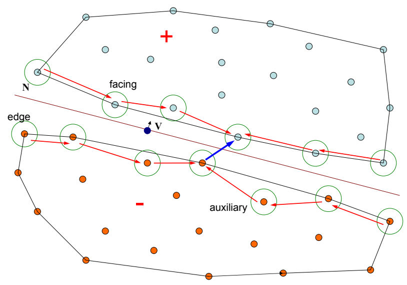


Figure 8. Message propagation along facing and auxiliary nodes for the hulls of Figure 2

The overall process of computing and forwarding the set $\mathcal{F}$ of facing nodes to node l_+ requires a number of message transmissions which $O(|\mathcal{S}_o| = O(\sqrt{|\mathcal{S}|})$, according to our uniformity assumption. Therefore the overall process is significantly cheaper compared to the centralized algorithm of [2], which requires that all sensors should send their readings *unconditionally* to a centralized server for processing.

Determination of Slope Interval After collecting both lists of *facing* nodes, sensor l_+ proceeds to the computational task of establishing the critical quadruplet. This computation operates with a mutating pair of opposite-signed facing nodes, $\{n_-, n_+\}$; the original seed for this is the pair of the two established leaders, $\{l_-, l_+\}$. The vector $\overrightarrow{n_- n_+}$ this pair forms, of slope $\mathsf{slope}(n_-, n_+)$, is then rotated by substituting its two members with an appropriate follower in the facing line. This way, the pairs of opposite-signed nodes that form the vectors of maximum clockwise (counterclockwise) difference from $\mathsf{slope}(l_-, l_+)$ are established. These two pairs from the *critical quadruplet* (or triplet, potentially) that we have been looking for. The algorithm (illustrated by the pseudo-code of Figure 9) returns a critical pair $\{n_1, n_2\}$ when executed in this (clockwise) version, and another critical pair $\{n_3, n_4\}$, if its mirror-image (counterclockwise) version (with left and right interchanged) is executed instead. These two critical pairs from the critical quadruplet (or triplet), i.e. define the lines D_1 and D_2 whose slopes delimit the moving object's velocity slope.

```
Algorithm RotateLine($l_-, l_+$)
Input: Seed of two leading nodes $l_-, l_+$;
Output: A pair of critical nodes $n_+, n_+$;
1.  $n_- := l_-$; $n_+ := l_+$
2.  do
3.      if $\exists$ adjacent node to $n_-$ on the facing line on the left of $\overrightarrow{n_- n_+}$
4.          then
5.          substitute $n_-$ by that adjacent node ($\overrightarrow{n_- n_+}$ rotates clockwise);
6.      if $\exists$ adjacent node to $n_+$ on the facing line on the right of $\overrightarrow{n_- n_+}$
7.          then
8.          substitute $n_+$ by that adjacent node ($\overrightarrow{n_- n_+}$ rotates clockwise);
9.  until pair $\{n_-, n_+\}$ does not change;
```

Figure 9. Rotation algorithm

Complexity Analysis Let $|\mathcal{F}|$ be the total number of facing nodes. In every *node substitution* step of algorithm RotateLine (Figure 9), the line $\overrightarrow{n_- n_+}$ is rotated in the same direction. Sameness of rotational direction for $\overrightarrow{n_- n_+}$ does not translate into sameness of movement direction along the two convex hulls, as there may be back-and forth movements of re-orientation. Such a case arises in Figure 10. Line $\overrightarrow{n_- n_+}$ starts out as line 1 in the picture and adjusts itself clockwise to line 2; after this line is reoriented to position 3, then a backward movement has to be made along the minus hull to position 4; thus node n is visited twice. However, such a repetition cannot occur *more* than once. In the above example, the algorithm cannot pass from node n again. The double-scanning of n implies that the second visit upon this node was made to rectify a previous erroneous substitution. The change of direction of *scanning* the $-$ convex hull means that the previous direction was not leading towards a critical node sought after; hence each node shall be visited at most twice. It follows that the complexity of RotateLine is $O(|\mathcal{F}|)$.

Notice that the computation performed in the distributed manner outlined above would amount to a convex hull algorithm if executed centrally. The state-of-the-art complexity of such an algorithm is $O(|\mathcal{S}| \log |\mathcal{S}|)$ on the number of nodes $|\mathcal{S}|$ [5]. Thus, apart from alleviating the burden of transmission to a sink, our distributed computation approach also lightens the total computation burden which is shared among individual nodes by exploiting their own geometrical awareness.

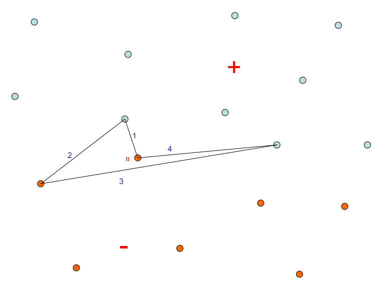


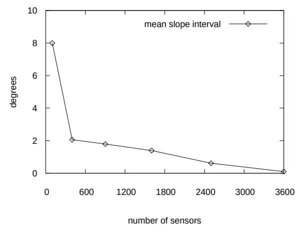
Figure 10. Worst-case scenario for back-and-forth movement along the convex hulls

5. Simulation

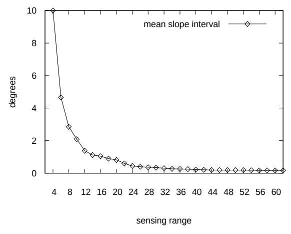
In this section, we evaluate through simulation experiments the effectiveness of our protocol. We implemented a simulation in C++ and the experiments were run on a Pentium 4 2.26GHz machine with 512MB of RAM. We focus on demonstrating the accuracy of the derived critical quadruplet in estimating the motion direction of an object. This accuracy depends in effect on the number of sensors that participate in the computation. We have formulated two variables which affect this number: (i) the total number of sensors in the same field, i.e. the sensor density, assuming a sensing range that encompasses the whole field for all sensors, and (ii) the sensing range itself, assumed to be equal for all sensing devices.

In our first experiment we have tuned the number of sensors in the same square-shaped experimental field, i.e., the uniform density of sensors. We assume that the communication range satisfies the uniform disk connectivity assumption for each sensor. Specifically, leting $|\mathcal{S}|$ be the number of sensors, we generate a square field of side $\sqrt{|\mathcal{S}|}$ and set the radio range to 3. For this experiment we have assumed that the motion sensing range encompasses the whole field for all sensors, thus all sensors generate $+$ or $-$ values whenever a moving object is present in the workspace. For each number of sensors, we averaged the size of the estimated slope intervals for a target object that assumes 10 different directions during its motion, while the network calculates one estimate for each different direction. Figure 11a depicts the results.

In our second experiment (Figure 11b), we have tuned the sensing range in the same 2-dimensional experimental field of 3600 uniformly distributed sensing devices. The field is square-shaped with each dimension being 60 units long. The sensing range varies from 4 to 62 units. In effect, the number of nodes that generate a bit ($+$ or $-$), increases and eventually reaches the total number of nodes (3600). As before, we set the communication range to 3. For each sensing range, we averaged the size of the estimated slope intervals for a target object that assumes 10 different directions



(a) Accuracy vs. Density



(b) Accuracy vs. Range

Figure 11. Variation of Accuracy depending on Sensor Density and Sensing Range

during its motion, while the network calculates one estimate for each different direction. Based on the simulation results, we conclude that the system achieves a high accuracy of estimation (less than 2 degrees of variation) when the number of participating nodes (sensing range, respectively) exceeds a reasonable amount of 400 nodes (10 units, respectively), with $\mathcal{S}_o$ being in the order of few tens.

6. Conclusions

In this paper we have introduced a novel, robust, and effective model for estimating the direction of motion of a target moving within the field of a binary sensor network. Our model is based on a minimalist pragmatic approach about the sensing capacities of the nodes, while it eschews the drawback of centralized processing which burdened previous approaches [2]. All computations are performed within the network, while the communication requirements between individual nodes are minimal. We have exploited the geometric characteristics of the network to a full extent and observe that the resulting error in direction estimation is low. Besides, in contrast to [2], our model is capable of estimating the direction of a target's motion from single instance of observations; it does not require multiple successive inputs of the sensors' readings. Moreover, the algorithmic burden assigned to individual sensor nodes is minimal and compatible to their computational capabilities. A computation that would amount to a convex hull algorithm if executed centrally is shared among different sensor nodes for each detection instance, imposing affordable computation demands on each. In a series of detections, different nodes participate in every new computation, so that the long-term power resource burden is equally shared by all nodes for objects following divergent motion paths. Our model can handle multiple moving targets if these are distinguishable based on their signatures. A future challenge is to develop heuristics that will enable the system to distinguish between different moving targets which are distinctly observed but not identifiable by their signatures.

References

[1] A. Arora, P. Dutta, S. Bapat, V. Kulathumani, H. Zhang, V. Naik, V. Mittal, H. Cao, M. Demirbas, M. Gouda, Y.-R. Choi, T. Herman, S. S. Kulkarni, U. Arumugam, M. Nesterenko, A. Vora, and M. Miyashita. A line in the sand: A wireless sensor network for target detection, classification, and tracking. *Computer Networks (Elsevier)*, 46(5):605–634, December 2004.

[2] J. Aslam, Z. Butler, F. Constantin, V. Crespi, G. Cybenko, and D. Rus. Tracking a moving object with a binary sensor network. In *Proc. of SenSys Conf.*, 2003.

[3] R. R. Brooks, P. Ramanathan, and A. M. Sayeed. Distributed target classification and tracking in sensor networks. *Proceedings of the IEEE*, 91(8):17–30, August 2003.

[4] M. Chu, H. Haussecker, and F. Zhao. Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks. *International Journal of High Performance Computing Applications*, 16(3), 2002.

[5] M. de Berg, M. van Kreveld, M. Overmars, and O. Schwarzkopf. *Computational geometry: algorithms and applications*. Springer-Verlag New York, Inc., 1997.

[6] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P. Nordlund. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on Signal Processing*, 50:425–435, February 2002.

[7] J. Heidemann and R. Govindan. An overview of embedded sensor networks. Technical Report ISI-TR-2004-594, USC/Information Sciences Institute, November 2004.

[8] D. Li, K. Wong, Y. Hu, and A. Sayeed. Detection, classification, and tracking of targets in distributed sensor networks. *IEEE Signal Processing Magazine*, 19(2):17–30, March 2002.

[9] J. Liu, P. Cheung, F. Zhao, and L. Guibas. A dual-space approach to tracking and sensor management in wireless sensor networks. In *Proc. of ACM WSNA*, 2002.

[10] S. Oh and S. Sastry. Tracking on a graph. In *Proc. of IPSN*, 2005.

[11] S. Pattem, S. Poduri, and B. Krishnamachari. Energy-quality tradeoffs for target tracking in wireless sensor networks. In *Proc. of IPSN*, 2003.

[12] Y.-C. Tseng, S.-P. Kuo, H.-W. Lee, and C.-F. Huang. Location tracking in a wireless sensor network by mobile agents and its data fusion strategies. In *Proc. of IPSN*, 2003.

[13] Q. Wang, W.-P. Chen, R. Zheng, K. Lee, and L. Sha. Acoustic target tracking using tiny wireless sensor devices. In *Proc. of IPSN*, 2003.

[14] H. Yang and B. Sikdar. A protocol for tracking mobile targets using sensor networks. In *Proc. of IEEE Workshop on Sensor Network Protocols and Applications*.

[15] F. Zhao and L. Guibas. *Wireless Sensor Networks: An Information Processing Approach*. Elsevier/Morgan-Kaufmann, 2004.

[16] F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration. *IEEE Signal Processing Magazine*, 19(2):61–72, March 2002.