GPS-Free Node Localization in Mobile Wireless Sensor Networks

Hüseyin Akcan
CIS Department
Polytechnic University
hakcan01@cis.poly.edu

Vassil Kriakov
CIS Department
Polytechnic University
vassil@cis.poly.edu

Hervé Brönnimann
CIS Department
Polytechnic University
hbr@poly.edu

Alex Delis
Dept. of Informatics & Telecom.
University of Athens
ad@di.uoa.gr

ABSTRACT

An important problem in mobile ad-hoc wireless sensor networks is the localization of individual nodes, i.e., each node's awareness of its position relative to the network. In this paper, we introduce a variant of this problem (*directional* localization) where each node must be aware of both its position *and* orientation relative to the network. This variant is especially relevant for the applications in which mobile nodes in a sensor network are required to move in a collaborative manner. Using global positioning systems for localization in large scale sensor networks is not cost effective and may be impractical in enclosed spaces. On the other hand, a set of pre-existing anchors with globally known positions may not always be available. To address these issues, in this work we propose an algorithm for directional node localization based on relative motion of neighboring nodes in an ad-hoc sensor network without an infrastructure of global positioning systems (GPS), anchor points, or even mobile seeds with known locations. Through simulation studies, we demonstrate that our algorithm scales well for large numbers of nodes and provides convergent localization over time, even with errors introduced by motion actuators and distance measurements. Furthermore, based on our localization algorithm, we introduce mechanisms to preserve network formation during directed mobility in mobile sensor networks. Our simulations confirm that, in a number of realistic scenarios, our algorithm provides for a mobile sensor network that is stable over time irrespective of speed, while using only constant storage per neighbor.

Categories and Subject Descriptors: C.2.1 [Network Architecture and Design]: Wireless Communication
General Terms: Algorithms.
Keywords: localization, mobility, sensor networks.

MobiDE'06, June 25, 2006, Chicago, Illinois, USA.

1. INTRODUCTION

Wireless sensor networks are composed of hundreds, possibly thousands, of tiny low-cost devices - *sensor nodes* that are capable of measuring various physical values, performing computations and, most importantly, communicating with each other and organizing themselves in order to cooperatively achieve a desired task [17].

An important aspect in most of the sensor networks application is the *localization* of the individual nodes [9]. For example, in aggregation networks, the node localization is needed in order to construct topology-aware routing structures that will increase message efficiency and reliability, and reduce transmission costs [8]. Broadly speaking, localization in wireless sensor networks is the problem of individual sensor's awareness of their position relative to a coordinate system common to the entire sensor network. In general, location awareness empowers routing algorithms to determine the most efficient message paths [12], or to achieve goals such as optimal area coverage [14]. In routing applications, it is sufficient for the nodes to know the positions of their neighbors *relative* to a local coordinate system [6]. We call this *relative* localization since the orientation of each sensor is completely independent of the network coordinate system. To support mobility applications, a node must move in a specific direction in a manner that is related to its neighbors. We call *directional* localization the problem of determining both the position *and* the orientation of each sensor in the common coordinate system.

Solutions for various problems of interest in which the localization is an important aspect traditionally rely on two assumptions: (1) the availability of global positioning systems (GPS) – which requires additional hardware at additional costs; and (2) the availability of a number of fixed-point reference nodes, or anchors, with globally known locations [7]. This is most commonly used in static networks [15] with recent efforts on mobile networks where a small subset of the moving nodes (seeds) are aware of their global positions [11].

Many applications require sensor network mobility in environments where GPS signals may not be available and pre-existing infrastructures do not exist. Consider a fire search mission inside a building where a set of mobile nodes explore a floor with the goal to locate the source of fire. The nodes move collaboratively, in a semi-rigid swarm. The swarm follows a path such that it covers the area while taking temperature measurements. To tackle the problem of localization management in such GPS-free settings in where the nodes are mobile, a number of issues must be taken into consideration. Most importantly, because of sensor mobility, the additive error in the estimated location can accrue to fairly high values. This is a consequence of mechanical errors in evaluating the direction and distance of movement, which may occur in every measurement. The sources for this type of errors are due to manufacturing defects or fluctuations in the environment (e.g., wind). Thus, as the motion evolves, the uncertainty on the position and direction of a node increases.

The main contribution of this work is that it presents a solution to the problem of *directional* localization in GPS-free sensor networks with mobile nodes. We introduce a novel, motion-based algorithm for node position and direction calculation with respect to each individual node's local coordinate system in mobile ad-hoc sensor networks, without global positioning information. Our algorithm works fast, in one step of the movement and does not require additional memory; in addition, it is not affected by cumulative position errors. More specifically, we propose an algorithm which

- provides *directional* neighbor localization in a network-wide coordinate system,
- works under fairly large motion and distance measurement errors,
- is unaffected by the speed of nodes,
- works for any network size,
- supports a stable network in mobility problems.

To experimentally validate our algorithm, we built a simulation framework. We analyzed the impact of the direction and distance errors on the location estimation errors, and our experiments based on the simulation demonstrated that the maximum localization error in a number of executional scenarios is bounded. We show how our algorithm can be utilized to create stable and structured sensor networks without an underlying infrastructure and without expensive positioning devices.

Although our algorithm can be used to construct routing topologies with various protocols, throughout this paper we will focus on *mobility* aspects. In particular, we introduce mechanisms for building a network of sensors to achieve a common goal such as following a path in a semi-rigid formation. Our simulations confirm that the proposed techniques preserve the network's stability over time while providing constant localization errors.

The remainder of this paper is structured as follows. Section 2 discusses related work. Our algorithm is described in Section 3, while Section 4 describes its use in mobility applications. Our experimental observations are presented in Section 5 and Section 6 provides concluding remarks and outlines directions for future research.

2. RELATED WORK

A majority of previous research works related to localization problems have primarily focused on static sensor networks [15]. Recently, however, more attention has been paid to mobile environments. Problems in mobile sensor networks have been investigated mainly in conjunction with a particular positioning infrastructure (anchors, seed nodes, beacons) or under random movement scenarios [6].

Low precision for close range and limited coverage (especially indoors) of GPS systems led researchers to explore GPS-free localization for mobile nodes. One common technique used is to exploit wireless communication. Bulusu *et al.* [3] use known reference points to send periodic beacon messages. By receiving beacons from enough sources , nodes can localize themselves. The accuracy of the localization depends on the distance to the reference points. Priyantha *et al.* [15] also use beacons for localization, but they assume the real locations of the reference points are unknown. The problem of calculating global geometry from local information is proved to be NP-hard [16]. For static nodes, and only using Euclidean distances, Bădoiu *et al.* [1] propose a constant factor, quasipolynomial-time approximation algorithm. The algorithm requires complete graph information, and is not practical for low processing power nodes.

In [6], *relative* localization in mobile sensor networks is accomplished through triangulation of neighbor nodes using a common one-hop neighbor. The authors propose algorithms for building a relative coordinate system based on a central node, or a dense group of nodes called Location Reference Group. Although this work is similar to ours in that it estimates positioning in a mobile environment without seed nodes, its primary focus is on negotiating a relative coordinate system for the entire network. While this solution finds applications in routing protocols, it is not applicable in mobility scenarios where directed motion is required.

In [11] a sequential Monte Carlo method is used to probabilistically estimate the locations of nodes in a network with a few seeds. Seeds are those nodes which know their precise location, through the use of GPS, for example. Due to the model's dependence on the previous estimates, the location errors are cumulative and a re-sampling step must be introduced. The re-sampling process requires each node to collect as much as fifty samples before a good estimate can be made. A method based on predictions is presented in [13], where nodes in the network use a dead reckoning model to estimate the movements of all other nodes. Position information is adjusted for granularity so that distant pairs of nodes maintain less accurate position information than pairs which are closer to each other.

Concerning distance and motion detection error, Rayleigh fading may introduce significant errors due to the motion of the sensor in cases where signal strength is used for neighbor distance estimation. This problem is studied in [2], where the location estimation is based on power measurement of signals received from two anchored beacons with known locations. The authors explore how the speed of mobile nodes detrimentally affects their localization accuracy. The mechanisms introduced in [2] can complement our work to improve the neighbor distance measurement error for high speed sensors. Distance measurement methods are surveyed in [4]. We use Time of Arrival (TOA) for neighbor distance measurements. The TOA method finds the distance between a transmitter and a receiver through the use of one way propagation time.

This paper introduces a directional localization algorithm which exploits node mobility to calculate neighbor positions without the use of any global information (e.g. GPS, anchor points, and seed nodes). We present mechanisms to utilize this algorithm in realistic mobility scenarios. Our methods are deterministic and no past-position information is stored; therefore there is no cumulative position estimation error.

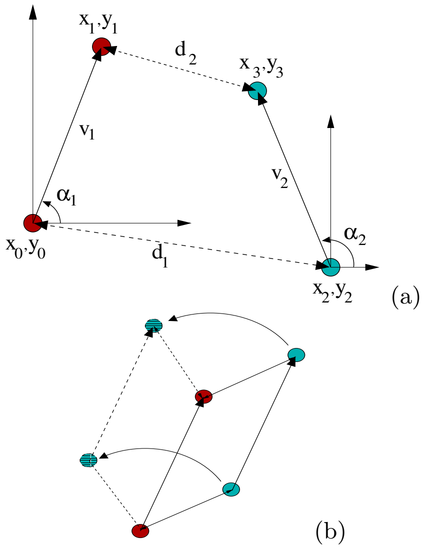

Figure 1: Typical movement of two nodes, with angles and distances (a). An example non-rigid geometry, where nodes move parallel keeping the distances exactly. Nodes can have infinite positions around each other (b).

3. LOCALIZATION ALGORITHM

In this section, we present our GPS-free localization algorithm. The algorithm works under the following assumptions:

- Each node has a compass pointing North (or any other common reference direction)
- Nodes can measure the distance to their neighbors using a well known range measurement method (e.g. Time of Arrival (TOA) [4])
- Motion actuators allow each node to move a specific distance in a specific direction (with respect to North)
- Actuator, compass and distance measurements are subject to errors caused by various real world disturbances
- Other than the above, no additional positioning equipment or infrastructure is required.

First, we describe our core localization algorithm with two neighbors, n_1 and n_2, that generates two possible relative positions. Later we discuss our verification algorithm which uses a common third neighbor to select the correct solution.

Core localization algorithm. The core localization algorithm works on well defined *rounds*, where each round essentially consists of three steps:
1. It begins with distance measurement between neighbors,
2. It continues with individual movement of the nodes,
3. It ends with an exchange between neighbors of direction and distance values for that round.

Rounds are initiated by nodes whenever they need localization. We do not require any other continuity or pattern between rounds. We also do not assume anything about the temporal duration of the rounds, however, we do assume that the nodes do not change their directions within a round.

A typical movement of two nodes n_1 and n_2 in a round is shown in Figure 1(a). At time t_1, n_1 is at position (x_0, y_0) and n_2 at (x_2, y_2), and the nodes measure the initial inter-distance d_1. Between time t_1 and t_2, each node $\{n_i \mid i = 1, 2\}$ moves in a direction α_i and covers a distance v_i. At time t_2, the nodes, now at positions (x_1, y_1) and (x_3, y_3), calculate their inter-distance d_2 and exchange v_i and α_i information. After receiving all the information, each node selects itself as the origin and calculates the position and direction of the other node, in its local coordinate system. To solve the equations in the local system of n_1, we choose the position (x_0, y_0) of n_1 as the origin and write:

$$x_1 = v_1 \cos \alpha_1, \qquad y_1 = v_1 \sin \alpha_1, \tag{1}$$

$$x_3 = x_2 + v_2 \cos \alpha_2, \qquad y_3 = y_2 + v_2 \sin \alpha_2, \tag{2}$$

$$(x_3 - x_1)^2 + (y_3 - y_1)^2 = d_2^2, \qquad x_2^2 + y_2^2 = d_1^2. \tag{3}$$

Substituting equations (1) and (2) into equation (3), we get:

$$x_2 A + y_2 B = C, \tag{4}$$

with the appropriate definitions:

$$A = v_2 \cos \alpha_2 - v_1 \cos \alpha_1, \qquad B = v_2 \sin \alpha_2 - v_1 \sin \alpha_1,$$

$$C = \frac{1}{2}\left(d_2^2 - d_1^2 - v_1^2 - v_2^2 + 2 v_1 v_2 \cos(\alpha_1 - \alpha_2)\right).$$

Substituting $x_2 = (C - y_2 B)/A$ and $y_2 = (C - x_2 A)/B$ into $x_2^2 + y_2^2 = d_1^2$, we get:

$$x_2^2 D - 2 x_2 E + F = 0, \qquad y_2^2 D - 2 y_2 G + H = 0, \tag{5}$$

again with the appropriate definitions:

$$D = A^2 + B^2, \qquad E = AC, \qquad F = C^2 - d_1^2 B^2,$$

$$G = BC, \qquad H = C^2 - d_1^2 A^2.$$

Note that the coefficient of x_2^2 and y_2^2 is the same in both equations (5), namely, D.

Using (5), each variable solves independently to

$$x_2 = \frac{E \pm \sqrt{E^2 - DF}}{D}, \qquad y_2 = \frac{G \pm \sqrt{G^2 - DH}}{D} \tag{6}$$

and solutions can be paired up by using equation (4), as long as $D \neq 0$. In practice, one would compute either x_2 or y_2 using (5) and deduce the other variable using (4). When $A = 0$ but $B \neq 0$, one would compute x_2 using (6), and when $A \neq 0$ but $B = 0$, one would compute y_2 using (6) instead. If both $A = B = 0$, then $D = 0$ and we have an *exceptional* configuration (further discussed below).

The core localization algorithm to calculate the position of n_2 from n_1 is presented in Figure 2. Solving the equations, each node finds two possible positions for each of its neighbors. Since only one of these solutions is realistic (the other one is due to "symmetry"), each node has to complete a verification step, this time using an additional common neighbor (n_3).

Verification algorithm. In Figure 2, we give the algorithm to verify a neighbor's position using a third neighbor. After solving equations (4) and (6) in the previous section, node n_1 has two position estimates $\{n_j^{1,2} \mid j = 2, 3\}$ for each of its neighbors n_2 and n_3. In order to find the positions and direction, n_1 retrieves the inter-distance $d_{2,3}$ of n_2 and n_3 from either one of these nodes, and simply finds the correct pair of positions $\{n_j^{1,2} \mid j = 2, 3\}$ that has a matching inter-distance.

CoreLocalization(n_1, n_2, v_1, α_1)

$d_1 \leftarrow$ inter-distance(n_1, n_2)
Move node n_1 by v_1 and α_1
$d_2 \leftarrow$ inter-distance(n_1, n_2)
Retrieve v_2 and α_2 from n_2
Calculate positions of n_2 using equations (4),(5) and (6)

Verification(NeighborList NL)

for each neighbor pair (m, n) in NL **do**
  if m and n are neighbors **then**
    $d_{m,n} \leftarrow$ measured inter-distance(m, n)
    for each position pair $\{m^i, n^j \mid i, j = 1, 2\}$ **do**
      Compute Euclidean distance D between m^i and n^j
      if $D = d_{m,n}$ **then**
        mark m^i and n^j as exact positions

Figure 2: Core localization algorithm for n_1: calculates two possible positions for n_2. Verification algorithm evaluates the position estimations of neighbor nodes such that only 1 out of 4 position pairs validates the distance.

For rigid geometries and configurations without errors, there can only be one pair verified. However, for configurations with errors, we relax the algorithm slightly to select the pair with the closest inter-distance value to $d_{2,3}$.

Exceptional configurations. The above localization algorithm works for rigid geometries where two possible positions per neighbor are estimated. However, there exist exceptional movement configurations when the core algorithm cannot find any meaningful results, namely *equal parallel movement* and *excessive error* configurations.

Equal parallel movement configurations occur for $D = 0$ in equation (6). This also implies that $A = 0$ and $B = 0$ since $D = A^2 + B^2$. An example *equal parallel movement* configuration is shown in Figure 1(b). In this case, the nodes move in parallel and keep the exact same distances (d and v) between them, so that node n_2 can be anywhere on a circle at a distance d away from node n_1, and vice versa. The geometry is not rigid and infinitely many possible solutions exist for both neighbors.

The other exceptional configuration is the *excessive error* configuration. The main sources of error in our algorithm occur due to distance, actuator and compass measurement inaccuracies. When highly erroneous d, v and α values create a non-rigid geometry, such that $E^2 - DF < 0$ or $G^2 - DH < 0$ in equation (6), our core algorithm cannot localize n_1 and n_2.

Although it is hard to avoid the above exceptional configurations, they can be detected easily within the core algorithm. Once detected, nodes can skip that round and can make necessary adjustments (e.g. random changes) to their speed and direction to avoid the same ill-configuration in the next round.

In this section, we presented the core localization algorithm, verification algorithm and possible ill-configurations where localization is not possible. In Section 5, we study our algorithms behavior in various settings, such as random movement, directed movement and error scenarios. We also give practical evidence that the ill-configurations do not dominate our algorithms behavior, and even though they do occur, our algorithm successfully detects and recovers from these configurations.

MoveNode(Node N, NeighborList NL, DirectionVector $\vec{D}$, int k, RangeFactor RF)

$\vec{V} \leftarrow 0$
$count \leftarrow 0$
for each localized neighbor n in NL **do**
  ◃ $\vec{u}_{N,n}$ *is the vector from N to n*
  $\vec{V} \leftarrow \vec{V} + \vec{u}_{N,n}$
  $count \leftarrow count + 1$
if $count < k$ **then**
  $RF \leftarrow$ RF /2
$\vec{V} \leftarrow (RF * range(N) * \vec{V} + \vec{D})/(count + 1)$
Move node N by $\vec{V}$

Figure 3: k-neighborhood mobility algorithm.

4. SENSOR NETWORK MOBILITY

Our directional localization algorithm is most useful in mobile applications where the entire network must move in a specific path in order to accomplish a goal. To analyze the behavior of our localization algorithm in a realistic mobility scenario, we adapt a mobility model based on the Reference Point Group Mobility (RPGM) model [10]. While considering other mobility models, as surveyed in [5], we decided to base our analysis on RPGM due to the generality of the model. In the RPGM model, the random motion of the individual nodes is modeled in relation to a randomly chosen directional motion of the entire group. Each node in the group moves randomly around a fixed reference point and the entire group of reference points moves along the group's logical center. Our localization algorithm computes locations and orientations for nodes and their neighbors. In that respect, we further generalize the RPGM model so as to make individual sensors independent of the reference points. Furthermore, because our sensor network can maintain a semi-rigid structure based solely on local positioning, it is unnecessary for nodes to be aware of the group's center and only the destination point must be specified. It is possible to remove the reference points because the individual random motion within the group is contextualized by the random motion of a node's immediate (one-hop) neighbors. In that sense, the neighbors represent the reference points of motion.

Our adapted mobility algorithm is presented in Figure 3. The network moves with respect to a direction vector $\vec{D}$. To maintain a semi-rigid formation without disconnecting the network, we impose a minimum neighbor count k that each node strives to attain. This is a best-effort k-connected algorithm where nodes attempt to maintain a neighbor distance that is a fraction RF of their wireless range. RF is adjusted dynamically with the number of neighbors so that nodes with neighbors fewer than k stay closer while still moving with the network. This avoids network partitioning. The $range$(Node N) function returns the wireless range of the given node. For brevity, we do not present the case where a boundary is reached and a new direction is calculated. However, in our simulations we implement this as a ricochet off the boundary surface.

The benefit of our approach is that while an initial direction of motion is specified for the group, the structure of the network remains cohesive but independent. An example application of this approach is a swarm of mobile sensors which move in a general pattern with a specific goal. For example, an oil-sensor network may move in a zig-zag pat-

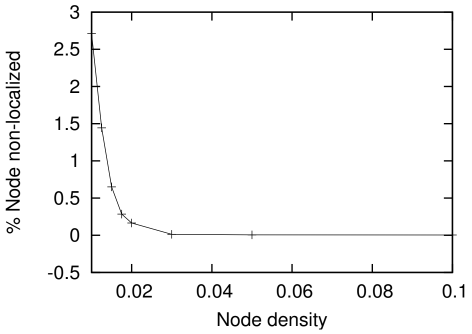

Figure 4: Percent of non-localized nodes for different node densities.

tern, with the goal to discover an oil spill and cover the contaminated area once it is found. In this example, only a virtual boundary must be specified and the network of sensors will maintain sufficient proximity to communicate, while covering the area.

The mobility algorithm presented in this section is a general network movement algorithm that requires only local position information. Because our localization algorithm requires each node to communicate only with its direct neighbors and there is no message propagation, the algorithm scales up to any network size. As shown in the experiments below, the localization error of the algorithm is only sensitive to the node density of the network.

5. EXPERIMENTS

In this section, we first present the results for our localization algorithm, free from measurement errors. After the initial results, we introduce independent errors on angle and distance measurements to simulate real world disturbances. We present how our algorithm behaves under such errors, and possible ill-configurations. Finally, we compare our localization algorithm to an absolute positioning algorithm in random and directed mobility scenarios.

Experiments under ideal conditions. In this experiment we simulate nodes randomly placed in a 100x100 area. Each simulation is run for 100 rounds, and the results are averaged. At each round, nodes perform a random walk with random speed $[0, 5)$, random angle $[0, 2\pi)$ and fixed radio range of 6. Node density represents the number of nodes over the total deployment area. As described in Section 3, our localization algorithm requires two neighbors to accurately find neighbor positions. Figure 4 displays the percentage of nodes, whose positions are not calculated accurately for different node densities. For small node densities, we observe that not all nodes can be localized. The reason is that nodes do not have neighbors to calculate positions, or do not have common neighbors. As we can see from Figure 4, the percent of non-localized nodes approaches zero for densities greater than 0.02. From this graph we can conclude that our algorithm calculates almost all nodes positions for dense networks, and introduces minor node localization failures, as small as 3%, for sparse networks.

Introducing measurement errors. Previously we presented how our algorithm behaves under ideal conditions. Now we relax these assumptions and introduce errors on distance and angle measurements. In real world, measurements may be quite inaccurate due to weather, terrain conditions and equipment failures. To simulate these errors, we add uniform random noise to all our measurements. For distance measures we add percent error relative to the measured value, and for angle measures we add absolute percent error (percent of 2π) to the measured value. Introducing the errors changes our algorithm's behavior in one of two ways: (1) the algorithm calculates the positions with limited accuracy; or (2) *excessive error* configurations (defined in Section 3) prevent the algorithm from localizing some of the nodes. Figure 5 (left) shows the average position error of our algorithm for different values of noise on angle and distance measurements. The effects of *excessive error* configurations on our algorithm appear in Figure 5 (right). From this figure we can see that even with 30% noise on angle and distance measures, which is a quite high error rate for real world conditions, the number of non-localized nodes is at most 16%. Based on these results, we claim that our algorithm provides sufficient node localization. We provide additional support for this claim by testing our algorithm in random and directed movement scenarios below.

Comparison with an absolute positioning algorithm. In mobility scenarios we compare our localization algorithm to an absolute positioning algorithm. In such an algorithm, we assume that nodes know their initial positions in the deployment area, thanks to an anchor point or any other positioning infrastructure. We also assume that once nodes get their initial position, they do not receive any additional positioning information, relative or absolute. In order to know their absolute positions, nodes keep track of their own movements. By exchanging location information with immediate neighbors, each node is able to keep track of the positions of others. This scenario occurs in real world when nodes are deployed from a known point and asked to explore a possibly big area where they cannot keep a direct connection to the deployment point. Note that our algorithm, as described in Section 3, does not use any absolute position information.

We simulate two different mobility scenarios. The first is based on random movement, where 100 nodes with fixed radio range 15 can cover a distance of at most 5 units per round. The second scenario is the directed movement described in Section 4. Nodes sweep the area in a zig-zag manner, with radio range 5 and maximum per-round distance 3. An example trajectory of the nodes in the directed movement scenario is shown in Figure 8. There is no global path information available, rather, the nodes detect the boundaries of the environment and make movement decisions as a response to these environmental readings.

In Figure 6 we show the average errors for both algorithms over increasing number of rounds, for two different uniform random noise levels: high and low, in random motion (top) and directed motion (bottom) scenarios. High noise level is up to $\pm 30\%$ of distance measurements and up to $\pm 2\pi/10$ of angle measurements. Low noise level is up to $\pm 3\%$ of distance measurements and up to $\pm 2\pi/100$ of angle measurements. Since our algorithm calculates distances within each round and does not use any cumulative data, the er-

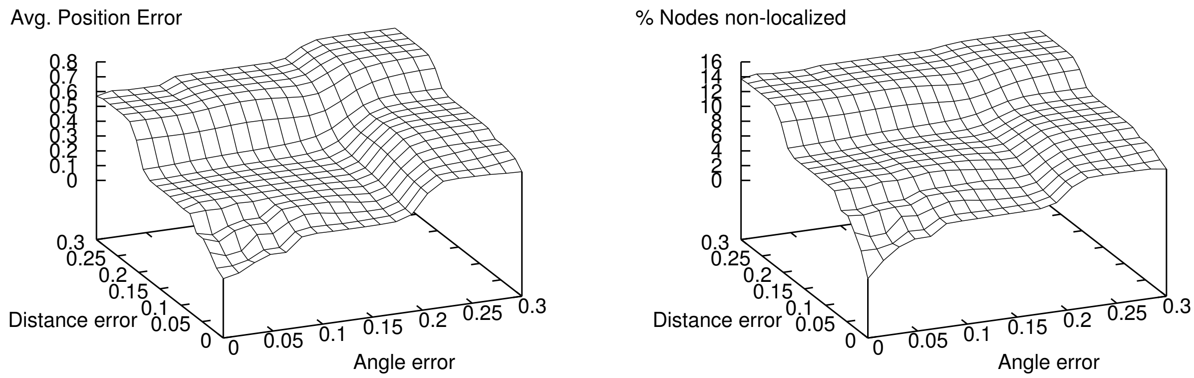

Figure 5: Effects of angle and distance measurement noise on position error (left), and percent of non-localized nodes (right).

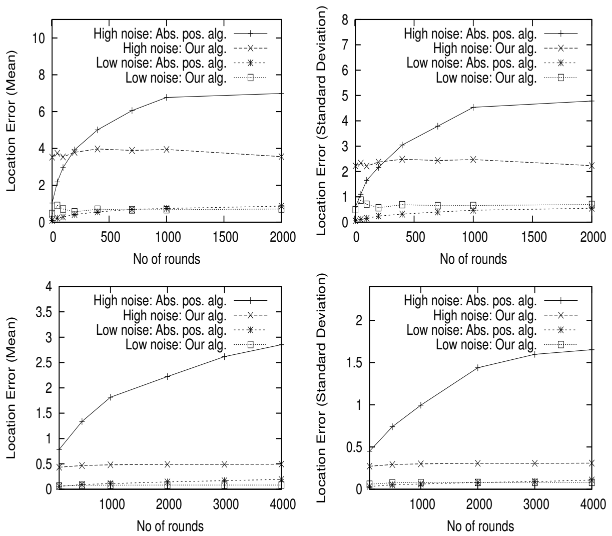

Figure 6: Mean and standard deviation of position error vs. number of rounds for our algorithm and the absolute positioning algorithm, using different levels of noise. Random movement (top), directed movement (bottom).

Movement	Noise	Our alg. Mean	Our alg. Stdev	Abs. pos. alg. Mean	Abs. pos. alg. Stdev
Random	low	0.71	0.69	0.86	0.55
	high	3.55	2.23	6.98	4.78
Directed	low	0.08	0.08	0.19	0.11
	high	0.49	0.30	2.85	1.65

Table 1: Mean and standard deviation errors of our algorithm and the absolute positioning algorithm, as shown in Figure 6.

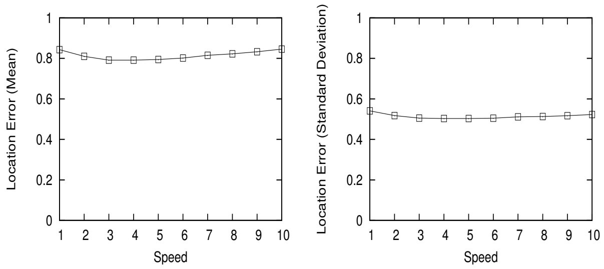

Figure 7: Mean and standard deviation of position error vs. speed of nodes with a wireless range of 10 units and speed measured in units per round.

ror of the algorithm is nearly constant over the number of rounds, for both scenarios. Although the absolute positioning algorithm starts with a low error value (more apparent in random motion), small measurement errors accumulate over each round and cause an ever-increasing error. The results of our simulations for random and directed movement are summarized in Table 1. The mean error of our algorithm is as much as 5 times lower than the mean error of the absolute positioning algorithm. The high values of standard deviations in the absolute positioning algorithm reflect on the effects of accumulative errors and show that it is not a robust solution for high noise scenarios. On the other hand, our localization algorithm provides consistent behavior in the above-mentioned scenarios.

In order to evaluate the effects of movement speed and wireless range we tested our algorithm under high noise, with a fixed wireless range (10) and variable speed values. We increased the speed only up to the wireless range distance per unit time. In any sensor network scenario, if a node moves by a distance grater than its wireless range in a unit time, it is highly probable that its neighborhood will change at each step, which would make it impossible to localize. We can see in Figure 7 that the localization error of our algorithm is nearly constant for increasing node speeds. The maximum speed supported by our algorithm is the wireless range distance per unit time which is 10 units per round in this experiments.

In Figures 9 and 10 we present the snapshots of the simulations of the absolute positioning algorithm and our algorithm, respectively, performing a zig-zag directed movement (as in Figure 8) under heavy noise. Both simulations

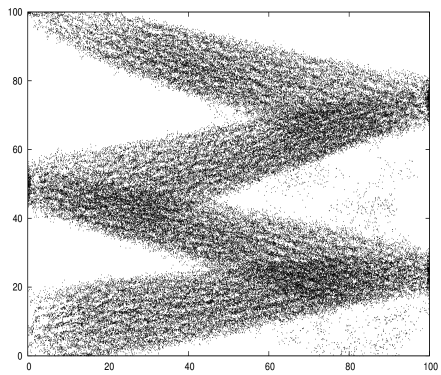

Figure 8: Directed trajectory of nodes performing zig-zag movement.

use the same movement algorithm as described in Section 4. Because of the cumulative errors, the absolute positioning algorithm is not capable of maintaining the topology of the network and becomes disorganized. On the other hand, our algorithm maintains connectivity at all times while forming a nice semi-rigid topology. The results in Figures 6 and 10 also support our claim that even under high noise settings, ill-configurations do not deteriorate our algorithm's behavior; effects of these are constant through rounds. As seen in Figure 10, occasionally a few nodes (two in this case) become disconnected from the network while running our algorithm. This happens when nodes near the network border cannot be localized. Although the number of these nodes is small, they can be further controlled by forcing stricter k-neighborhood rules.

In this section, we presented simulation results for our algorithm under ideal conditions as well as with simulated real world error and noise conditions. For ideal conditions, we find our algorithm quite accurate. Even under heavy noise and measurement errors, the algorithm can keep a near constant error bound over time.

6. CONCLUSION AND FUTURE WORK

To the best of our knowledge, our localization algorithm is the first GPS-free work on mobile nodes that only uses wireless communication properties and a compass to find positional *and* directional locations of neighbor nodes. We propose a straightforward and robust algorithm that requires only a single round of node movement to localize all neighbor nodes. Our algorithm only requires constant storage per one-hop neighbor during localization. Therefore, it is immune to common location errors accumulated through time, without any infrastructural support (eg. GPS, anchor point). The self-localization ability of our algorithm makes it cost effective and easy to deploy to areas where no global positioning infrastructure is available, such as indoor or disaster areas.

We are also exploring future applications for our algorithm. One area where our algorithm will contribute is when a stricter control of the geometric formation of the network is needed for mobility and coverage applications. This is useful for applications such as increased coverage during mobility, surrounding or covering a target area.

7. REFERENCES

[1] M. Badoiu, E. D. Demaine, M. T. Hajiaghayi, and P. Indyk. Low-dimensional embedding with extra information. In *Symposium on Computational Geometry*, pages 320–329, 2004.

[2] P. Bergamo and G. Mazzini. Localization in Sensor Networks with Fading and Mobility. In *Personal, Indoor and Mobile Radio Communications*, pages 750–754, 2002.

[3] N. Bulusu, J. Heidemann, and D. Estrin. Gps-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5):28–34, October 2000.

[4] J. Caffery and G. Stber. Overview of radiolocation in cdma cellular systems. *IEEE Communications Mag.*, 36(4):38–45, 1998.

[5] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing*, 2(5):483–502, 2002.

[6] S. Capkun, M. Hamdi, and J.-P. Hubaux. GPS-free Positioning in Mobile Ad Hoc Networks. *Cluster Computing*, 5(2):157–167, 2002.

[7] K. Chintalapudi, R. Govindan, G. Sukhatme, and A. Dhariwal. Ad-Hoc Localization Using Ranging and Sectoring. In *INFOCOM*, 2004.

[8] J. Considine, F. Li, G. Kollios, and J. W. Byers. Approximate Aggregation Techniques for Sensor Databases. In *ICDE*, pages 449–460, 2004.

[9] J. Hill, R. Szewczyk, A. Woo, S. Hollar, D. E. Culler, and K. S. J. Pister. System Architecture Directions for Networked Sensors. In *Architectural Support for Programming Languages and Operating Systems*, pages 93–104, 2000.

[10] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang. A group mobility model for ad hoc wireless networks. In *MSWiM '99*, pages 53–60, New York, NY, USA, 1999. ACM Press.

[11] L. Hu and D. Evans. Localization for mobile sensor networks. In *MOBICOM*, pages 45–57, 2004.

[12] Y.-B. Ko and N. H. Vaidya. Location-Aided Routing (LAR) in mobile ad hoc networks. *Wireless Networks*, 6(4):307–321, 2000.

[13] V. Kumar and S. R. Das. Performance of dead reckoning-based location service for mobile ad hoc networks. *Wireless Communications and Mobile Computing*, 4(2):189–202, 2004.

[14] S. Poduri and G. S. Sukhatme. Constrained Coverage for Mobile Sensor Networks. In *ICRA*, 2004.

[15] N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller. Anchor-free distributed localization in sensor networks. In *SenSys 2003*, pages 340–341, Los Angeles, California, USA, November 5–7 2003.

[16] Y. Yemini. Some theoretical aspects of position-location problems. In *FOCS*, pages 1–8, 1979.

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

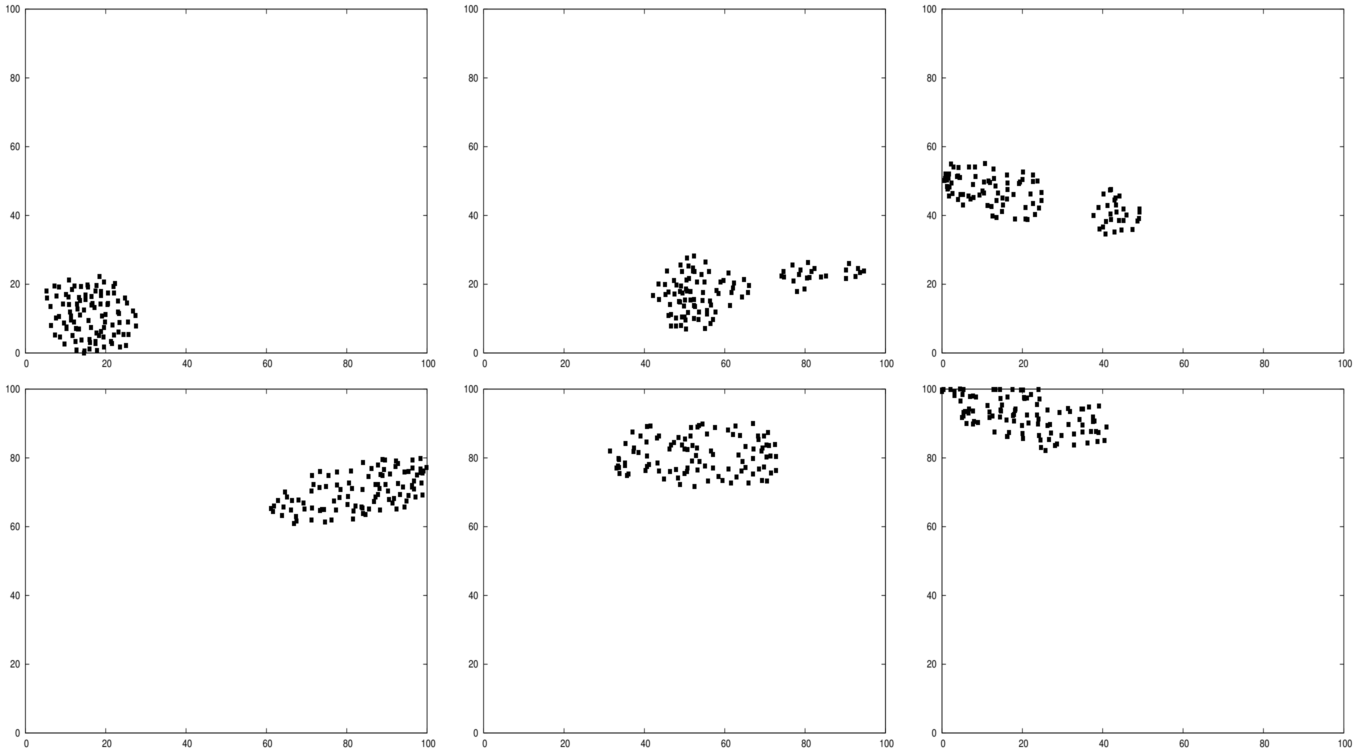

Figure 9: Snapshots of absolute positioning algorithm performing directed motion in Figure 8.

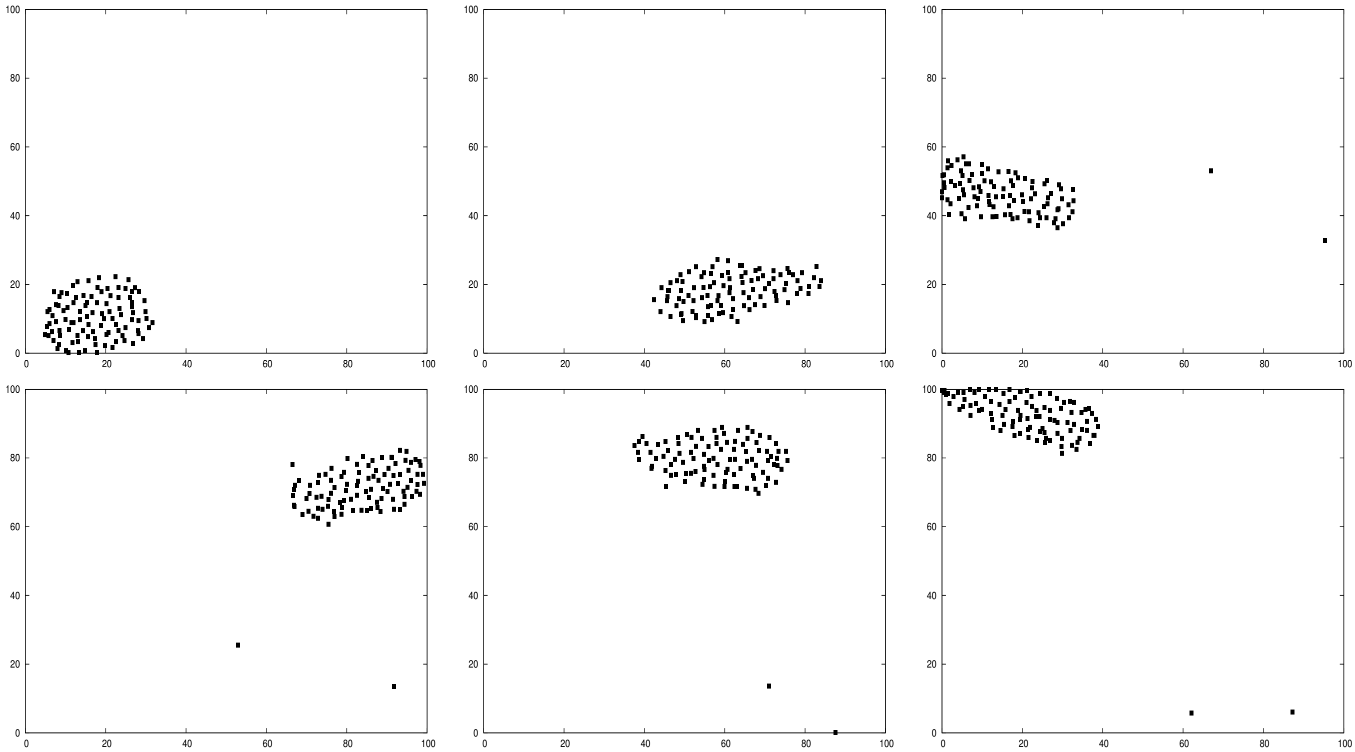

Figure 10: Snapshots of our localization algorithm performing directed motion in Figure 8.