Managing Cohort Movement of Mobile Sensors via GPS-Free & Compass-Free Node Localization

Hüseyin Akcan[1] Vassil Kriakov[2] Hervé Brönnimann[2] Alex Delis[3]

[1] *Izmir University of Economics, 35330, Balçova, Izmir, Turkey*
huseyin.akcan@ieu.edu.tr
[2] *Polytechnic Institute of New York University, Brooklyn, NY 11201*
{vassil,hbr}@cis.poly.edu
[3] *University of Athens, Athens 15784, Greece*
ad@di.uoa.gr

Abstract

A critical problem in mobile ad-hoc wireless sensor networks is each node's awareness of its position relative to the network. This problem is known as localization. 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 its neighbors. Directional localization is relevant for applications that require uniform area coverage and coherent movement. Using global positioning systems for localization in large scale sensor networks may be impractical in enclosed spaces, and might not be cost effective. In addition, a set of pre-existing anchors with globally known positions may not always be available. In this context, we propose two distributed algorithms based on *directional* localization that facilitate the collaborative movement of nodes in a sensor network without the need for global positioning systems, seed nodes or a pre-existing infrastructure such as anchors with known positions. Our first algorithm, GPS-free Directed Localization (GDL) assumes the availability of a simple digital compass on each sensor node. We relax this requirement in our second algorithm termed GPS and Compass-free Directed Localization (GCDL). Through experimentation, we demonstrate that our algorithms scale well for large numbers of nodes and provide convergent localization over time, despite errors introduced by motion actuators and distance measurements. In addition, we introduce mechanisms to preserve swarm formation during directed sensor network mobility. Our simulations confirm that, in a number of realistic scenarios, our algorithms provide for a mobile sensor network that preserves its formation over time, irrespective of speed and distance traveled. We also present our method to organize the sensor nodes in a polygonal geometric shape of our choice even in noisy environments, and investigate the possible uses of this approach in *search-and-rescue* type of missions.

1 Introduction

Wireless sensor networks are composed of hundreds, possibly thousands, of low-cost *sensor nodes* that are capable of making environmental measurements, performing computations, and communicating with one another. Most importantly, through small motors or motion actuators, these devices are capable of physically organizing themselves in order to cooperatively achieve a desired task [43].

An important problem in mobile sensor networks is each sensor's awareness of its position and direction of movement relative to the entire network. This problem is commonly known as *localization* [17]. In general, such location awareness empowers routing algorithms to determine the most efficient message paths [21], achieve goals such as optimal area coverage [31], or in mission critical applications [39]. For example, in aggregation networks [9], node localization is needed in order to construct topology-aware routing overlays that will reduce message transmission time, increase reliability, and reduce power consumption. In routing applications, it is sufficient for nodes to be aware of the coordinates of their neighbors *relative* to a local coordinate system common to the entire network [7]. We call this *relative* localization since each node's position is relative to the local coordinate system. To support mobility applications, a node must move in a specific direction in a manner that is related to its neighbors. To achieve this, in addition to knowing its neighbors' positions relative to a common coordinate axis, a node must be aware of the positions of its neighbors relative to its own direction of movement. This is the node's orientation. We call *directional* localization the problem of determining the position *and* the orientation of each sensor in the common coordinate system. In the remainder of the paper we may refer to "directional localization" as simply "localization," unless otherwise stated.

Providing support for directional localization in mobile sensor networks is a difficult task. Traditional solutions rely on information supplied externally in one of two forms: (1) via global positioning systems (GPS) – which requires additional hardware at additional costs, or (2) via fixed-point reference nodes, or anchors, whose global locations are known a-priori [8]. Such methods are most commonly used in static networks [32]. Recent efforts on mobile networks assume that only a small subset of the moving nodes (seeds) use GPS [19].

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-and-rescue* 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, taking temperature measurements while following a path that covers the area. Additional factors may exacerbate the problem further. Environmental errors must be taken into consideration, otherwise due to node mobility the additive error in the estimated location can accrue, rendering any algorithm impractical. This is a consequence of mechanical errors in evaluating the direction and distance of movement, which may occur in-between individual measurements. This type of errors can be due to manufacturing defects or fluctuations in the environment (e.g. surface friction). Thus, as the movement of the network evolves, the uncertainty of the position and direction of a node increases.

Our main contributions are distributed algorithms for solving the problem of *directional* localization in sensor networks with mobile GPS-free nodes. We introduce novel, motion-based algorithms 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 first algorithm, GPS-free Directed Localization (GDL), assumes the availability of a digital compass on each node, and calculates a node's directional localization from a single-step movement. In our second algorithm, GPS and Compass free Directed Localization (GCDL), we relax the compass requirement and compute directional localization with a 2-step motion algorithm. Our algorithms perform localization in a few steps of movement and are memoryless; in addition, they are not affected by cumulative position errors. More specifically, our proposed algorithms:

- provide *directional* neighbor localization in a network-wide coordinate system,
- can function under fairly large motion and distance measurement errors,
- are unaffected by the speed of nodes,
- support a stable network in mobility problems,
- help organize the sensor network in any polygonal shape of our choice.

To experimentally validate our algorithms, we built a simulation framework tailored to handle the mobility aspects of our algorithms. We analyzed the impact of the direction and distance errors on the location estimation errors, and our experiments in diverse operational scenarios demonstrated that the average localization errors of our algorithms are near constant throughout the movement. We show how our algorithms can be utilized to create a stable and structured swarm of sensors without an underlying infrastructure or global positioning devices. We also effectively organize nodes in any geometric shape of our choice even in the presence of localization errors.

The remainder of this paper is structured as follows. Our localization algorithms are described in Section 2, while Section 3 outlines their use in coherent mobility scenarios. In Section 4 we sketch out methods to organize the sensor network in any polygonal shape by the use of our localization algorithms. The main results of our experimentation are presented in Section 5. Section 6 discusses related work, and Section 7 provides the concluding remarks.

2 Localization Algorithms

In this section, we present our GPS-free localization algorithms. Our first algorithm, GDL, 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 known range measurement method, such as Time of Arrival (TOA) [5], Time Difference of Arrival (TDOA) [14], or Ultra-Wideband radios [12].
- 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 such as wind, rough terrain, equipment failures etc.
- Other than the above, no additional positioning equipment or infrastructure is required.

We give details of our first algorithm in Section 2.1. In Section 2.2 we present our second algorithm, GCDL, which has the exact same assumptions listed above but does not require a compass.

2.1 GPS-Free Directed Localization (GDL)

The GDL algorithm makes use of a digital compass and achieves localization in a 3-stage process termed epoch. GDL consists of two sub-algorithms; Core localization and Verification. The core localization algorithm generates two possible relative positions for each neighbor that participates in the localization, and the verification algorithm uses a third neighbor to yield the correct final solution from these two relative positions. Below, we provide a detailed description of these two algorithms.

2.1.1 Core localization algorithm

The core localization algorithm works with variable length *epochs*, where each epoch involves three distinct stages:
1. Distance measurement between neighbors,
2. Individual movement of the nodes,
3. Exchange of direction and distance values for that epoch between neighbors.

Epochs are initiated by nodes whenever they need localization. Possible causes of localization are sudden increase or decrease in the number of neighbors, which hints clustering or partitioning in the network, and may require re-positioning of the nodes. We do not require any other continuity or pattern between epochs. We also do not assume anything about the temporal duration of the epochs. However, we assume that the nodes do not change their direction of movement within an epoch.

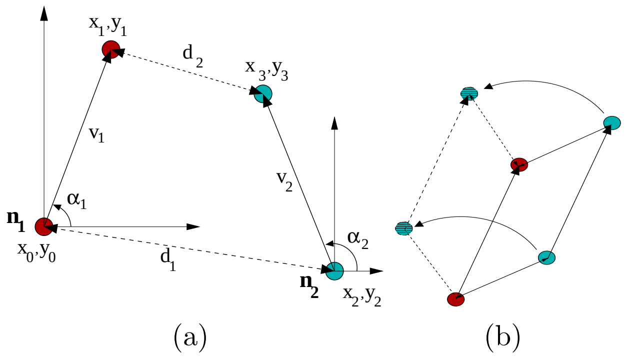


Fig. 1. Two-step movement where (a) nodes can be localized and (b) in the non-rigid geometry exceptional configuration.

A typical movement of two nodes n_1 and n_2 in an epoch 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 this 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:

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

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

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:

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

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 subsequently an *exceptional* configuration is formed; we discuss this case later in exceptional configurations.

```
CoreLocalization($n_1$, $n_2$, $v_1$, $\alpha_1$)
1: $d_1 \leftarrow$ inter-distance($n_1$, $n_2$)
2: Move node $n_1$ by $v_1$ and $\alpha_1$
3: $d_2 \leftarrow$ inter-distance($n_1$, $n_2$)
4: Retrieve $v_2$ and $\alpha_2$ from $n_2$
5: Calculate positions of $n_2$ using equations (4),(5) and (6)
```

```
Verification(NeighborPairList NPL)
1: for each neighbor pair $(m, n)$ in NPL do
2:   if $m$ and $n$ are neighbors then
3:     $d_{m,n} \leftarrow$ measured inter-distance$(m, n)$
4:     for each position pair $\{m^i, n^j \mid i, j = 1, 2\}$ do
5:       Compute Euclidean distance $D$ between $m^i$ and $n^j$
6:       if $D = d_{m,n}$ then
7:         mark $m^i$ and $n^j$ as exact positions
```

Fig. 2. The core localization algorithm calculates two possible positions. The verification algorithm determines the exact positions between neighbors.

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).

2.1.2 Verification algorithm

In Figure 2, we provide an algorithm that verifies a node's position using a third neighbor. This step is required to solve the ambiguity of two possible positions per neighbor calculated in core localization algorithm. 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 distance between n_2 and n_3 ($d_{2,3}$) 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 distance.

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

2.1.3 Exceptional configurations

The above localization algorithm works for rigid geometries where two possible positions per neighbor are estimated. Due to various real world disturbances and equipment errors, nodes do not always get a rigid geometry from their measurements. In this case equations (4) and (6) have no use. Any time the core algorithm cannot find meaningful results we reach what we term exceptional movement configurations. We distinguish two such configurations named *equal parallel movement* and *excessive error* configurations that we discuss below.

- *Equal parallel movement* configurations occur when D is equal to zero in equation (6). This also implies that $A = 0$ and $B = 0$ since $D = A^2+B^2$. An example of *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 verse. The geometry is not rigid and infinitely many possible solutions exist for both neighbors.
- The second exceptional configuration is that of *excessive error*. 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 we cannot entirely avoid the above exceptional configurations, the core localization part of GDL algorithm can readily detect them. Once detection takes place, nodes can skip that epoch and can make necessary adjustments (e.g. random changes) to their speed and direction to avoid the same ill-configuration in the next epoch.

Additional cost on hardware and unfavourable physical conditions altering magnetic field restrict the use of compass in certain hostile environments such as disaster areas. To enhance versatility, we relax the requirement for a compass and achieve localization using our algorithm that controls the mobility of the nodes. The main idea is to divide the nodes into two groups, *blue* (dark) and *red* (light), and move each group in a stepwise manner while the other remains stationary. We show that, through the use of geometric properties, we can localize the neighbors in such a 2-step motion for each group. After localization, nodes can agree on a common *virtual* north, which essentially has the same effect as having a compass. This common north resolution also enables the nodes to perform coherent movement as a network, which is one of our main goals. Furthermore, by relaxing the requirement for a compass, we relieve our GCDL algorithm from compass related failures such as measurement or equipment errors, and improve the robustness of our approach.

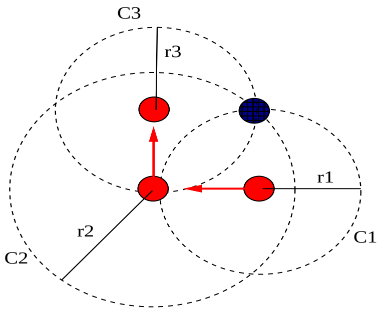


Fig. 3. The *dark* node remains fixed while the *light* node makes two steps, allowing it to localize the *dark* node.

We outline our 2-step motion algorithm (Figure 3) using a *blue* (dark) and a *red* (light) node. The *blue* node is stationary, and the *red* node performs a 2-step motion to localize the exact position of the *blue* node. Each time the *red* node communicates with the *blue* node, a virtual communication circle is formed, and the distance between the two nodes are measured using a known range measurement method. As shown in Figure 3, the first virtual communication circle (C1) results in infinite possibilities for the position of the *blue* node. The *red* node can reduce the possible positions to two by calculating the intersection of the first and the second (C2) virtual communication circles. After the second step of the *red* node, the third (C3) virtual communication circle is formed, allowing the *red* node to calculate the exact location of the *blue* node. Therefore, in order to exactly find the position of the *blue* node, three virtual communication circles are needed. By keeping track of its own movement distance, in the worst case, the *red* node gathers enough data to localize the *blue* node only after 2-step motion. Note that if the movement of the *red* node is directly towards (or away from) the *blue* node, one step of the motion is sufficient for localization, since in this case the geometry forms two mutually tangent circles within each other, which intersect at the exact location of the *blue* node.

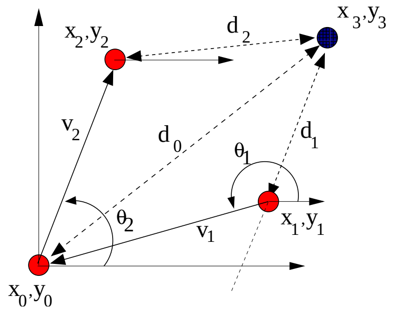


Fig. 4. GCDL Geometry: the *light* node moves from x_1, y_1 to x_0, y_0 and x_2, y_2 in order to determine x_3, y_3, the position of the stationary *dark* node.

In order to outline the geometric calculations necessary for localization after 2-step motion we assume the *blue* node in Figure 4 is stationary at position $(x_3,\ y_3)$, and the *red* node moves to positions $(x_1,\ y_1)$, $(x_0,\ y_0)$, and $(x_2,\ y_2)$ respectively. We also assume that each node has a local coordinate system, based on an arbitrary north, and nodes can mechanically track their movements relative to this local coordinate system. For example nodes can mechanically turn 90° based on their local north without requiring a compass. The movement of the *red* node is represented by a $< distance, angle >$ pair, where the angle is relative to the local coordinate system of the *red* node. The first movement from position $(x_1,\ y_1)$ to $(x_0,\ y_0)$ is represented as $< v_1, \theta_1 >$, and the second movement from position $(x_0,\ y_0)$ to $(x_2,\ y_2)$ is represented by $< v_2, \theta_2 >$ pair. For each position of the *red* node, we write the following formulas:

$$(x_3 - x_1)^2 + (y_3 - y_1)^2 = d_1^2, \tag{7}$$

$$(x_3 - x_2)^2 + (y_3 - y_2)^2 = d_2^2, \tag{8}$$

$$x_3^2 + y_3^2 = d_0^2. \tag{9}$$

We write the first $(x_1,\ y_1)$ and last positions $(x_2,\ y_2)$ of the *red* node as:

$$x_1 = v_1 \cos(\theta_1 - \pi), \tag{10}$$

$$y_1 = v_1 \sin(\theta_1 - \pi), \tag{11}$$

$$x_2 = v_2 \cos\theta_2, \tag{12}$$

$$y_2 = v_2 \sin\theta_2. \tag{13}$$

Rewriting Eq.(7) and substituting Eqs.(9,10, 11) we get:

$$(x_3 - x_1)^2 + (y_3 - y_1)^2 = d_1^2,$$

$$x_3^2 - 2x_1x_3 + x_1^2 + y_3^2 - 2y_2y_3 + y_1^2 = d_1^2,$$

Observing that $x_1^2 + y_1^2 = v_1^2$,

$$d_0^2 - y_3^2 - 2v_1x_3\cos(\theta_1 - \pi) + v_1^2 + y_3^2 - 2v_1y_3\sin(\theta_1 - \pi) = d_1^2,$$

We calculate x_3 as:

$$x_3 = \frac{d_0^2 + v_1^2 - 2v_1 y_3 \sin(\theta_1 - \pi) - d_1^2}{2v_1 \cos(\theta_1 - \pi)}, \tag{14}$$

Using x_3 in Eq.(8), and substituting Eqs.(12,13) we get:

$$(x_3 - x_2)^2 + (y_3 - y_2)^2 = d_2^2,$$

$$x_3^2 - 2x_2x_3 + x_2^2 + y_3^2 - 2y_2y_3 + y_2^2 = d_2^2,$$

Again observing that $x_2^2 + y_2^2 = v_2^2$,

$$d_0^2 - \frac{d_0^2 + v_1^2 - d_1^2 - 2v_1 y_3 \sin(\theta_1 - \pi)}{v_1 \cos(\theta_1 - \pi)} v_2 \cos\theta_2 + v_2^2 - 2v_2 y_3 \sin\theta_2 = d_2^2, \tag{15}$$

We can now calculate y_3 from Eq.(15). To ease the presentation in Eq.(15), we define B as:

$$B = \frac{v_2 \cos\theta_2}{\cos(\theta_1 - \pi)} 2\sin(\theta_1 - \pi) - 2v_2 \sin\theta_2,$$

if we simplify, B becomes:

$$B = \frac{2v_2 \sin(\theta_1 - (\pi + \theta_2))}{\cos(\theta_1 - \pi)}, \tag{16}$$

Again, to ease the presentation in Eq.(15), we define A as:

$$A = d_2^2 - d_0^2 - v_2^2 + \frac{v_2 \cos\theta_2}{v_1 \cos(\theta_1 - \pi)}(d_0^2 + v_1^2 - d_1^2), \tag{17}$$

where using Eqs.(16, and 17) y_3 becomes:

$$y_3 = \frac{A}{B} \tag{18}$$

The *red* node can localize the *blue* node by solving equations (14) and (18). Next, the *red* and the *blue* nodes switch roles, and the *blue* node moves while the *red* node remains stationary, which is outlined in *Lock-step Movement* section below. After both groups of nodes complete their 2-step motion, they agree on a common north for the entire network, which is described in *Selecting a Common North* section below.

2.2.1 Lock-step movement

For applications where the mobile sensor nodes are required to move as a cohort, we configure our GCDL algorithm to perform directional movement.

We randomly assign *red* and *blue* colors to nodes at each iteration to uniformly color the nodes in the swarm. Once the direction of movement is known by all nodes, each group performs a *zig-zag* movement following the direction, as shown in Figure 5. A zig-zag movement fits with our algorithm's requirement of the 2-step motion, while still allowing the node to move towards a certain direction. We would like to highlight here that the *zig-zag* movement can be performed with different step movement angles. For simplicity of presentation, we use orthogonal movements as shown in Figure 5.

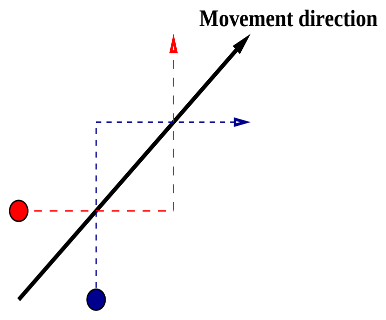


Fig. 5. Zig-zag motion of nodes towards movement direction.

2.2.2 Selecting a common north

In GCDL, nodes coordinate their movement based on a pseudo-north, since no compass is used. As the pseudo-north is selected arbitrarily by each node, and altered by the localization errors throughout the movement, nodes have to agree on a common north with their neighbors in order to move as a cohort. When both group of nodes complete their lock-step motion, neighbor nodes can further communicate to agree on a common north for the entire swarm. In order to achieve this, each node exchanges calculated position information of each of its one-hop neighbors, based on its local coordinate system. Each node, by cross examining its neighbor's position relative to itself ($\angle a$ in Figure 6) and its own position relative to that specific neighbor ($\angle b$), calculates the skew in local coordinate systems ($\angle\alpha$ in Figure 6). Once the skew between local coordinate systems is calculated, the nodes can use two methods in order to agree on a common coordinate system: (1) use a proactive approach and force the entire swarm to agree on a single coordinate system by using a hierarchy (e.g. TAG tree [25]), or (2) use a reactive method and store only the variance (α) for each neighbor, and initiate a local correction each time a direction information is received from neighbors. The use of the proactive or the reactive approach is specific to the characteristic of the network and the application. Both approaches can be used with our GCDL algorithm.

In this section, we presented our localization algorithms. GDL performs localization in single step of the motion through the use of a compass on each node, while GCDL performs localization without the use of a compass, following a 2-step motion.

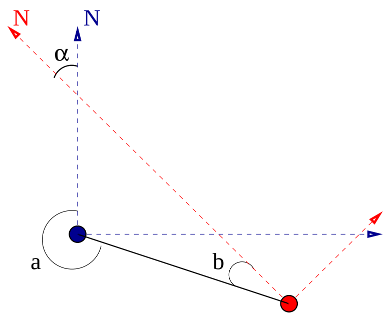


Fig. 6. When the relative positions of two nodes are a and b, the skew in their local coordinate systems is α.

3 Cohort Sensor Network Mobility

Our algorithms are 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 algorithms in a realistic mobility scenario, we adapt a mobility model based on the Reference Point Group Mobility (RPGM) model [18]. Although we considered a range of mobility models [6], we decided to base our analysis on RPGM due to its generality and simplicity. Here, 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 algorithms 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.

The mobility algorithm we use for directed motion is presented in Figure 7. 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 algorithm where nodes attempt to maintain a neighbor distance that is a fraction of their wireless range. The neighbor distance is adjusted dynamically with the number of neighbors so that nodes with fewer than k neighbors stay closer while still moving with the network, which avoids network partitioning especially at the perimeter of the network. Localization of the one-hop neighbors is a pre-requisite for the mobility algorithm to run efficiently. However, the algorithm does not strictly require all neighbors to be localized, it considers

```
MoveNode(Node N, NeighborList NL,
DirectionVector $\vec{D}$, int $k$, RangeFactor RF)
 1: $\vec{V} \leftarrow 0$
 2: $count \leftarrow 0$
 3: for each localized neighbor n in NL do
 4:     /* $\vec{u}_{N,n}$ is the vector from N to n */
 5:     $\vec{V} \leftarrow \vec{V} + \vec{u}_{N,n}$
 6:     $count \leftarrow count + 1$
 7: if $count < k$ then
 8:     $RF \leftarrow$ RF /2
 9: $\vec{V} \leftarrow (RF * range(N) * \vec{V} + \vec{D})/(count + 1)$
10: Move node N by $\vec{V}$
```

Fig. 7. The mobility algorithm for directed motion.

only the localized neighbors and performs the necessary calculations based on these neighbors. The *range*(Node N) function returns the wireless range of node N in terms of distance. When a boundary is reached, a new direction of movement must be chosen. 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, a swarm of mobile sensors may move in a zig-zag pattern, with the goal to discover an oil spill and cover the contaminated area once this spill 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. Our localization algorithms require each node to communicate only with its one-hop neighbors thus avoiding flooding of the network. In this respect, other mobility algorithms can be plugged in to perform various tasks using our localization algorithms.

4 Organizing Sensor Nodes In a Geometric Shape

In this section, we present a method to arrange nodes in a certain geometric shape controlled by a specific coordinator node in the network. In *search-and-rescue* type of scenarios nodes move on unknown territory, and explore the area through the help of their sensors. These can be temperature, noise, infrared, or similar sensors and can be used collectively to help guide the node movement in unknown or hostile territories. As a motivational example, assume that we are interested in a network of mobile sensor nodes with temperature sensors, exploring an unknown territory on a mission to detect the source of a fire. In this scenario, mobile nodes often have to make a decision on which direction they should move to achieve their objective. In order to do so, nodes process the temperature readings obtained from the boundary nodes of the network, and move towards the direction with the highest temperature readings, as shown in Figure 8. We investigate the idea of arranging the nodes in a connected polygonal shape of our choice, so that nodes can gather data more efficiently from diverse fronts of the territory. The size and shape of the polygon depends on the application or the physical properties of the territory. Figure 8(a) shows a sensor network organized in a $4x4$ square, trying to determine the region of the fire. As the figure shows, the square shape is not the best choice for this case compared to the L-shape shown in Figure 8(b). Even though the L-shape uses the same number of nodes as the square shape, it can expand further into both directions increasing the chances of detecting the correct region of the fire.

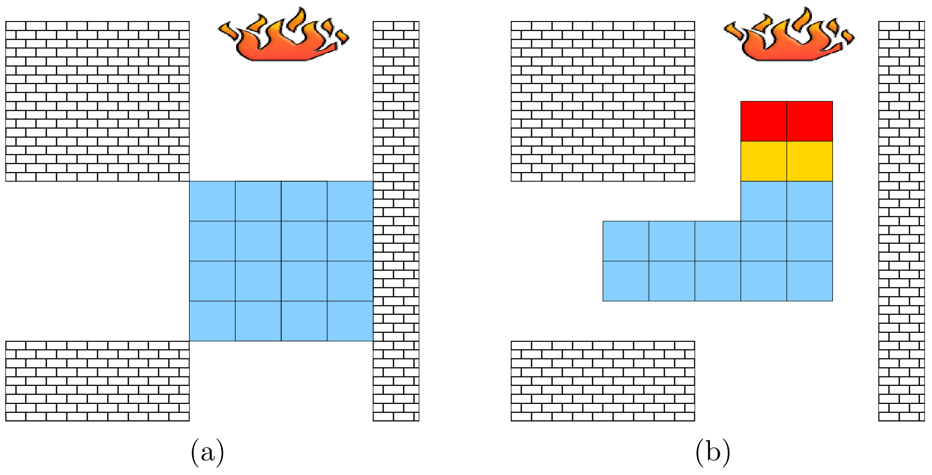


Fig. 8. Two configurations of a mobile sensor network using temperature sensors to locate a fire.

In order to organize nodes in a spanning polygonal form, we use one of our nodes as the coordinator. The coordinator function can be performed by any of the nodes in the swarm, or a specially equipped node depending on the application. Our method does not have particular requirements on the coordinator, other than that it defines the origin point for the polygon and it remains stationary during polygon formation. Nodes accept the coordinator as the origin $(0, 0)$ and the polygon coordinates are defined relative to the coordinator. Whenever there is a requirement for polygonal network formation, the constraints regarding the size and shape of the polygon is broadcast by the coordinator to the rest of the network. The polygon is expressed as a list of point coordinates that define the boundary lines relative to the origin. The

```
FormPolygon(Polygon p)
 1: if Coordinator then
 2:    /*Arrange the nodes in geometric shape*/
 3:    Coordinator stops moving
 4:    Coordinator broadcasts Polygon information
 5:    Coordinator broadcasts its location as the origin
 6: else
 7:    Nodes that receive Polygon info run FillPolygon algorithm
```

```
FillPolygon(Polygon p)
 1: Node calculates its position relative to the Coordinator node
 2: Node re-broadcasts Polygon info and its relative position
 3: if Node is inside the polygon then
 4:    if Node is close to the boundary then
 5:       Reduce the speed of the node
 6:       Force to stay inside the polygon
 7:    else
 8:       Perform regular area coverage as in Fig. 7
 9: else
10:    Find the closest line segment of the polygon
11:    Move to the closest line segment to enter inside the polygon
```

Fig. 9. *FormPolygon* determines the geometric shape of the entire network. *FillPolygon* is used by each node to rearrange its position.

algorithm for arranging the network into polygonal formation is presented in Figure 9(*FormPolygon*). The area of the polygon should be chosen properly based on the number of nodes. Nodes should fill the entire area evenly without segmentation, while still maintaining a multi-hop path to the coordinator node necessary to receive further updates.

Nodes have two fundamental tasks during polygon formation; area coverage within the polygon, and boundary detection. The localization steps essential to determine the boundary of the polygon and the neighbor positions are handled by our available localization algorithms. Using our localization algorithms, each node can measure the relative positions of each of its neighbors. Upon receiving the request for the polygonal network formation, nodes within one-hop distance of the coordinator will update their positions relative to the origin (coordinator), and re-broadcast the request by including these relative positions. Iteratively, each node i receiving the request from its neighbor j receives the polygon information, and the position of node j relative to the origin. Combining this information with the inter-distance of nodes i and j obtained from our localization algorithms, each node i can calculate its own position relative to the origin (coordinator). By the time the request is received by the entire network, each node receives the polygon border coordinates, and can calculate its own position relative to the polygon. Nodes use the ray casting algorithm [38] to test whether they are inside the polygon or not. Nodes use the algorithm presented in Figure 9(*FillPolygon*) in this stage of the polygon formation. Nodes that are outside the polygon border calculate the position of the closest line segment of the polygon and move towards that line segment's coordinate to cross the boundary of the polygon. Once the nodes are inside the polygon area, they do follow the regular area coverage algorithm previously described in Figure 7. There are two exceptions to this algorithm; (1) nodes force themselves to stay inside the polygon, (2) nodes that are close to the boundary of the polygon reduce their movement capability to establish a physical boundary of nodes near the border for the rest of the network. These exceptions reduce the unnecessary back and forth movement of the nodes close to the boundary (especially corners of the polygon) and form a layer around the boundary much like the effect of surface tension in liquids.

5 Experimental Evaluation

In this section, we evaluate our algorithms in three different types of experimental settings. First, in Section 5.1 we evaluate properties specific to our GDL algorithm under various simulated error settings, and show that even significant environment errors do not dominate the performance of our algorithm. Later, in Section 5.2 we evaluate our GCDL algorithm under various errors and observe how the errors affect the accuracy of the localization and common north resolution algorithms. In Section 5.3 we compare GDL and GCDL with an Absolute Positioning algorithm that uses dead reckoning in various random and directed mobility scenarios, and observe the effects of cumulative errors on the localization algorithms. Section 5.4 compares the behaviour of our algorithms with the Absolute Positioning algorithm during coherent movement of the swarm. Finally, in Section 5.5 we evaluate our polygonal network formation method for various polygonal shapes.

5.1 Evaluation of the GDL algorithm

Initially, we present results from the GDL algorithm under ideal conditions, without measurement errors. Subsequently, we introduce independent errors on angle and distance measurements to simulate real world disturbances.

5.1.1 Experiments under ideal conditions

In this experiment we simulate nodes randomly placed in a 100x100 area under a uniform spacial distribution. Each simulation is run for 100 epochs, and the results are averaged. At each epoch, nodes perform a random walk with random speed $[0, 5)$, random angle $[0, 2\pi)$ and fixed radio range of 6. The speed and angle are selected using a uniform random distribution. Node density represents the number of nodes over the total deployment area. GDL requires two neighbors to accurately find each other's positions. Figure 10 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 10, the percent of non-localized nodes approaches zero for densities greater than 0.02. From this graph we can conclude that our algorithm calculates node positions for dense networks, and introduces minor node localization failures, less than as 3%, for sparse networks.

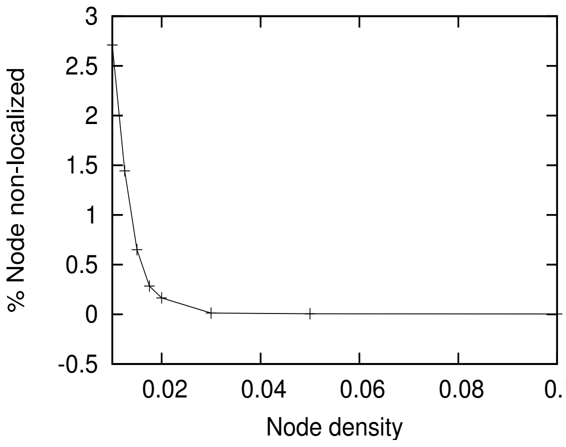


Fig. 10. Percent of non-localized nodes for different node densities.

5.1.2 Introducing measurement errors

We now relax the ideal condition assumption and introduce errors on distance and angle measurements. In the 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. The reason for introducing absolute percent error for angle measures is to simulate large errors that may occur due to mechanical part inaccuracies or failures during rotations. These errors change 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 2, prevent the algorithm from localizing some of the nodes. Figure 11 (a) 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 11 (b). We postulate that 30% noise on angle and distance measurements is a significant error rate for real world conditions. Even so, Figure 11 (b) shows that the portion of non-localized nodes remains below 16% at all times, and drops below 10% for errors lower than 10%. These results indicate that our GDL algorithm provides node localization for the larger proportion of the swarm, even in the face of extreme measurement deviations. We further corroborate this claim by carrying out tests in random and directed movement scenarios in Section 5.3.

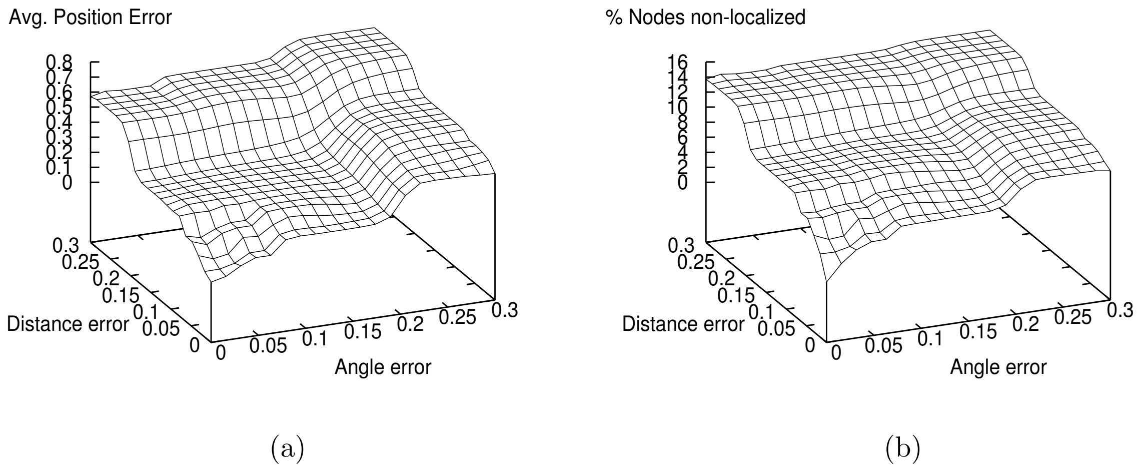


Fig. 11. Effects of angle and distance measurement noise on position error (a), and percent of non-localized nodes (b), for GDL

In order to evaluate the effects of movement speed and wireless range we tested our GDL algorithm under high noise, with a fixed wireless range (10 units) and variable speed values for nodes. [1] We apply an upper limit to the speed, such that the nodes do not move a distance greater than their wireless range per epoch. In any sensor network scenario, if a node moves by a distance greater than its wireless range in one epoch, it is highly probable that its neighborhood will change at each step, which would make it impossible to localize. We can see in Figure 12 that the localization error of our algorithm is nearly constant for increasing speed. The maximum speed supported by our algorithm is the wireless range distance traveled throughout an epoch. In our experiments, this is 10 units of distance traveled per epoch.

5.2 Evaluation of the GCDL algorithm

In this section, we evaluate the effects of various environment errors on the common north resolution method of our GCDL algorithm. As the metric to

[1] As our GCDL algorithm performs a lock-step motion, we exclude GCDL from this movement speed experiment.

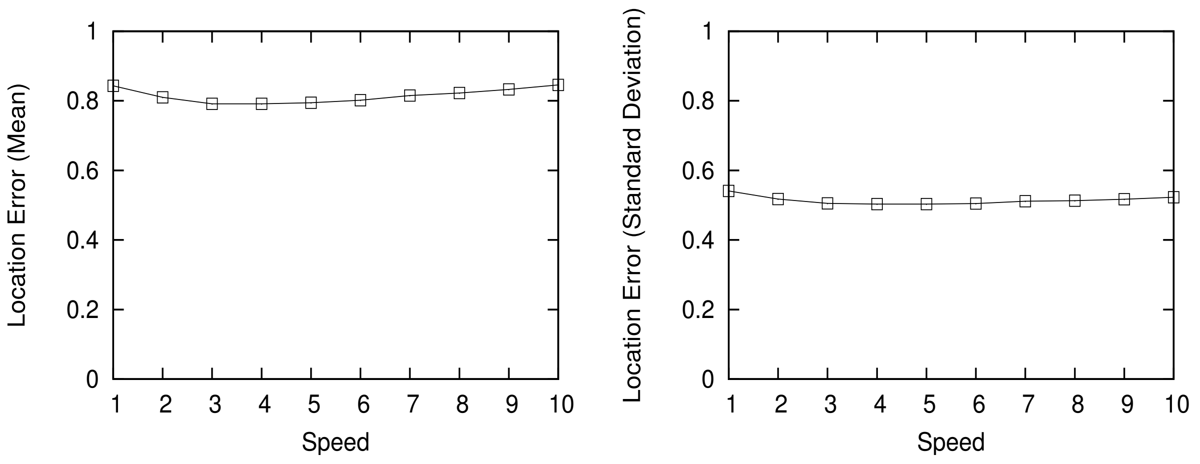
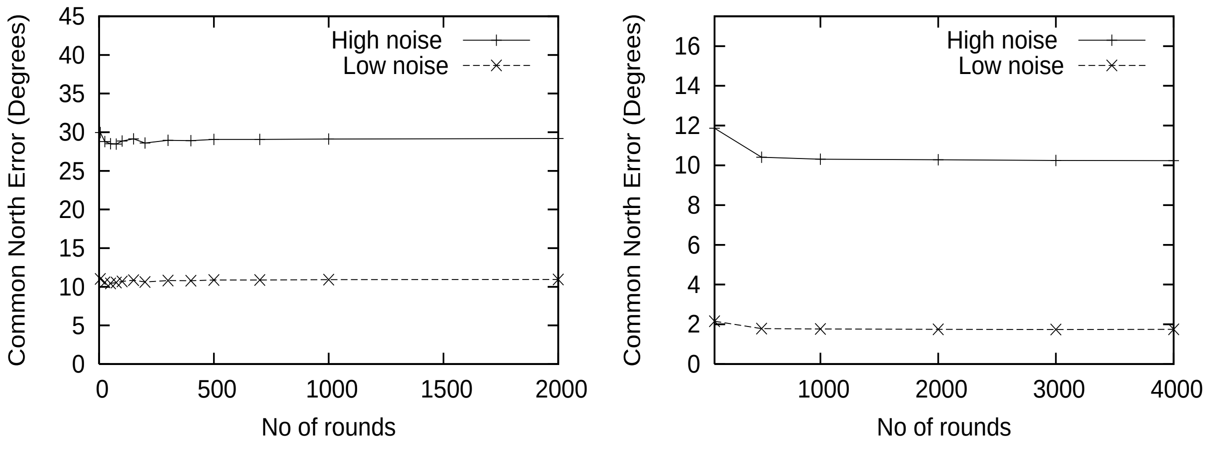


Fig. 12. Mean and standard deviation of position error of GDL vs. speed of nodes with a wireless range of 10 units and speed measured in units per epoch.

measure the common north error of the swarm, we use the average difference between the real and the calculated norths of each neighbor per epoch.

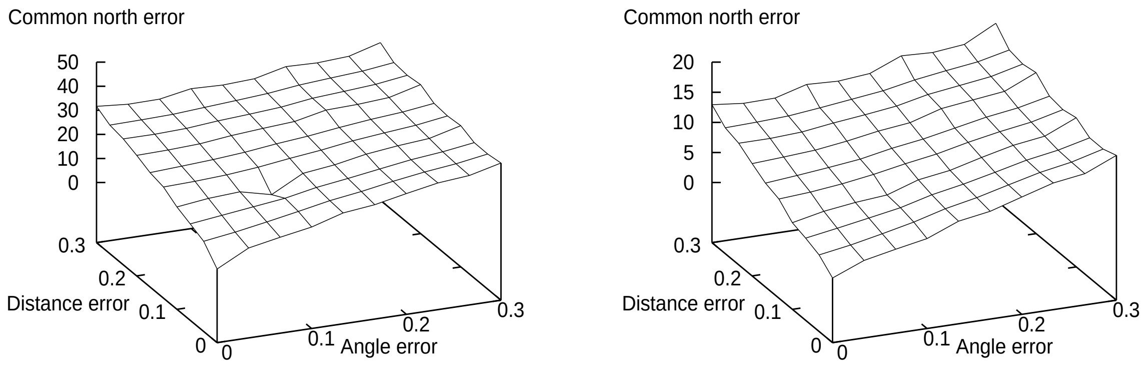
We first evaluate whether the duration of the movement affects the common north error. We simulate 100 nodes in a 100x100 area. In random motion, nodes can cover at most 5 units and have a fixed radio range of 15 units. We assume that in directed motion nodes move at a maximum speed of 3 units per epoch and the radio range is set to 5 units. All experiments are performed 100 times, and the average values are reported. We test our algorithm for two different uniform random noise levels: high and low as well as in random motion and directed motion 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. In this experiment, after each epoch, nodes resolve the common north variance with their neighbors and the average error per node is calculated. Figure 13 shows that for both random and directed movement scenarios, the average error on common north remains constant throughout the total number of epochs. This is expected as we do not store any information other than the current motion per epoch. All calculations are performed from scratch based on the new neighborhood after the 2-step motion. Thus, GCDL is free from incremental errors.

Having established that the common north error is not affected by the number of epochs, we vary the distance and angle measurement noise and see how it affects the common north error in Figure 14. In this figure, we can observe that the amount of noise on the angle and the distance measurements both affect the common north error of the GCDL algorithm. The common north error is zero for environments free of error, and the error increases linearly with the angle and distance errors when they are introduced. For a low noise level, the common north errors are approximately 10° and 2° respectively



(a) (b)

Fig. 13. Effect of number of epochs on selected common north angle, (a) random motion, and (b) directed motion, for GCDL



(a) (b)

Fig. 14. Percent error of angle and distance measurement noise on selected common north error in degrees, (a) random motion, and (b) directed motion, for GCDL.

for random and directed motion. For a high noise level, the errors become approximately 34° and 14°, respectively. The common north errors are quite reasonable considering that we assume $\pm 2\pi/100$ ($\pm 3.6°$) and $\pm 2\pi/10$ ($\pm 36°$) errors on angle measurements caused by node actuators, respectively for low and high noise levels.

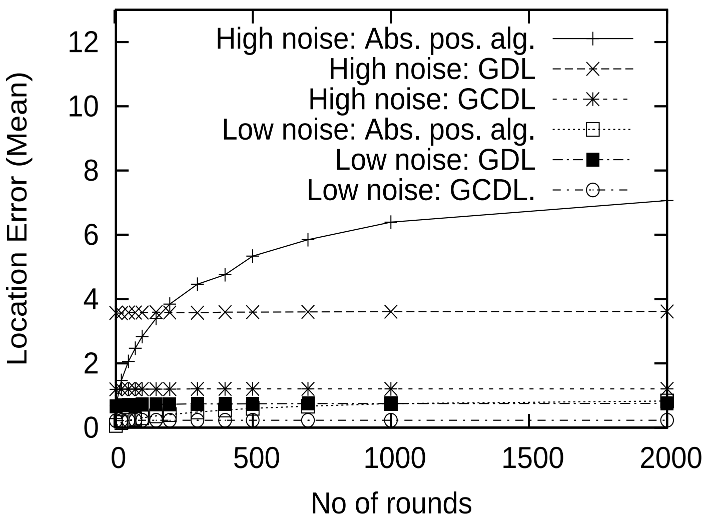
5.3 Comparison with an absolute positioning algorithm

In this section, we compare our algorithms with an Absolute Positioning algorithm. In this algorithm, we assume that nodes know their initial positions in the deployment area, thanks to an anchor point or another type of global positioning infrastructure. We also assume that once nodes receive their initial position, they do not receive any additional positioning information, relative or absolute. To this effect, 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 when nodes are deployed from a known position and asked to explore a possibly remote area where they cannot maintain communication with the anchor at the deployment position.

We simulate two different mobility scenarios. The first is based on random movement, where 100 nodes with fixed radio range of 15 units can cover a distance of at most 5 units per epoch. The second scenario is the directed movement described in Section 3. Nodes sweep the area in a zig-zag manner, with radio range of 5 units and maximum per-epoch distance of 3 units. An example trajectory of the nodes in the directed movement scenario is shown in Figure 20. 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 Figures 15 and 16, we show the average errors for all algorithms over the progression of up to 4,000 epochs, for two different uniform random noise levels: high and low, in random motion (Figure 15) and directed motion (Figure 16) scenarios. Since our algorithms calculate distances within each epoch and do not use any cumulative data, the errors of our algorithms are nearly constant over the number of epochs, for both scenarios. Although the absolute positioning algorithm starts with a low error value, small measurement errors accumulate over each epoch and cause a continuously increasing error. The mean error of the GDL is as much as 2 times and the mean error of the GCDL is as much as 10 times lower than the mean error of the absolute positioning algorithm. The high error values in the absolute positioning algorithm reflect on the effects of cumulative errors and show that it is not a robust solution for high noise scenarios. On the other hand, our localization algorithms provide consistent behavior in the above-mentioned scenarios.

In order to further evaluate the effects of the noise level on the performance of our algorithms, we create three different noise distributions, all based on high noise level. In Figure 17 we present these distributions. Linear distribution is the high noise level already used in previous experiments (See Figs. 15 and 16). The SQRT represents a scenario where the random noise increases with the square root of the distance. Similarly, the SQR represents that the random noise increases with the square of the distance. For example, if a node performs a movement of 5 units in SQR noise distribution, the noise level will be $\pm 30\% * 5^2 = \pm 7.5$, which means the node will move a distance in the range of between -2.5 and 12.5 units. Since we already presented the results of the Linear noise distribution, here we only test our algorithms with the SQRT and SQR noise



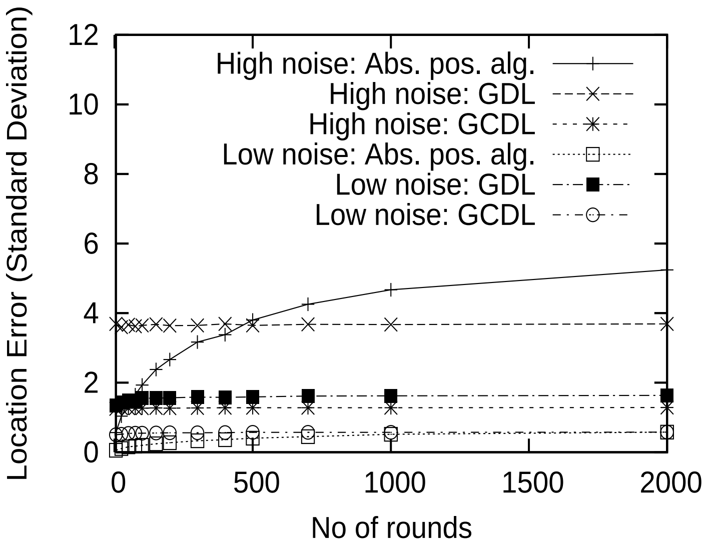


Fig. 15. Mean and standard deviation of position error vs. number of epochs for our algorithms and the absolute positioning algorithm performing random movement.

distributions performing random motion, and present the results in Figure 18. In both noise distributions our algorithms outperform the absolute positioning algorithm. In SQRT noise distribution, the mean error of GDL is as much as 2 times, and the mean error of GCDL is as much as 18 times lower than the mean error of the absolute positioning algorithm. In SQR noise distribution, the mean error of GDL and GCDL are respectively 3.5 and 11 times lower than the mean error of the absolute positioning algorithm. For all algorithms, the errors in SQRT noise distribution is less than the errors in SQR distribution. We also observe that even in SQR noise distribution, the errors of our algorithms are almost constant for increased number of epochs, which again is a feature of the memoryless design of our algorithms. As a cross examination of the behavior of the algorithms in Linear and SQR noise distributions, we observe that although the error of absolute positioning algorithm increases by a factor

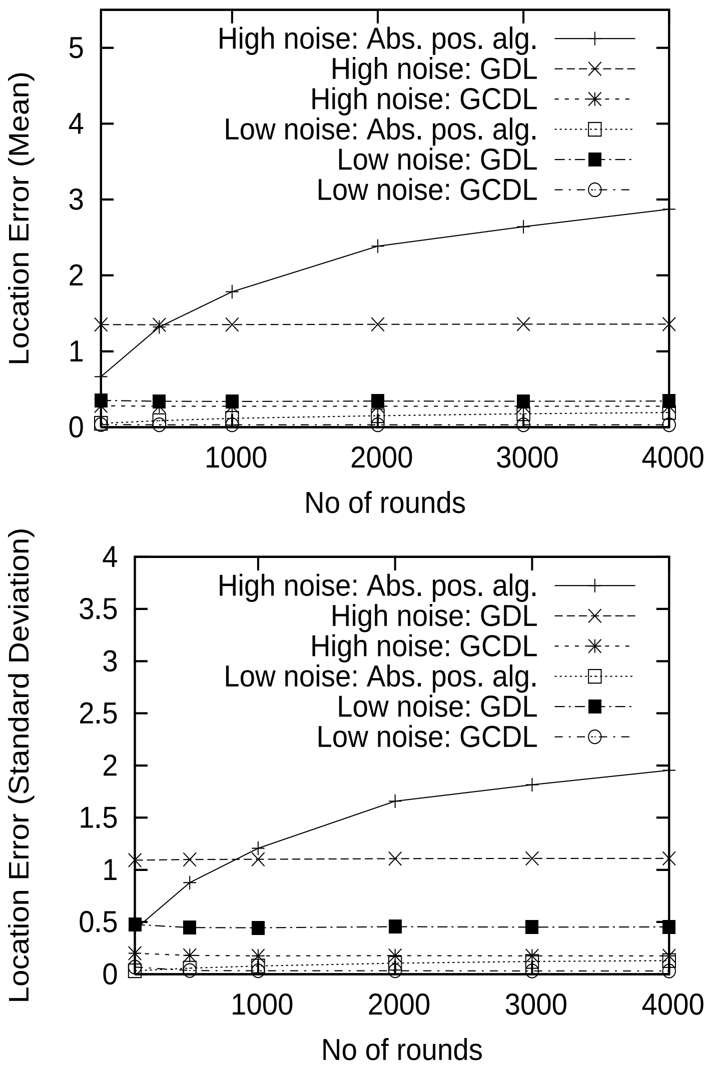


Fig. 16. Mean and standard deviation of position error vs. number of epochs for our algorithms and the absolute positioning algorithm performing directed movement.

of 3, the errors of our algorithms increase by a factor of less than 2. We can conclude that our algorithms are more immune to noise than the absolute positioning algorithm.

Figure 19 presents the average CPU time each node spent calculating Equations (4) and (6) in GDL and Equations (14) and (18) in GCDL, for both random and directed mobility scenarios. Since we do not have access to real sensor hardware, we ran the experiments on a Pentium IV 3 GHz machine and present the average time spent per localization in order to compare the performance of our algorithms. As shown in the figure, the CPU overhead is less for GCDL compared to GDL, which is a direct result of the number of calculations required by each algorithm.

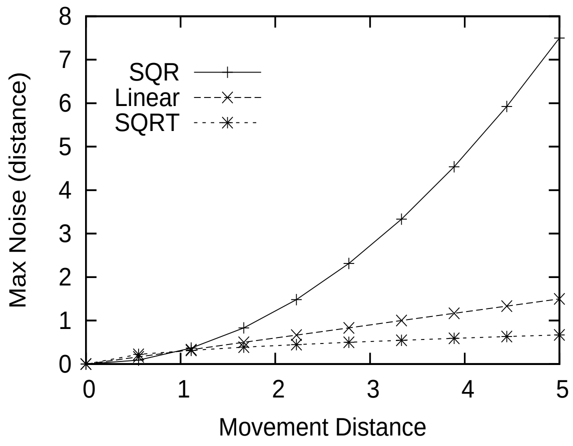


Fig. 17. Various distributions of high noise levels used in our experiments.

5.4 Observing cohort movement of mobile nodes

In Figures 21, 22 and 23 we present the snapshots of the simulations of the absolute positioning algorithm, our GDL and GCDL algorithms, respectively, performing a zig-zag directed movement (as in Figure 20) under high noise. All simulations use the same movement algorithm as described in Section 3. Because of the cumulative errors, the absolute positioning algorithm is not capable of maintaining the topology of the network and becomes disconnected (Figure 21). On the other hand, our algorithms maintain connectivity at all times while forming a nice semi-rigid topology, which complies with one of our main goals to enable coherent movement of nodes as a swarm even under high noise. The swarm organization for GCDL algorithm is more "packed" than its GDL counterpart. This behavior is caused by the separation of nodes into two groups, red and blue. Nodes can only localize their neighbors in the opposite group, and not the ones in their own group. The movement algorithm instructs the nodes to keep a specific distance between localized neighbors. This works for the opposite group nodes, since they are localized, but the nodes in the same group cannot localize each other and for this reason they might stay close to each other. Even in this case, GCDL succeeds to maintain the connectivity and semi-rigid topology of the swarm. The results in Figures 15, 16, 22, and 23 also support our claim that even under high noise settings, environmental errors do not deteriorate our algorithm's behavior; effects of these errors are constant through epochs. As seen in Figure 22, occasionally a few nodes (two in this case) disconnect from the network. This happens when nodes near the network's perimeter cannot be localized. Although the number of these nodes is small, they can be further controlled by forcing stricter k-neighborhood rules in the mobility algorithm.

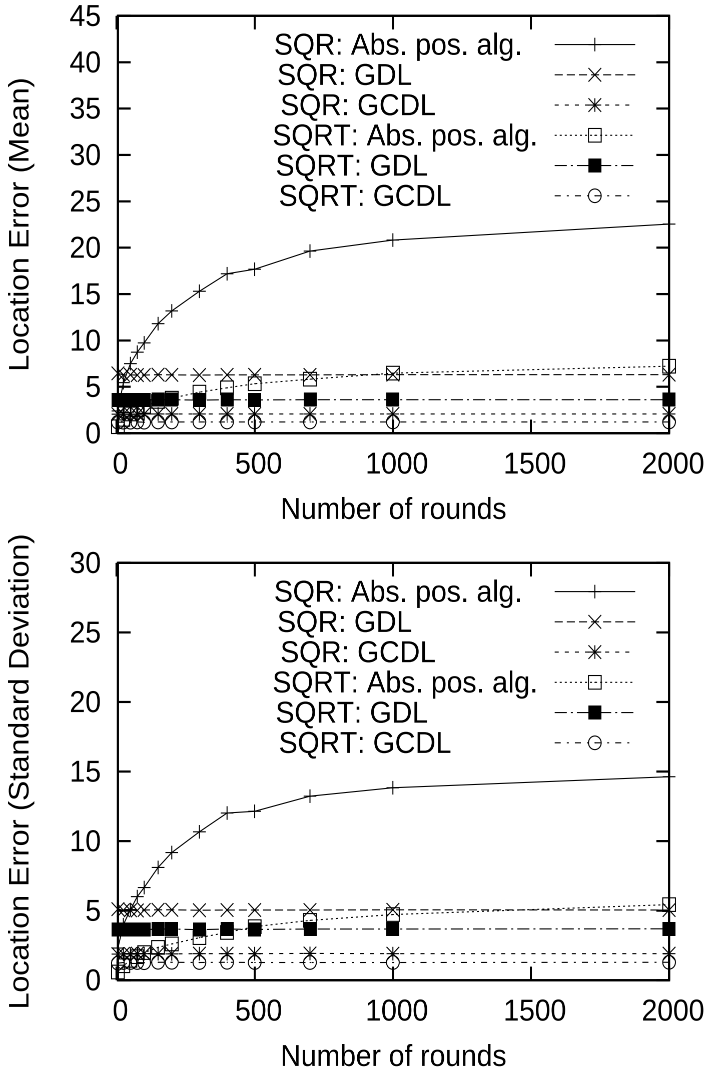


Fig. 18. Mean and standard deviation of position error vs. number of epochs for our algorithms and the absolute positioning algorithm performing random movement.

Movement	GDL	GCDL
Random	1.95 ms	1.08 ms
Directed	2.06 ms	1.12 ms

Fig. 19. Average time in milliseconds (ms) spent for localization calculations in core GDL and GCDL algorithms.

5.5 Forming geometric shapes

In previous sections, we observe in general that if area coverage and cohort movement algorithms are applied, sensor network acquires an oval or circular shape. As we presented in Section 4, there might be cases which require the

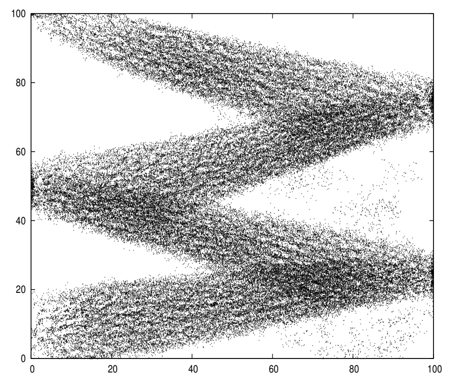


Fig. 20. Directed trajectory of nodes performing zig-zag movement.

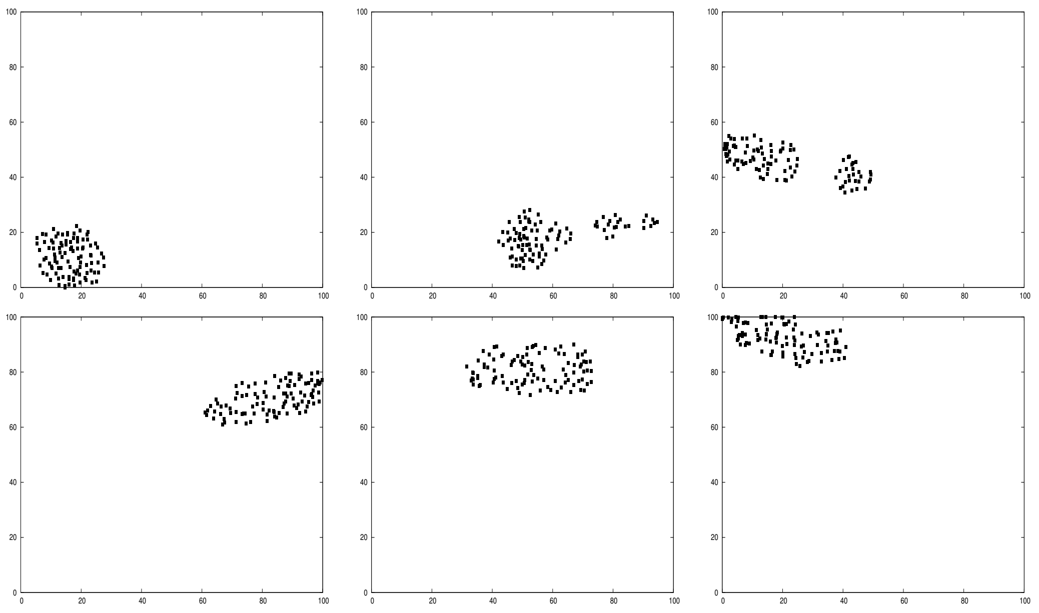


Fig. 21. Snapshots of absolute positioning algorithm performing directed motion in Figure 20.

sensor network to be organized in a predefined geometric shape. In this section we apply our algorithms presented in Section 4 for various geometric shapes and present the results. Figure 24 shows a sensor network of 100 nodes organizing themselves in various geometric shapes (triangle, rectangle, octagon and L-shape). The red node in the middle is the coordinator node, and the circle around the coordinator node represents the wireless range of each sensor. The blue nodes assume they are on the inside and the purple nodes are on the outside of the geometric shape. The top row shows the sensor locations when there are no measurement errors. The nodes can organize themselves quite well. With some exceptions in the corner cases the shapes are quite accurate. We can see the surface tension effect near the borders (especially in the triangle shape), and how this tension effect helps the inner nodes perform area coverage. The middle row shows a simulation in low noise scenario. Again in this low noise environment, nodes can organize themselves quite well. Com-

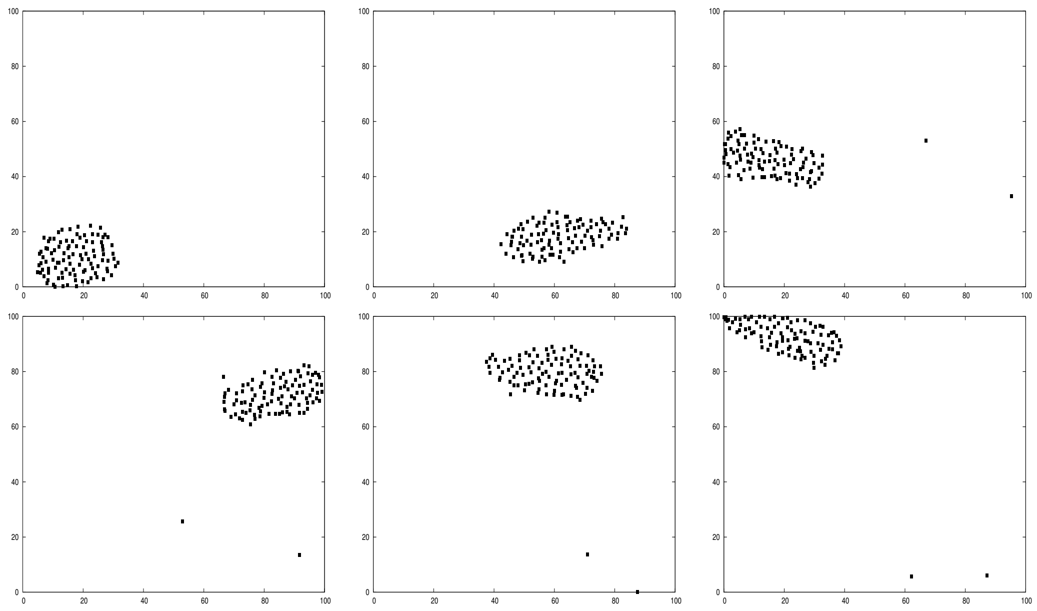


Fig. 22. Snapshots of GDL algorithm performing directed motion in Figure 20.

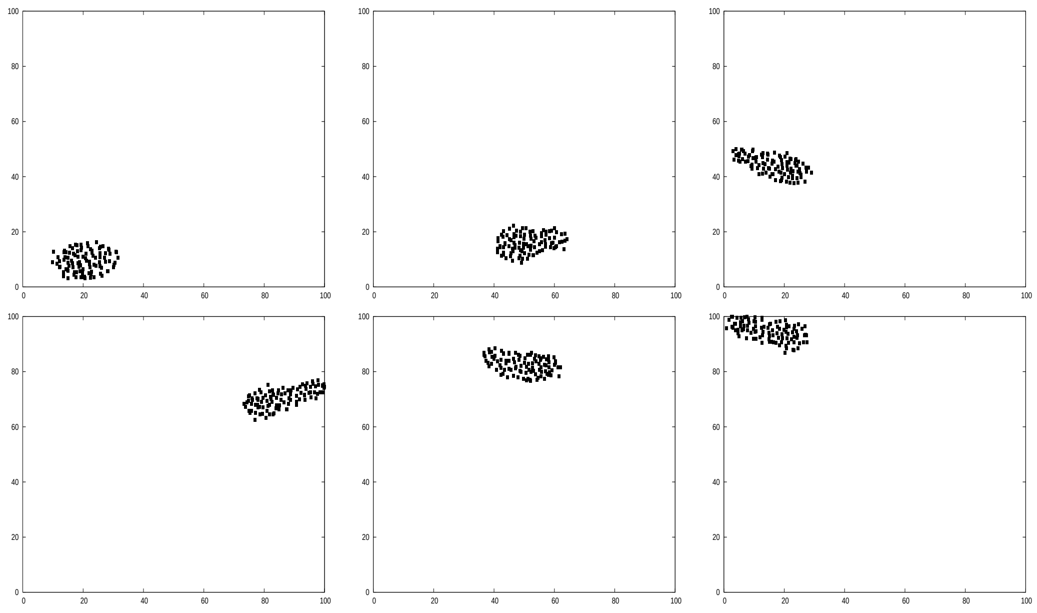


Fig. 23. Snapshots of the GCDL algorithm performing directed motion in Figure 20.

pared to the ideal case without errors, the shapes have some deformations, and some miss-located nodes. Nevertheless, the observed shapes visually resemble well the expected shapes. The bottom row shows a simulation in a high noise scenario in which the error level is too high in order for the nodes to be able to precisely organize themselves in a geometric shape. The nodes cannot form the border lines in this case because of the large errors, and once the borders are not established, the rest of the nodes cannot organize themselves. Even in this case resemblances to the original shape in triangle and partly in L-shape can be observed.

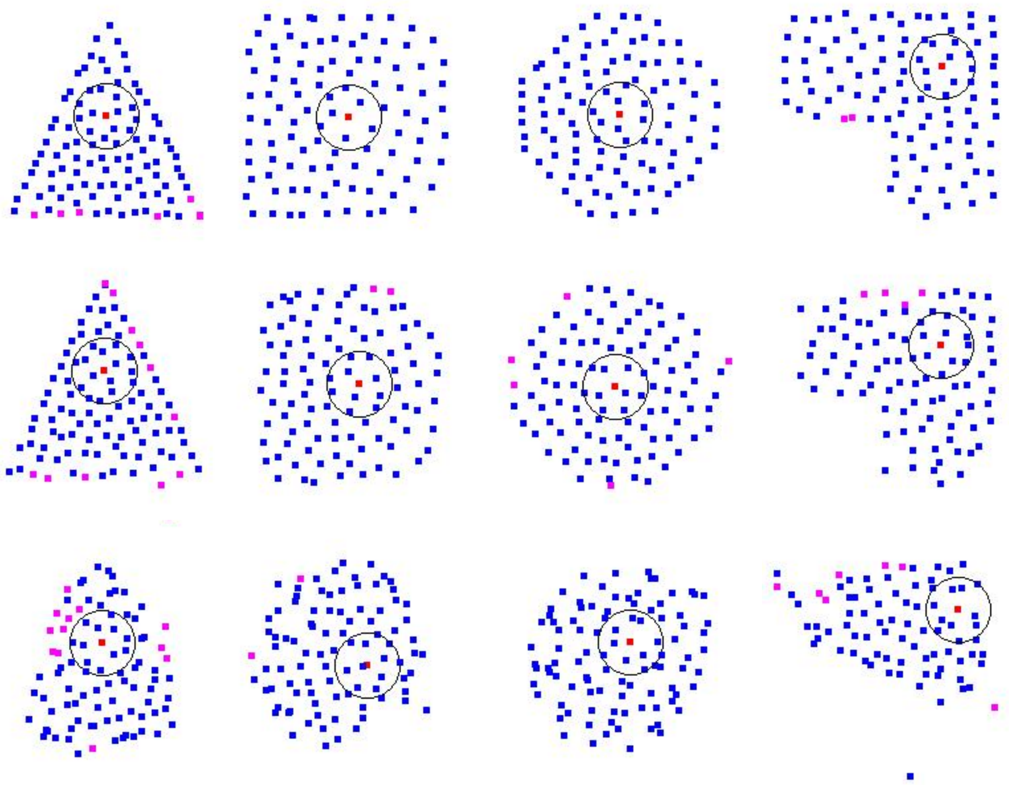


Fig. 24. Snapshots of geometric shape formation. Top row: without noise. Middle row: under low noise. Bottom row: under high noise.

6 Related Work

Hightower *et al.* [16] has a survey on location systems. Early research on sensor localization problems have primarily focused on static sensor networks [10, 13, 15, 20, 26–28, 32, 35–37]. Recently, however, more attention has been given 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 [7].

Low precision for close range and limited coverage (especially indoors) of GPS systems led to the investigation of GPS-free localization for mobile nodes. One common technique used is to exploit wireless communication. Bulusu *et al.* [4] use known reference points to send periodic beacon messages. By receiving beacons from these reference points, nodes can localize themselves. The accuracy of the localization depends on the distance to the reference points. Priyantha *et al.* [32] 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 [41]. For static nodes, and only using Euclidean distances, Bădoiu *et al.* [2] propose a constant factor, quasipolynomial-time approximation algorithm. The algorithm requires complete graph information, which results in substantial communication overheads [2] in mobile wireless networks. Priyantha *et al.* [33] propose assisting the static network in distance measurements with a mobile node to achieve the global rigidity required for accurate localization.

In [7], *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, because the relative coordinate system used does not map to the real node positions.

Infrastructure free localization is also pursued in the robotics. Kurazume *et al.* [23] uses a lock-step motion similar to the one used in our GCDL algorithm, but assumes the initial locations of the robots are known, and robots can measure angles between each other, which are extra constraints that we are able to avoid by introducing the zig-zag motion. Franceschini *et al.* [11] and Langendoen *et al.* [24] have side by side comparisons of some of the existing localization algorithms.

In [19], 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 prior 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. Rudafshani *et al.* [34] again use Monte Carlo method in both mobile and static sensor networks to achieve localization. A method based on predictions is presented in [22], 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. Dead reckoning is also used in [42] to localize nodes in sparse mobile sensor networks. Xu *et al.* [40] use accelerometers to detect mobility in wireless sensor networks and efficiently update the neighborhood information based on the mobility. Since no directional localization is used, the method is not suitable for group mobility.

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 [3], 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 [3] can complement our work to improve the neighbor distance measurement error for high speed sensors. Distance measurement methods are surveyed in [5, 30]. The Time of Arrival (TOA) method finds the distance between a transmitter and a receiver through the use of one way propagation time. Time Difference of Arrival (TDOA) is another method to estimate the distance [14]. The TDOA method uses RF and ultrasound signals to estimate the distance accurately, at the expense of additional ultrasound transmitters and receivers. Ultra-Wideband radios enable accurate ranging for sensor networks. Gezici *et al.* [12] surveys the recent advances in the standardization of Ultra-Wideband radios, and possible applications to sensor localization. Recently, Park *et al.* [29] combined the ranging features of Ultra-Wideband radios with hyperbola fitting to perform neighbor localization without using any infrastructure.

7 Conclusions

We propose two distributed algorithms to address the directional node localization problem in wireless sensor networks: GDL where each node has a digital compass, and GCDL which does not require a compass. Both of our directional localization algorithms enable nodes to coordinate their movement relative to their one-hop neighbors, and maintain a semi-rigid structure that results in a coherent movement of the swarm. During mobility, measurement errors or real world disturbances tend to accumulate over time and affect the structure of the swarm, eventually disorganizing it. To avoid this undesirable effect, our algorithms perform localization in a few epochs of the node movement and work only with the data gathered within that time frame. We design our algorithms to work with local knowledge only, without the use of any global positioning infrastructure such as GPS, anchor points, and seed nodes.

We also propose a method to dynamically organize mobile sensor nodes into various polygonal network formations in order to guide their movement especially in *search-and-rescue* type of missions. We tested our algorithms with various simulated real-world errors, in both random and directional mobility scenarios and observed that they perform coherent movement even in high noise settings. In addition, we have compared our algorithms with an absolute positioning approach and observed that our techniques are superior both in terms of localization errors and in maintaining coherent swarm movement in diverse mobility scenarios.

8 Acknowledgments

A preliminary version of this work appeared in the Proceedings of the 5th ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'06), in June 2006 [1]. This work is partially supported by NSF CAREER Grant CCR-0133599 and the University of Athens Research Foundation.

The authors also wish to thank Leonidas Guibas of Stanford University for the insightful discussion leading to 2-step motion, and Goce Trajcevski of Northwestern University for his comments and suggestions.

References

[1] H. Akcan, V. Kriakov, H. Brönnimann, and A. Delis. GPS-Free node localization in mobile wireless sensor networks. In *Proceedings of the 5th ACM International Workshop on Data Engineering for Wireless and Mobile Access (MobiDE'06)*, pages 35–42, Chicago, Illinois, USA, June 2006.

[2] 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, Brooklyn, New York, USA, 2004.

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

[4] 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.

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

[6] 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.

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

[8] K. Chintalapudi, R. Govindan, G. Sukhatme, and A. Dhariwal. Ad-Hoc Localization Using Ranging and Sectoring. In *INFOCOM*, pages 2662 – 2672, Hong Kong, China, 2004.

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

[10] L. Doherty, K. Pister, and L. El Ghaoui. Convex position estimation in wireless sensor networks. In *INFOCOM*, pages 1655–1663, Anchorage, AK, USA, 2001.

[11] F. Franceschini, M. Galetto, D. Maisano, and L. Mastrogiacomo. A review of localization algorithms for distributed wireless sensor networks in manufacturing. *International Journal of Computer Integrated Manufacturing*, pages 1–19, June 2007.

[12] S. Gezici, Z. Tian, G. Giannakis, H. Kobayashi, A. Molisch, H. Poor, and Z. Sahinoglu. Localization via ultra-wideband radios: a look at positioning aspects for future sensor networks. *Signal Processing Magazine, IEEE*, 22(4):70–84, July 2005.

[13] G. Giorgetti, S. K. S. Gupta, and G. Manes. Wireless localization using self-organizing maps. In *Proceedings of the 6th International Conference on Information Processing In Sensor Networks (IPSN'07)*, pages 293–302, Cambridge, Massachusetts, USA, April 2007.

[14] L. Girod and D. Estrin. Robust range estimation using acoustic and multimodal sensing. In *IEEE/RSI Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1312–1320, Maui, HI, USA, 2001.

[15] T. He, C. Huang, B. M. Blum, J. A. Stankovic, and T. Abdelzaher. Range-free localization schemes for large scale sensor networks. In *Proceedings of the 9th annual International Conference on Mobile Computing and Networking (MobiCom'03)*, pages 81–95, San Diego, CA, USA, September 2003.

[16] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, 2001.

[17] 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, Cambridge, MA, USA, 2000.

[18] X. Hong, M. Gerla, G. Pei, and C.-C. Chiang. A group mobility model for ad hoc wireless networks. In *Proceedings of the International Workshop on Modeling Analysis and Simulation of Wireless and Mobile System (MSWiM'99)*, pages 53–60, Seattle, WA, USA, 1999.

[19] L. Hu and D. Evans. Localization for mobile sensor networks. In *Proceedings of the International Conference on Mobile Computing and Networking (MOBICOM'04)*, pages 45–57, Philadelphia, PA, USA, 2004.

[20] R. Iyengar and B. Sikdar. Scalable and distributed GPS free positioning for sensor networks. In *Proceedings of the IEEE International Conference on Communications (ICC'03)*, pages 338–342, Anchorage, AK, USA, May 2003.

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

[22] 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.

[23] R. Kurazume, S. Nagata, and S. Hirose. Cooperative positioning with multiple robots. In *IEEE International Conference on Robotics and Automation*, pages 1250–1257 vol.2, San Diego, CA, USA, May 1994.

[24] K. Langendoen and N. Reijers. Distributed localization in wireless sensor networks: a quantitative comparison. *Computer Networks*, 43(4):499–518, 2003.

[25] S. Madden, M. J. Franklin, J. M. Hellerstein, and W. Hong. Tag: A tiny aggregation service for ad-hoc sensor networks. In *Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI)*, Boston, MA, USA, 2002.

[26] D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys'04)*, pages 50–61, Baltimore, MD, USA, 2004.

[27] R. L. Moses, D. Krishnamurthy, and R. M. Patterson. A self-localization method for wireless sensor networks. *EURASIP Journal on Applied Signal Processing*, 2003(4):348–358, 2003.

[28] R. Nagpal, H. E. Shrobe, and J. Bachrach. Organizing a global coordinate system from local information on an ad hoc sensor network. In *Proceedings of the 2nd International Conference on Information Processing In Sensor Networks (IPSN'03)*, pages 333–348, Palo Alto, California, USA, 2003.

[29] J. Park, E. D. Demaine, and S. J. Teller. Moving-baseline localization. In *Proceedings of the 7th International Conference on Information Processing In Sensor Networks (IPSN'08)*, pages 15–26, St. Louis, Missouri, USA, April 2008.

[30] N. Patwari, J. Ash, S. Kyperountas, I. Hero, A.O., R. Moses, and N. Correal. Locating the nodes: cooperative localization in wireless sensor networks. *Signal Processing Magazine, IEEE*, 22(4):54–69, July 2005.

[31] S. Poduri and G. S. Sukhatme. Constrained Coverage for Mobile Sensor Networks. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 165 – 171, New Orleans, LA, USA, 2004.

[32] N. B. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller. Anchor-free distributed localization in sensor networks. In *Proceedings of the First International Conference on Embedded Networked Sensor Systems (SenSys'03)*, pages 340–341, Los Angeles, CA, USA, 2003.

[33] N. B. Priyantha, H. Balakrishnan, E. D. Demaine, and S. J. Teller. Mobile-assisted localization in wireless sensor networks. In *INFOCOM*, pages 172–183, Miami, FL, USA, March 2005.

[34] M. Rudafshani and S. Datta. Localization in wireless sensor networks. In *Proceedings of the 6th International Conference on Information Processing In Sensor Networks (IPSN'07)*, pages 51–60, Cambridge, Massachusetts, USA, April 2007.

[35] C. Savarese, J. M. Rabaey, and K. Langendoen. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In *USENIX*, pages 317–327, Monterey, CA, USA, 2002.

[36] A. Savvides, C.-C. Han, and M. B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Proceedings of the 7th annual International Conference on Mobile Computing and Networking (MobiCom'01)*, pages 166–179, New York, NY, USA, 2001. ACM Press.

[37] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz. Localization from mere connectivity. In *Proceedings of the 4th ACM International Symposium on Mobile Ad Hoc Networking & Computing (MobiHoc'03)*, pages 201–212, Annapolis, Maryland, USA, 2003.

[38] I. E. Sutherland, R. F. Sproull, and R. A. Schumacker. A characterization of ten hidden-surface algorithms. *ACM Comput. Surv.*, 6(1):1–55, 1974.

[39] G. Trajcevski, P. Scheuermann, and H. Brönnimann. Mission-critical management of mobile sensors: or, how to guide a flock of sensors. In *Proceedings of the First International Workshop on Data Management for Sensor Networks (DMSN'04)*, pages 111–118, Toronto, Canada, 2004.

[40] Y. Xu, Y. Ouyang, Z. Le, J. Ford, and F. Makedon. Mobile anchor-free localization for wireless sensor networks. In *Proceedings of the Third IEEE International Conference on Distributed Computing in Sensor Systems (DCOSS'07)*, pages 96–109, Santa Fe, NM, USA, June 2007.

[41] Y. Yemini. Some theoretical aspects of position-location problems. In *20th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1–8, San Juan, Puerto Rico, 1979.

[42] P. Zhang and M. Martonosi. Locale: Collaborative localization estimation for sparse mobile sensor networks. In *IPSN '08: Proceedings of the 7th International Conference on Information Processing In Sensor Networks*, pages 195–206, St. Louis, Missouri, USA, April 2008.

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