2011 50th IEEE Conference on Decision and Control and
European Control Conference (CDC-ECC)
Orlando, FL, USA, December 12-15, 2011

Distributed Unmanned Ground Vehicle Navigation in Coordinate-Free and Localization-Free Wireless Sensor and Actuator Networks

Guyu Zhang, Christian A. Duncan, Jinko Kanno and Rastko R. Selmic*

***Abstract*—We present a distributed algorithm for the navigation of an Unmanned Ground Vehicle (UGV) towards a set of identified target nodes in coordinate-free and localization-free wireless sensor and actuator networks. The navigation algorithm proceeds in two phases: first, a node level is determined based on a hop distance from the target nodes, which is accomplished by the network nodes without the need for any UGV action; and second, the UGV uses potential fields created by the network actuators to move towards the target nodes, requiring cooperation between certain actuator nodes and the UGV. The hop distance to the target nodes is used to control the main moving direction while the potential field, which can be measured by listeners on the UGV, is used to determine the UGV's movement. The major contribution of this paper is that the algorithm is fully distributed compared to the existing results in the field, which is suitable for real-time implementation. Meanwhile, the presenting algorithm uses three actuators to generate a potential field, which makes the algorithm more robust and flexible. A study on the communication complexity of the algorithm is presented as well as simulation examples that verify the presented algorithm.**

I. Introduction

A Wireless Sensor and Actuator Network (WSAN) is a distributed, self-organized system that consists of sensors and actuators that are connected over wireless communication links. Typical WSANs can be used in applications such as home automation, intelligent traffic control, and cyber-physical systems, which may have different quality of service (QoS) requirements. Usually, these requirements are specified as connection reliability, time varying delay and packet loss. Applications where a fully covered sensing domain is preferred, such as in habitat monitoring and military surveillance, also measure QoS based on the area coverage of the WSAN.

Unmanned Ground Vehicles (UGVs) are autonomous mobile robotic platforms that can be employed in a remote and inaccessible environment. Navigation of UGVs is a challenging problem, especially when expensive and energy consuming localization modules such as GPS are not available. Predefined maps or landmarks are usually used in UGV navigation. However, this information is not always available since WSANs are usually deployed in remote and coordinate-free areas. Moreover, these kinds of offline navigation methods use prior data, making the UGV unable to adapt readily to the dynamic changes of the environment.

In this paper, we explore the problem of distributed, coordinate-free, and localization-free UGV navigation in the WSAN covered area. To make our approach applicable to as many areas as possible, we are not focusing on any specific sensing application, but instead study WSANs using equivalent communication graphs. The underlying principle of interaction between the UGV and WSAN is that the WSAN serves as a medium to guide the UGV. As in [1], for simplicity, we treat the network as an unweighted, undirected communication graph, where two nodes in the network are connected by an edge if and only if they can communicate directly. Hole boundary nodes are located on the boundary of the coverage hole. Buchart and Yao *et al.* [2], [3] propose centralized and distributed algorithms to identify the coverage hole boundary nodes, respectively. We consider the problem of a UGV navigation to the hole boundary nodes, which is triggered after the detection and identification of hole boundary nodes [2], [1] and which occurs before the patching of the coverage holes [3]. Since we treat the hole boundary nodes simply as target destinations for the purpose of UGV navigation, this problem can be generalized to any navigation of a UGV to a pre-determined set of target nodes within a WSAN.

*Authors are with Louisiana Tech University, Ruston, Louisiana 71272, USA.
Contact info: rselmic@latech.edu

II. Related Work

Batalin *et al.* [4] propose a localization-free navigation method that proceeds in two phases. In the first phase, each node calculates transition probabilities to determine the optimal navigation direction. In the second phase, a more reliable and accurate signal strength based method is employed to drive the robot. As do we, Li *et al.* [5] use a hop-distance metric based on the minimum number of hops as a measure of a node's distance from a given target. In [6], Chen *et al.* propose a localized Delaunay triangulation based, distributed guiding navigation protocol that allows for multiple paths and multiple events in the network. Fu *et al.* [7] use a wireless sensor network for indoor robot navigation, employing prior knowledge of sensor positions to localize a robot's position and orientation by acquiring the information of pre-set radio emission sensors. Chen *et al.* [8] propose a set of distributed algorithms for in-network planning where sensor nodes whose coordinates are known serve as landmarks for the navigation. The algorithms ensure that each source node has at least one safe route to the destination, which can be dynamically changed.

While all the cited works use one sensor/actuator node as the navigation beacon, the presenting algorithm uses three actuators to generate a potential field to guide the navigation.

III. Problem Formulation

To model our system, we make certain simplifying assumptions on the capabilities of the WSAN and the UGV:

1. Nodes in the network are identical with regard to both communication and actuation capabilities. Each node is capable of producing an actuating signal with an amplitude a at up to three distinct frequencies f_k for $k \in \{1, 2, 3\}$;
2. The WSAN is arbitrarily deployed in an obstacle-free environment and sensor nodes are stationary after the deployment;
3. Communication between nodes is symmetric, uniform and constant where two nodes can always communicate if and only if they are within distance r_c;
4. The actuating model is omni-directional with actuation range $r_a \geq 2r_c$;
5. The UGV has sufficient control to move in a given direction, i.e., the UGV is a point mass as in [4] without any kinematic dynamics.
6. The UGV can communicate with sensor nodes within distance r_c and is equipped with a set L of listener devices capable of detecting actuator signals at frequencies f_k within distance r_a;
7. The target node(s) are identified before the start of the navigation algorithm at time t_0.

At any instant in our algorithm, a subset of the nodes in the network can be transmitting an actuation signal at a given frequency. For time t, let S_k^t, for $k \in \{1, 2, 3\}$, be the set of nodes currently transmitting at frequency f_k. and let $S^t = \cup S_k^t$ be the set of all nodes currently actuating. From our assumptions, each node $i \in S_k^t$ can generate a radially symmetric potential field U_{ik} at frequency f_k. For each node $i \in S_k^t$, the potential field at each listener $j \in L$ on the UGV, is given by

$$U_{ijk}^t = a \cdot e_{ij}, \tag{1}$$

where e_{ij} is the signal strength that listener j gets from node i. Signal strength e_{ij} is inversely proportional to the path loss $(d_{ij})^m$ [9], where d_{ij} is the distance from node i to listener j and m is the path loss coefficient, usually $m \geq 1$. For simplicity, we assume $e_{ij} = (d_{ij})^{-m}$. The combined potential field at listener j for frequency f_k is given by

$$U_{jk}^t = \sum_{i \in S_k^t} U_{ijk}^t = a \sum_{i \in S_k^t} (d_{ij})^{-m} \tag{2}$$

From this real potential field, the UGV constructs an artificial potential field at each listener, which it uses to navigate the network, given by $U_j^t = \sum_k 1/U_{jk}^t$. In our current algorithm, we ensure that at most one node is transmitting at a specific frequency at any given time. Thus, U_j^t simplifies to

$$U_j^t = \sum_k \frac{1}{U_{jk}^t} = \frac{1}{a} \sum_{i \in S^t} (d_{ij})^m. \tag{3}$$

We place the listener devices on the UGV such that all (but one) listeners $j \in L$ are equally spaced at angles θ_j on a circle centered around the remaining listener 0, as shown in Fig. 1. We designate θ_0 specially to indicate the center of the circle. (For Cricket [10] systems, the precision of distance measurement between nodes is 1-3 cm.) The number of listeners and the radius ρ can be adjusted based on the accuracy requirement for the control of the UGV. At time t, the UGV determines its new relative moving direction θ^t by finding the local minimum value among the potential fields given by

$$\theta^t = \theta_j, \text{where } U_j^t = \min_{j \in L} U_j^t \tag{4}$$

If $\theta^t = \theta_0$, then the UGV is assumed to have reached a local minimum position.

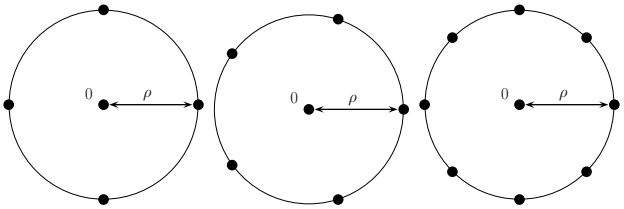

Fig. 1. Arrays of listeners on the UGV.

The UGV moves step by step with predefined step size. The step size is a tradeoff between accuracy and energy consumption. In our simulation tests, we use the step size that equals to ρ. For larger step size, the accuracy will be lower, and the UGV will oscillate around the local minimum of the potential field. However, in a sparse network where sensor's communication radius is much larger than ρ, it is not energy efficient to make the small step size. Algorithms with adjustable step size will be topics of future work.

IV. Navigation Algorithms

There are two sub-algorithms: a level assignment algorithm and a UGV control algorithm.

A. Level Assignment Algorithm

The control algorithm requires that each node has a graph theoretic notion of its distance to the target node(s), called the *hop distance*. A hop is simply a communication link from one node to another. Thus, the hop distance between two nodes is equivalent to the smallest number of edges in all paths in the communication graph between them.

We determine this value incrementally. For each node, let the *level assignment* l represent the known shortest hop distance from the node to any target node. Initially, l is 0 for all target nodes and infinite for all other nodes. As the algorithm proceeds, a node adjusts its level assignment whenever it receives a message indicating a shorter hop distance, subsequently transmitting its revised level assignment to all of its neighbors. This process is similar to a distributed shortest path problem from target nodes to all other nodes, where in this case the weight of each edge equals one. Many distributed shortest-path finding methods already exist [11], [12], [13], but they focus more on providing algorithms to handle changes in network topology. In this paper we do not consider link or node failures since we assume that faulty nodes are the reason for possible holes in the network coverage and they have already been detected and identified. In the following section, we will discuss that the assumption has little influence

on UGV navigation. Thus, proposed algorithm is simpler than the listed complex distributed shortest path algorithms. First, we avoid loops in finding the minimum weights by using unit weight for every edge; second, the message transmitted in the algorithm is very simple, only the local level number itself; third, we do not employ any techniques to detect changes in the network, which is helpful for the time and energy savings in the WSAN. Since the application domain considered here is to navigate a UGV towards a single target hole consisting of possibly many target nodes, we consider the entire set of target nodes as a single target set. Algorithm 1 presents the pseudo-code of the level assignment algorithm for the single-target navigation problem.

Algorithm 1 Level Number Assignment Algorithm for Single-Target WSAN

```
1: if node is target node then
2:     l ← 0; broadcast l
3: else
4:     l ← ∞
5: end if{Initialization phase}
6: while time remaining do
7:     l_r ← the received level number from a neighbor
8:     if l > l_r + 1 then
9:         l ← l_r + 1; broadcast l
10:    end if
11: end while{Level number assignment}
```

The algorithm is straightforward and every node will eventually receive the correct level number under the assumption that there are no topological changes in the communication graph. What is essential and addressed in different manners in other schemes is when to terminate the process. In our case, we are more concerned with communication cost over performance time. Therefore, we assume that we have an estimated bound on the time it takes a single message to propagate from a target node to every other node. In particular, we assume that the time taken is $O(D)$ where D is the distance to the farthest node from the target in the graph. Clearly, $D < n$ but it could be significantly less in practice. We therefore have each node run the level assignment process for some factor of D units of time. In practice, this update can simply be a part of the regular message retrieval system whereby the nodes can update their level numbers as new information arrives.

We define the *communication complexity* as the maximum number of messages transmitted during the execution as in [11], [13], [14]. In this scenario, a message broadcast to multiple neighbors counts as one underlying cost. Since Algorithm 1 is event driven, messages are generated exclusively when a level number changes. Therefore, the communication complexity is asymptotically bounded by the maximum number of times that a level number changes. In [12], [13], the authors discuss communication complexity for synchronous communication models. In our distributed model, we look at asynchronous communication resulting in the inevitability of redundant messages because of potential transmission delays, as can be found for example in [15, Chapter 5].

Suppose there are n nodes in the network, with k target nodes. During the initialization phase of Algorithm 1, only target nodes send a message, yielding a communication complexity of $O(1)$ for each target node and $O(k)$ for all target nodes. In the level number assignment phase of Algorithm 1, the if-condition is always false for target nodes. For every non-target node, the first new level value received must necessarily be no more than $n-k$. Since each broadcast by a node occurs exclusively when the level number decreases until reaching a minimum of at least one, each node can therefore broadcast at most $n-k$ times. Thus, the total number of messages generated is $O(k+(n-k)\cdot(n-k))$ or $O(n^2)$ where k is a constant. In practice, particularly in coverage related problems, we expect the graph to be sparse.

B. UGV Control Algorithm

The control algorithm we present here uses similar concepts as in [16], which generates potential fields by a series of three actuator nodes. In the centralized algorithms of [3], [16], the sequence of the active actuator triplets is predetermined off-line. We propose a distributed, on-line navigation algorithm that proceeds in a series of steps. In each step, there are two phases: a communication phase where the specific potential field is determined for an intermediate target area, and a step movement phase where the UGV moves through the field towards this intermediate target area. The specific active potential field is determined in the communication phase. In the step movement phase, the UGV first calculates the next moving direction based on Equation 4 and then moves by the predefined step size in that direction, after which the UGV calculates a new direction. When the UGV reaches a local minimum of the potential field, the current step is completed and the communication phase of the next step starts.

For control purposes, it is more flexible and robust to use three actuator nodes at any given moment as opposed to a single node. For example, in a real-time implementation we can control the movement of the UGV in order to avoid certain coverage areas by adjusting the amplitude of a in Equation 1 independently for each actuator node. We use three actuators since the fourth or more actuators are not guaranteed can be found since we do not have any assumptions on network topology. Meanwhile, there becomes a tradeoff between the performance and the energy consumption and frequency interference when using more actuators.

At the initial step and whenever the UGV reaches a local minimum, the algorithm transits to the communication phase, where it will assign a triplet of actuator nodes, labeled node A, node B, and node C. To determine node B, the UGV communicates with its neighbors to pick the one with the lowest level number, for example node 2 in Fig. 2. When there is more than one neighbor with the same lowest level number, the UGV arbitrarily picks one of them. Once node B is chosen, node B communicates with its own neighbors and picks two neighbors to be nodes A and C. It is possible that in the initial step the first node chosen for node B has only one neighbor. However, as we show in the following section, this condition is trivial.

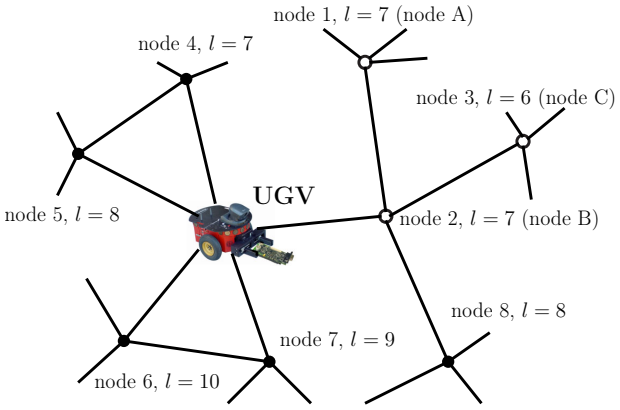

Fig. 2. UGV navigation in a WSAN, where each node's ID, level number and connections are shown. At this step in the example, we have the potential field generated by the triplet of actuator nodes A, B, and C, shown as circles, which are assigned to nodes 1, 2, and 3 respectively.

From Algorithm 1, we know that node B, unless it is a target node, has at least one neighbor that is at a level lower than node B. In the UGV control algorithm, node B assigns the role of node C to one of its lower leveled neighbors and then arbitrarily picks one other neighbor as node A. The triplet of actuator nodes A, B and C generates the current active potential field for the following movement phase. For simplicity, in the rest of the paper, we assume communication between the UGV and the triplet of actuator nodes is done primarily through node B. We now prove that when the UGV arrives at the local minimum of the potential field, it is always within distance r_c of all three active actuators.

Lemma 1: Any local minimum point p of the potential field is located in the area within distance r_c of all three active actuator nodes A, B and C.

Proof: Define d_{ip} to be the distance from node i to a local minimum point p. Without loss of generality, we can set $a = 1$ in Equation 3 making the combined potential field at point p to be $U_p = (d_{Ap})^m + (d_{Bp})^m + (d_{Cp})^m$. Assume for the sake of contradiction that our lemma is false and that p lies outside the stated area. Then at least one of the distances is larger than r_c. First, let us assume that B is furthest from p. Let q be a point infinitesimally closer to B on the ray extending from p to B. That is, $q = p + \epsilon(B - p)$ for some $\epsilon < 0$. Observe that $d_{qB} < d_{pB}$. Now examine the triangle formed by B, A and p. Since the edge from B to A has length $d_{BA} \leq r_c$ and since the edge from B to p has length $d_{Bp} > r_c$, the angle at p must have value less than $90°$ as edge BA cannot be the longest side. But this means that q lies inside the circle centered at A of radius d_{Ap}, for sufficiently small ϵ. Therefore, $d_{Aq} < d_{Ap}$. Similarly, we can show that $d_{Cq} < d_{Cp}$. This implies then that $U_q = (d_{Aq})^m + (d_{Bq})^m + (d_{Cq})^m < (d_{Ap})^m + (d_{Bp})^m + (d_{Cp})^m = U_p$. This contradicts the fact that p is a local minimum.

Now, let us assume that A is furthest from p, the case for C being symmetric. Again let $q = p + \epsilon(A - p)$ be a point infinitesimally closer to A. Using the property of the triangle formed by A, B and p as before, we can show that $d_{Bq} < d_{Bp}$. However, the case for node C is a bit trickier. If $d_{AC} \leq r_c$, then we can again use the triangle argument to show that $d_{Cq} < d_{Cp}$ yielding the contradiction that $U_q < U_p$. However, it is possible that $d_{AC} > r_c$. We now look at the change in the sum of the two distance terms associated with A and C as we move from p towards A. That is, we consider $\Delta = (d_{Aq})^m + (d_{Cq})^m - (d_{Ap})^m - (d_{Cp})^m$. Observe from our choice of q, that $d_{Aq} = d_{Ap} - x$ for some x infinitesimally close to 0 and that $d_{Cq} \leq d_{Cp} + x$. Thus, let $\Delta \leq f(x) = (d_{Ap} - x)^m + (d_{Cp} + x)^m - (d_{Ap})^m - (d_{Cp})^m$. Clearly, $f(0) = 0$. If we look at the first derivative of this function at 0, we see that $f'(x) = m((d_{Cp} + x)^{m-1} - (d_{Ap} - x)^{m-1})$. Consequently, $f'(0) = m((d_{Cp})^{m-1} - (d_{Ap})^{m-1}) \leq 0$ since from our assumption $d_{Ap} \geq d_{Cp}$ and $m \geq 1$. This means that $f(x) \leq 0$ for x infinitesimally close to 0. So, we have $U_q - U_p = (d_{Aq})^m + (d_{Bq})^m + (d_{Cq})^m - (d_{Ap})^m - (d_{Bp})^m - (d_{Cp})^m = (d_{Bq})^m - (d_{Bp})^m + \Delta$. Since $\Delta \leq f(x) \leq 0$ and $d_{Bq} < d_{Bp}$, we have that $U_q - U_p < 0$ which again contradicts the fact that p was a local minimum. ■

Lemma 1 guarantees that the UGV can always connect to a lower level node after reaching the local minimal point. That is, the next communication phase is guaranteed to pick a new node B whose level number is at least as low as the old node C for the next potential field. In general, when the UGV ultimately reaches a target node, there exists a series of nodes from high level nodes to the final level-0 node along the UGV's path that all acted as node B during certain step of the UGV control. The fact that the UGV can always communicate with node B also implies that it can detect the actuator signals from all three active actuator nodes as the distance from the UGV to the farthest of these three nodes is at most $2r_c \leq r_a$.

Based on Algorithm 1, every node except the level-0 node identifies its own level number by adding 1 to the smallest level number of its neighbors. For a degree-one node i, there is no higher level neighbor existing. As the UGV always travels from higher level nodes to lower level nodes, even if the UGV detects node i in its vicinity during its navigation, it always can pick another suitable node B with degree at least two. Thus, the UGV is only forced to assign node i as node B if the UGV is in the initial position of the navigation. However, there are many ways to avoid this situation. For example, for our original purpose of UGV navigation, the UGV can deploy one node at the current position to act as the third node needed to construct a potential field. Another advantage of Lemma 1 is that the UGV can send a single command to nodes A, B and C to turn the actuators off when the current step is completed, which helps to conserve energy in the network nodes. The pseudo-code of the UGV control algorithm is shown in Algorithm 2. In the pseudo-code, we do not include the process to choose nodes A, B and C, which has been discussed earlier.

The communication complexity of the control algorithm is fairly straightforward. At each phase, at most a constant number of messages is transmitted from the UGV to establish and turn on and off a triplet of actuator nodes at the current

Algorithm 2 UGV control algorithm

```
{Code for the UGV}
repeat
    broadcastMSG("UGV_request"); wait for response from all neighbors
    select node B, the neighbor with the smallest level number
    send message "UGV_nB_on" to B
    receive node ids of A and C from B
    repeat
        listen for actuator signals from nodes A, B and C
        calculate potential field at each listener
        move towards the minimum in the potential field
    until at a local minimum
    send message "UGV_off" to nodes A, B and C
until node B is a target node

{Code for all network nodes}
turn off actuator
loop
    m → recvMSG()
    if m == "UGV_request" then
        send message with level l to the UGV
    else if m == "UGV_nB_on" then
        broadcastMSG("NB_N" + l); wait for response from neighbors
        select nodes A and C based on lowest level numbers received
        send message "NB" to nodes A and C
        send ids of A and C to the UGV; turn on actuator
    else if m == "UGV_off" then
        turn off actuator
    else if m == "NB" then
        turn on actuator
    else if m == "NB_N" + B.l then
        {B.l is the level number from sender}
        if l < B.l then
            send level number l to B
        end if
    end if
end loop
```

UGV location and each actuator node responds at most once to this call. This means that there are $O(D)$ messages transmitted by the UGV and each node in the network transmits at most $O(1)$ messages, though in practice far fewer nodes will be involved. Thus, the communication complexity is $O(n)$.

Contrary to our initial assumptions, if the network topology of the WSAN changes, it is possible for the UGV to be jammed somewhere in the middle of navigation because a lower level node cannot be found. Though we do not formally cover this situation in this paper, it is possible to solve this problem once some corrective algorithms (including the algorithms we present in [3]) are triggered to start over again. We leave this as future work.

V. Simulation Results

A simulation testbed is built in Java, where the distributed algorithms are implemented using multi-threads: each individual node is designated a thread. The node is active only when the corresponding thread is running. Although we do not consider any communication delay models in this paper, the test-bed simulates asynchronous working patterns based on the properties of multi-threading. Messages might not be received in proper sequence since we do not enforce the running sequence of the threads, which inherently simulates the communication delays to some extent. In the simulations, we arbitrarily set one node as the destination node, arbitrarily set the UGV somewhere on the outer boundary of the WSAN and assume the navigation is complete once the UGV chooses the destination node as node B. All the evaluation results are plotted using MATLAB.

For illustrative purposes, in Fig. 3 we deploy a network having a simple topology and highlight the path taken during the UGV navigation in the WSAN. We designate the destination node with a star and the start and end positions of the UGV with two black boxes. The dots and edges represent network nodes, whose level numbers are also shown, and the communication connections between them, respectively.

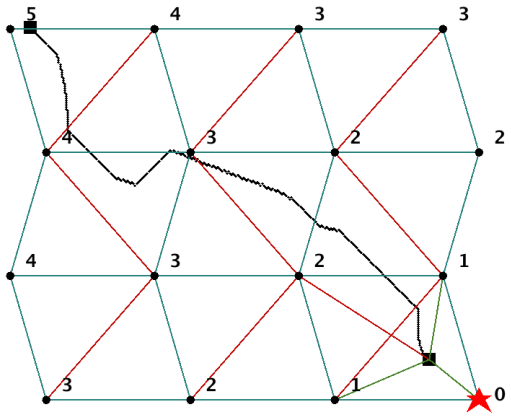

Fig. 3. Path taken by the UGV in the navigation.

We represent the overall network connections by network density, where the density of the WSAN is calculated by $\sigma = \frac{n \times \pi r_c^2}{Area}$ [17] with n being the number of nodes and $Area$ being the area of the sensor field. In all of our simulations, we use a fixed sensing area of size 800×600 and deploy 400 nodes. We alter the density by varying the value of r_c. To illustrate, consider a network in which the underlying communication graph G is a plane graph such that every region bounded by the edges of G, except the infinite region in the plane, is an equilateral triangle with edge length r_c. (The shape of G is similar to the graph shown in Fig. 3.) Setting $r_c = 35$ yields a network density of $\sigma = \frac{400(35^2)\pi}{800 \cdot 600} \approx 3.2$. We first analyze the average number of messages sent by each node for Algorithm 1 using different network densities; see Fig. 4. From the results, we can see that even when the network density is relatively high, the total number of messages sent is still far less than n^2, which indicates that Algorithm 1 should be practical for coverage related applications.

To evaluate the UGV control algorithm, we first measure the ratio $R = \frac{d}{d^*}$ as in [18], where d is the UGV's actual moving distance and d^* is the length of the shortest path from the starting point of the UGV through each of these minimum points in succession. A value of R closer to 1 indicates better accuracy. As we can see from the lower part of Fig. 5, the accuracy is improved with more listeners. Meanwhile, values of R do not change much with changes in network density. To see the influence of network density on the UGV control algorithm, with the same number of listeners on the UGV and

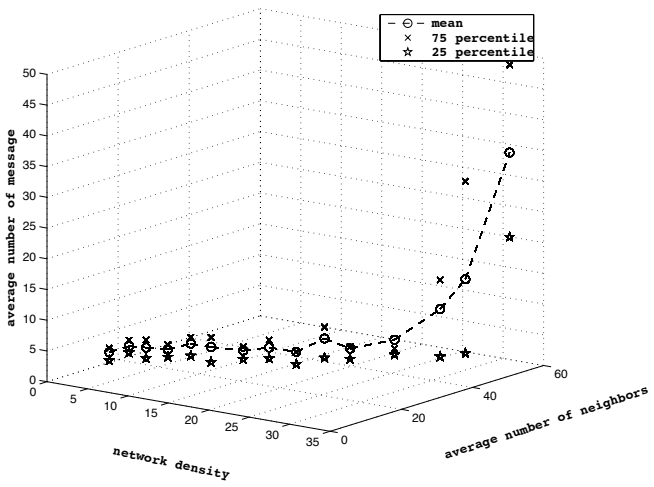

Fig. 4. Average number of messages sent by each node in the level assignment algorithm.

within the same network topology, we fix the start position of the UGV while varying the network density in each simulation trial. We calculate $p_i = d_i / \sum_1^{10} d_j$, where the numerator is the total distances travelled in the network with density i; denominator is the summation of all the travelled distances. We carried 100 trials in each network topology and the value of p_i is shown in the upper part of Fig. 5. Given that the straight line distance is almost the same since the start position and destination are fixed in each trial, p_i shows the percentage of travelled distances in each network density. In general, the UGV might travel longer distances in low density networks. This is because the maximum level number might be larger in lower density networks, which requires the UGV to adjust more frequently in order to find the right moving direction.

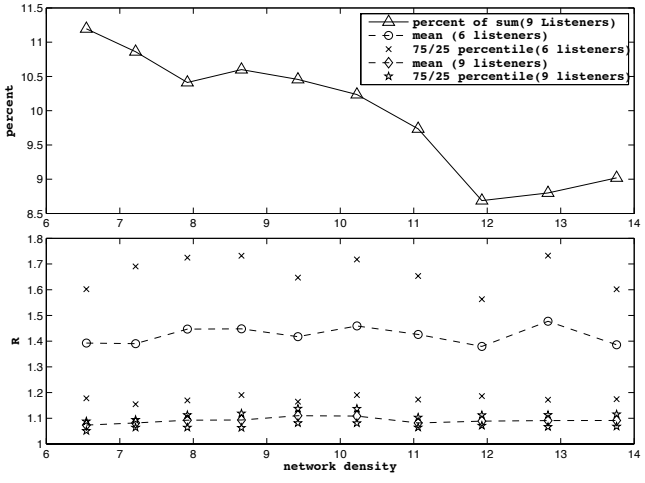

Fig. 5. Evaluation of the UGV moving algorithm.

VI. Conclusion and Future Work

We presented a theoretical analysis and simulation verification of a distributed UGV navigation algorithm in a coordinate-free wireless sensor and actuator network environment. We described algorithms for network hop-distance identification as well as UGV control including comunication complexity analysis. Future work will include multi-destination and multi-UGV navigation problems. A node energy model will also be formulated for consideration in the navigation path optimization.

Acknowledgment

This work was partially funded by Louisiana Board of Regents through PKSFI grant LEQSF(2007-12)-ENH-PKSFI-PRS-03.

References

[1] J. Kanno, J. Buchart, R. Selmic, and V. Phoha, "Detecting coverage holes in wireless sensor networks," in *17th Mediterranean conference on Control and Automation*, June 2009, pp. 452–457.

[2] J. G. Buchart, "Detecting coverage holes in wireless sensor networks," Master's thesis, Louisiana Tech University, May 2008.

[3] J. Yao, G. Zhang, J. Kanno, and R. R. Selmic, "Decentralized detection and patching of coverage holes in wireless sensor networks," in *Proc. SPIE Defense and Security*, vol. 7352, Orlando, FL, April 2009.

[4] M. A. Batalin, G. S. Sukhatme, and M. Hatting, "Mobile robot navigation using a sensor network," in *Proc. IEEE International Conference on Robotics and Automation*, vol. 1, 2003, pp. 636–641.

[5] Q. Li, M. DeRosa, and D. Rus, "Distributed algorithms for guiding navigation across a sensor network," in *Proc. the 9th Annual International Conference on Mobile Computing and Networking*, 2003, pp. 313–325.

[6] P. Chen, W. Chen, and Y. Shen, "A distributed area-based guiding navigation protocol for wireless sensor networks," in *Proc. IEEE International Conference on Parallel and Distributed Systems*, December 2008, pp. 647–654.

[7] S. Fu, Z. Hou, and G. Yang, "An indoor navigation system for autonomous mobile robot using wireless sensor network," in *Proc. IEEE International Conference on Networking, Sensing and Control*, May 2009, pp. 227–232.

[8] D. Chen, B. Kumar, C. Mohan, K. Mehrotra, and P. Varshney, "In-network path planning for distributed sensor network navigation in dynamic environments," in *Proc. IEEE International Conference on Mobile Ad Hoc and Sensor Systems*, Octobor 2008, pp. 511–513.

[9] T. Rappaport, *Wireless Communications: Principles & Practice*. New Jersey: Prentice-Hall, Inc, 1996.

[10] The cricket indoor location system. [Online]. Available: http://cricket.csail.mit.edu/

[11] P. A. Humblet, "An adaptive distributed dijkstra shortest path algorithm," Massachusetts Institute of Technology, Laboratory for Information and Decision Systems, Tech. Rep., 1988.

[12] P. Merlin, "Design and analysis of distributed routing algorithms," Master's thesis, University of California, Santa Cruz, 1994.

[13] J. Spinelli, "Broadcasting topology and routing information in computer networks," Master's thesis, Massachusetts Institute of Technology, May 1985.

[14] J. Tsitsiklis and G. Stamoulis, "On the average communication complexity of asynchronous distributed algorithms," *Journal of the ACM*, vol. 42, no. 2, 1995.

[15] D. Bertsekas and R. Gallager, *Data networks*. Prentice-Hall, Inc, 1987, ch. 5.2.4.

[16] J. Schiff, A. Kulkarni, D. Bazo, V. Duindam, R. Alterovitz, D. Song, and K. Goldberg, "Actuator networks for navigating an unmonitored mobile robot," in *Proc. of IEEE Conference on Automation Science and Engineering*, Washington DC, August 2008.

[17] H. Zhang and J. C. Hou, "Maximizing α-lifetime for wireless sensor networks," in *Proc. of Third Int'l Workshop on Measurement, Modelling and Performance Analysis of Wireless Sensor Networks*, July 2005.

[18] G. Zhang, C. Duncan, J. Kanno, and R. R. Selmic, "Unmanned ground vehicle navigation in coordinate-free and localization-free wireless sensor and actuator networks," in *Proc. 2010 IEEE Multi-conference on Systems and Control*, Yokohama, Japan, September 2010.