Distributed Localization of Wireless Sensor Networks
Using Self-Organizing Maps

Jie Hu and Gordon Lee

Abstract - As larger sets of wireless sensor networks are being deployed, an important characteristic of the network which could enhance its capabilities is position awareness. While several approaches have been proposed for localization, that is, position awareness without using GPS, most techniques are either centralized or rely on anchor nodes. In this paper, a decentralized localization method is developed, based upon self-organizing maps. The algorithm is implemented for different size networks and the simulation results show the algorithm is efficient when compared to single processor or centralized localization methods; further the approach does not require anchor nodes. An error analysis shows that the proposed approach is a feasible method for computing the localization of sensor networks using a distributed architecture.

Keywords – Localization, self-organizing maps

I. INTRODUCTION

With the increased growth of wireless sensor networks (WSN) for both civilian and military applications, there is a need for developing better ways to deploy and use these sensor networks. An important piece of information which can enhance efficiency is position-awareness, that is, knowledge of where each sensor node is in a network, particularly if the node is moving (e.g., PDA or cell phone).

Having knowledge of sensor locations allows one to use diverse sensor data more efficiency, plan resource routing priorities to support network services or perform surveillance effectively [1]. Without sensor location information, a wireless network, for example, that detects large fluctuations in heat may trigger the fire alarm correctly but not allow for a specific action, such as closing the appropriate fire doors or activating the correct group of sprinklers, since geographical knowledge about sensor location is unavailable. When multiple sensors are detecting a common event, the locations are needed to judge the extent or content of the event. Also, knowledge of sensor locations may assist an operator in assessing whether or not certain sensor information is redundant.

For example, if multiple sensors in the same area indicate a similar event, this may verify the occurrence of that event. Or, in routing communication through a series of sensors, knowledge of sensor locations allows one to efficiently map the communication paths through the network.

Given the importance of localization, several recent research efforts have focused on incorporating position awareness in sensor network design. In [2], the authors propose a lightweight localization scheme that works without anchor nodes and does not rely on range measurements. The algorithm is based on a specific neural network architecture known as a Self-organizing Map (SOM), which generates virtual coordinates that describe the relative positions of nodes. The authors also discuss how this information is used to compute absolute positions.

The approach is a centralized one, however, and is also limited to a small to medium size networks. Langendoen and Reijers [1] compare several decentralized localization techniques; however these methods require anchor nodes or possess the capability to extract range and angle measurements.

In this paper, we propose an algorithm that is an extension of our efforts proposed in [3] which was based on [2] using Self-Organizing Maps; however the approach presented in [3] and extended here is a decentralized method. Further, as in [2], the proposed approach provides anchor-free and range-free schemes, meaning that none of the nodes are placed at known positions and no attempt is made to estimate the inter-node distances. The algorithm computes the map of a network using only connectivity information that specifies which nodes are in radio range.

Section 2 provides a summary of the classical Self-Organizing Map structure while, in Section 3, a description of the centralized localization algorithm developed in [2] is presented. In Section 4, another form of the SOM algorithm is discussed, one which has a distributed architecture. Using this architecture, the decentralized localization algorithm using SOMs is developed in Section 5, which is the main contribution of this paper. In this section, a generalization of the nearest neighbor matrix is employed. In Section 6, the proposed approach is applied to several simulated test scenarios. The evolution of the localization is also shown. In Section 7, a discussion on the computational complexity of the approach is presented while concluding remarks are made in Section 8.
II. CLASSICAL SELF-ORGANIZING MAPS

For completeness sake, a brief discussion of self-organizing maps (SOM) is presented here. A SOM is essentially a two-dimensional neural network (Figure 1) that contains two layers of cells or nodes – an input layer and a network layer. In the network layer, the nodes are arranged in a lattice-type array. This array can be constructed using any two-dimensional shape (e.g., a rectangular, square, or hexagon). In this paper, the lattice is constructed as a square. All the cells are associated with a set of weight vectors denoted by \( w_1, \ldots, w_N \in \mathbb{R}^n \) where \( N \) is the number of cells and \( n \) is the dimension of the weight vectors. The elements of the weight vector may be crisp or fuzzy. Here, crisp values are used; fuzzification is an area of future research. The input layers hold the data samples from which the networks will learn.

The input data can be represented by a set of weight vectors \( x = [\xi_1, \xi_2, \ldots, \xi_N]^T \in \mathbb{R}^n \). The dimensions of the weight vectors in both layers could be of high dimension; however, the dimension of both input vector and the network vector should be the same.

Following the notation of [3], a recursion for updating the weights is as follows:

\[
   w_i(t+1) = w_i(t) + \alpha(t)[x - w_i(t)] \quad \text{if} \quad i \in N_{BMU} \quad (1)
\]

Else, \( w_i(t+1) = w_i(t) \), where \( BMU \) is the location of the closest neuron to the data \( x \) (e.g., actual distance or signal intensity), and \( \alpha(t) \) is the learning rate which must satisfy some additional conditions for the stochastic algorithm (see [4], [5]); further \( t \) is the time index and \( N_{BMU} \) is a neighborhood of neuron BMU. The neuron BMU is called the Best Match Unit.

A SOM projects the input space to the two-dimensional plane defined by the lattice of neurons. This property has been widely exploited in many applications for data analysis and visualization of large data sets [4] [6]. More recently, SOMs have been used to implement localization schemes for mobile robots in unknown environments [7] [8] and the localization problem in wireless sensor networks [9].

III. CENTRALIZED LOCALIZATION BASED ON THE CLASSICAL SOM

The algorithm proposed in [2] is based on the classical SOM architecture and explicitly computes individual node positions as a result of the training phase of the map, without relying on sensor readings or time synchronization.

To use SOMs, one assumes a connected network with \( N \) nodes placed at unknown locations \( (x_i, y_i) \), \( i = 1, \ldots, N \). Further, one assumes that none of the nodes is equipped with hardware for position, range or angle estimation (e.g., GPS, ultrasound receivers or smart antennas); no assumption is made regarding the availability of the sensors at each location. Furthermore one assumes that each node can determine the set of its neighbors (e.g., through some signal intensity) and can transmit this information to a central point of computation. For completeness sake, the details of the algorithm can be described as follows [2]:

Input: matrix \( H_c \) - the matrix containing the hop count distances between nodes and \( N \) - the number of nodes
Output: \( (x_j, y_j), j = 1, \ldots, N \): node positions

//Initializations
1. Choose: \( a_{\text{max}} = 0.1 \) and \( a_{\text{min}} = 0.01 \) to be the maximum and minimum values of the learning rate, respectively.
2. Choose \( r_{\text{max}} = \max H_c(i,j)/2 \) and \( r_{\text{min}} = 0.001 \) to be the maximum and minimum radius of the neighborhood function [2], respectively.
3. for all nodes \( j \) do
4. \( (x(j), y(j)) = \text{random}(0,1) \)
5. end for % generate uniformly distributed random % positions between [0,1]

// Main loop
6. for \( t = 1: N_{\text{iter}}-1 \) do
7. \( \alpha(t) = a_{\text{max}} \times t^*(a_{\text{max}}-a_{\text{min}})/(N_{\text{iter}}-1) \);
8. \( r(t) = r_{\text{max}} \times t^*(r_{\text{max}}-r_{\text{min}})/(N_{\text{iter}}-1) \); % linearly decrease values
9. \( (x,y) = \text{random}(0,1) \)
10. \( BMU = \arg \min_j ||(x,y) - (x_j, y_j)|| \) % BMU is index of best matching unit
11. for all nodes \( j \) do
12. \( h = \exp \{-H_c(BMU,j)/2r(t)\} \)
13. \( (x(j), y(j)) = (x(j), y(j)) + \alpha(t)h[(x,y) - (x(j), y(j))] \)
% updates new node positions

14. end for
15. end for

Since the algorithm is based on the classical SOM, which is centralized, each node needs to communicate the list of its neighbors to the unit in charge of the computation so that this information can be used to build the adjacency matrix \( G_{\text{NET}} \), and then compute the hop-count distances between each pair of network nodes, which are stored in matrix \( H_c \). The matrix \( H_c \) is a required input to the centralized localization algorithm. The complexity of this computation is about \( O(N^3) \) [2]. Hence, for small to medium size networks (10 ~ 100 nodes) with low connectivity, the algorithm is demonstrated to be efficient in solving the localization problem [2].

In Section 5, a decentralized localization algorithm is developed based on a distributed SOM algorithm [4]. This distributed SOM algorithm is now discussed.

IV. A DISTRIBUTED ALGORITHM FOR GENERATING THE SOM

The development of a distributed SOM in [4] uses the same set of neurons and sequence of observations as the SOM described in Section 2; however an energy function is constructed which leads to a distributed architecture for the self-organizing map. The algorithm is described as follows.

Let the weighted vector be updated according to the recursion:

\[
\sum_{j=1}^{N} \delta_{ij} [x - w_i(t)]
\]

\[
\delta_{ij} = \frac{g_{ij} \exp\{-\frac{||x - w_i||^2}{\tau}\} \exp\{-\frac{||x - w_j||^2}{\tau}\}}{\sum_{k=1}^{N} \sum_{m=1}^{N} g_{km} \exp\{-\frac{||x - w_k||^2}{\tau}\} \exp\{-\frac{||x - w_m||^2}{\tau}\}}
\]

(2)

where \( x \) is an observation sample at time \( t \) and \( g \) indicates the neighborhood relation between two neurons:

\[
g_{ij} = \begin{cases} 
1 & \text{ith, jth neurons are neighbors} \\
0 & \text{otherwise}
\end{cases}
\]

This algorithm forms a topographic map resembling the self-organizing map, as developed in [4]. Note in this algorithm, there is no need to find the Best Match Unit (BMU) first and then update its neighborhood nodes \( N_{\text{BMU}} \). All nodes are updated simultaneously, thus providing a distributed approach in the construction of the SOM.

V. THE DISTRIBUTED LOCALIZATION ALGORITHM

Based on the distributed SOM generation algorithm, the localization algorithm in Section 3 can be modified as follows:

Assume there are \( N \) nodes in the network and they are arranged in a two dimensional grid. A node always sends/receives information to/from its closest neighbor nodes. Without loss of generality, let each node (with some trimming at the edges) have \( q \) neighbors. This is an extension of previous work [3] in which only four nearest neighbors were assumed. We define \( g_{ij} = 1 \) if node \( j \) is one of \( q \) neighbors of node \( i \), otherwise \( g_{ij} = 0 \). Since all nodes are located in a two dimensional grid, \( g_{ij} \) can be calculated easily.

Input: \( N \), the number of nodes; \( G = (g_{ij}) \), knowledge of nearest neighbors
Output: node positions \( P_i = (x_i, y_i) \), \( i = 1, ..., N \)

// Initialization of the node locations
1. for all nodes \( i \) do
2. \( P_i = (x_i, y_i) = \text{random}(\) ;
3. end for

// Main Loop
4. for \( t = 1 \) to \( N_{\text{iter}} \) do
5. \( P = (x,y) = \text{random}(\) ;
6. for all network nodes \( i \), update its location
7. \( x_i(t+1) = x_i(t) + \alpha(t) \sum_{j=1}^{N} \delta_{ij} [x - x_i(t)] \)
8. \( y_i(t+1) = y_i(t) + \alpha(t) \sum_{j=1}^{N} \delta_{ij} [y - y_i(t)] \)
9. end for
10. end for

Notice in this algorithm, no central node is necessary. Each node can update its location in steps (7) and (8). At each loop step \( t \), a node updates its location by the information it received from its neighbors (as shown in the numerator in step 7) and the information from all other nodes (as shown in the denominator in step 8). If the information has not been received and updated, the node can implement the algorithm in the following two different ways:

(i) wait for the newest updates needed from all other nodes to start next update; or
(ii) start the next update using the most recently received information without waiting.
We call (i) a synchronous implementation and (ii) an asynchronous implementation. In the case of the asynchronous implementation, since outdated information is used, convergence becomes an issue. In [10], the convergence of asynchronous Self-Organizing Maps is discussed in detail.

The formula for asynchronous computation in steps 7 and 8 are:

7. 
\[
x_i(t+1) = x_i(t) + \alpha(t) \sum_{j=1}^{N} \delta_{ij} [x - x_i(t)]
\]

8. 
\[
y_i(t+1) = y_i(t) + \alpha(t) \sum_{j=1}^{N} \delta_{ij} [y - y_i(t)]
\]

\[
\delta_{ij} = \frac{g_{ij} \exp\{-\frac{||P - P_i||^2}{\tau}\} \exp\{-\frac{||P - P_j||^2}{\tau}\}}{\sum_{k=1}^{N} \sum_{m=1}^{N} g_{km} \exp\{-\frac{||P - P_k||^2}{\tau}\} \exp\{-\frac{||P - P_m||^2}{\tau}\}}
\]

where \(P_k = [x_k(\sigma_k(t)), y_k(\sigma_k(t))]\)  \(\text{ (3)}\)

and \(\sigma_k(t)\) is a previous time interval satisfying 
\[
t - B \leq \sigma_k(t) \leq t
\]
\[
\sigma_k(t) = t
\]

Further, \(B\) is the constant.

We note that, by generalizing the number of nearest neighbors (rather than limiting the number to four as in [3]), the accuracy and speed of convergence, i.e., the computational cost is reduced (as can be seen by comparing Table 1 below and Table 1 in [3]; further the generalization provides more flexibility in finding the convergent topology.

In the next section, we implement the proposed algorithm and show that convergence can be achieved with a decentralized architecture.

VI. SIMULATION RESULTS

In this section, the algorithm is simulated on a distributed, parallel machine in the Supercomputing Institution at University of Minnesota. To perform the simulations, we chose the same approach as in [2]; that is, initially, equally spaced nodes were generated and then perturbed by adding Gaussian noise with zero mean and variance \(\sigma\). Then we implemented the proposed decentralized localization algorithm developed in Section 5. We chose 25, 36, 49, 64, 81 and 100 processors to simulate the nodes in different wireless networks to run the algorithm.

Figure 2 shows a typical run, that is, convergence plots of a 25 nodes network using 25 processors, 5000 iterations and \(q = 8\). The standard deviation of the Gaussian noise was chosen at 0.2 and the synchronous mode of the proposed algorithm was used.

We note that the results are comparable to those obtained in [2] but, by using a decentralized approach, the need for knowledge of the node hop distance matrix \(H_c\) is eliminated; rather only knowledge of a node’s nearest neighbors is required. The approach also provides convergence faster and more efficient as is described in the next section. The following section also presents an error analysis of the decentralized localization approach.

Fig. 2. Convergence plots of the decentralized SOM network with 25 nodes

VII. ANALYSIS OF THE DECENTRALIZED LOCALIZATION ALGORITHM

We analyze the proposed decentralized localization algorithm in several ways. First an error analysis is performed. To do this, we measure the average distance from each node to its neighboring nodes to check how the graph evolves into an ordered map, resulting in convergence.

Figure 3 shows the results of an error analysis at certain iterations of the algorithm. To perform the error analysis, we calculate the average distance from each node to all of its neighbors, then find the average distance error for all the nodes and plot this error at each iteration. We note that the error decreases until at about 4000 iterations, it reaches a constant error value. The results are comparable with those found in [2] but by using less computation, as will be discussed below.

Next we wish to investigate the computational burden of the proposed decentralized localization algorithm compared with the centralized approach of [2].

Using \(N\) processors (where \(N\) is the total number of nodes), step 7 in the new algorithm requires \(N\) operations.
while step 8 requires $N^2$ operations at most (actually much less due to a lot of zeros in $G$). Hence conservatively, the new decentralized localization algorithm's computational burden is of order $O(N^2)$, which is better than the centralized approach in [2] which is of order $O(N^3)$.

Further, we can compute the computation time and communication time in using the decentralized localization approach proposed here (Table 1). We note that the total time is much less than the case when a centralized method is used, even with the additional communication time.

A comparison of localization methods that employ one processor versus multiple processors for the synchronous and asynchronous modes is discussed in [3] for the case of using four nearest neighbors. The trend is similar to the method proposed in this paper where the number of nearest neighbors is generalized (eight is used in the simulations here).

As not all nodes in a network have the computational power to perform the operations defined in the decentralized algorithm, the sensor network can be partitioned into smaller regions so that each region has one node in charge of the computation for the locations of all rest of the nodes in that region. We call this case supervisory decentralized localization. To simulate this case, we looked at two cases: a network with 225 and a second network with 400 nodes. We divided the first network into 45 equal regions, where for each region one is selected to be in charge of the computation for the locations of the remaining of nodes in the region. We used 45 processors to simulate the nodes in charge of these computations. Similarly for the second case, we divided the network into 80 regions and used 80 processors.

The convergent topologies and execution cost are shown in Figure 4 and Table 2 for the case when four nearest neighbors are used.

![Evolution of the average distance error as a function of the number of iterations](image1)

**Fig. 3.** Evolution of the average distance error as a function of the number of iterations

**TABLE I**

<table>
<thead>
<tr>
<th># of nodes</th>
<th>Computational time (s)</th>
<th>Communication time (s)</th>
<th>Total time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>0.150873</td>
<td>17.8491</td>
<td>18.000</td>
</tr>
<tr>
<td>36</td>
<td>0.248936</td>
<td>31.7511</td>
<td>32.000</td>
</tr>
<tr>
<td>49</td>
<td>0.381786</td>
<td>50.6182</td>
<td>51.000</td>
</tr>
<tr>
<td>64</td>
<td>0.604475</td>
<td>71.3955</td>
<td>72.000</td>
</tr>
<tr>
<td>81</td>
<td>0.773779</td>
<td>96.2262</td>
<td>97.000</td>
</tr>
<tr>
<td>100</td>
<td>1.04816</td>
<td>143.952</td>
<td>145.000</td>
</tr>
</tbody>
</table>

**TABLE II**

<table>
<thead>
<tr>
<th># of nodes</th>
<th>One processor (Sec.)</th>
<th>Synchronous (Sec.)</th>
<th>Asynchronous (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>225</td>
<td>18512</td>
<td>615.8</td>
<td>352</td>
</tr>
<tr>
<td>400</td>
<td>733061</td>
<td>1056.76</td>
<td>659</td>
</tr>
</tbody>
</table>

We note that even in this case, the decentralized localization approach performs faster than the centralized method of [4]. Hence we conclude that the proposed approach is more efficient and faster than the centralized localization approach, even with the additional communication time.

**VIII. CONCLUSIONS**

In this paper, a distributed localization algorithm for wireless sensor networks is proposed. The algorithm is based on the use of Self-Organizing Maps, which does not rely on any central nodes or anchor nodes. We demonstrate that by formulating a decentralized approach, the algorithm is efficient with computational burden in the order of $O(N^2)$; further, the convergence properties are illustrated through simulation results. The algorithms can easily be implemented on a parallel or distributed parallel system.

Future work includes investigating an extension of the proposed algorithm to the case when information of anchor nodes is available as well as developing a fuzzy version of the SOM for decentralized localization.

**IX. REFERENCES**


