ECE790 Masters Research - Project Report

Localization in Sensor Networks using Message Passing Algorithm

Advisor: James Bucklew *Written by: Usman Khan*

1 Abstract

We describe a powerful and different approach to solve the localization problem of sensor networks with a very few anchor nodes (we define anchor nodes as those sensors that are equipped with GPS receivers so that they are fully aware of their location). The triangulation procedure (employed usually to solve this problem) fails when we don't have the anchor nodes lying close to each other, since it requires for any sensor node, at least three anchor nodes in its neighborhood to find its exact position estimate. Our algorithm finds out the position estimates of these sensor nodes. The sensor nodes then act as anchor nodes (for the sensor nodes whose positions are still unknown) increasing the number of anchor nodes in the network, making it possible to carry out the triangulation procedure. The algorithm is based upon the two dimensional distance matrix computed for all the nodes present in the network and a simple message passing scheme. The network is converted into a graph (a dense graph if the number of nodes is large) and subsequently random spanning trees are generated from it. Working on the spanning trees is beneficial since we avoid getting caught in the cyclic nature of the graph. The algorithm takes into account the random nature of the different spanning trees selected. The message is then passed on these spanning trees one by one, their order selected at random. The message propagates from the root of the tree to the succeeding levels/generations. The algorithm then updates the position of the current node by minimizing a two dimensional equation based on distance matrix and the information contained in the message[1]. We further use a cross validation procedure to verify our estimates.

2 Introduction and Motivation

Sensors networks have numerous possible applications e.g., measuring the temperature in a given field, measuring the velocity of some vehicle etc. They are used in many scenarios where remote access is the only possible way to explore an area and physical measurements taken by humans are out of question. The reason for no human access could be that the area is enemy territory or is contaminated with some environmental hazard. For many applications the number of sensors present in a field is large. Large networks with thousands of inexpensive, battery operated, wireless sensors are becoming a reality [2]. The exact geographical position of the sensors is often an important piece of information required to organize the results they provide. It is inefficient to put a lot of computation power, memory and smart processors in these sensors. What often desirable is to have many inexpensive sensors in the field so that even if a number of them fail, it would not affect the overall goal and performance.

Consider a battlefield where a number of sensors are deployed to discover the movement/presence of enemy vehicles. These sensors could communicate to a central node which can process information from all the sensors to achieve some result. If we were to find the position of some target in the field, it would be desirable to know the exact position of these sensors. The position of these sensors could not be known beforehand since all of them are deployed randomly. For this localization problem, it is highly expensive if we put a GPS receiver in each sensor. So an algorithm that can use a small amount of local information and give us the position estimate of these sensor nodes would prove very useful.

In this paper, we present a method of computing the position estimates of the sensors with taking into account only the distance matrix of the network and a very few number of anchor nodes (i.e., the sensors equipped with GPS receivers).

[1]The idea of message passing is similar to the one presented by Wainwright et. al. in [1].

3 Related Work

Shang et al. [2] based their algorithm on multi-dimensional scaling yielding coordinates that provide the best fit to estimated pairwise distances lying in an arbitrary rotation and translation. Then they use some anchor nodes to find the absolute positions for all the nodes in the network. This technique works well with reasonably few anchor nodes and high connectivity. Patrick Biswas et al. [3] describes an SDP (Semi Definite Programming) relaxation based method for this problem. The basic idea behind their technique is to convert the non-convex quadratic distance constraints into linear constraints by introducing a relaxation to remove the quadratic term in the formulation. Niculescu et al. [4] describe a 'DV Hop' approach in which the anchor nodes disseminate their positions throughout the network and then each node performs a triangulation using three or more anchor nodes.

Bahl et al. [5] suggest estimating distance between the nodes in the network based on the signal strength by applying a wall attenuation factor (WAF) based signal propagation model. This distance information is then used to locate a user by triangulation. This approach, however, yielded lower accuracy than RF mapping of signal strengths corresponding to various locations for their system. Their RF-mapping-based approach is quite effective indoors but requires extensive infrastructural effort.

Another localization method is the proprietary Location Pattern Matching technology, used in U.S. Wireless Corporation's RadioCamera system [6]. They employ a signal structure database generated by a vehicle that drives through the coverage area. This vehicle transmits signals to a monitoring site which compiles a unique signature for each square in the location grid. To determine the position the RadioCamera system matches the the transmitter's signal to an entry in the database. The major drawback of this approach, as with RADAR[5], is the substantial effort needed for generation of the signal structure database [7]. Another drawback of this system is its non-feasibility in a battle field or a remote area where access is not possible.

Mazzini [8] has based his localization procedure by fixing two anchor nodes at the corner of a rectangle. From there triangulation is done to get the estimates. Some other approaches also place a similar kind of constraint on the anchor nodes that they should be placed at some fixed positions in the setting. We describe a setting where anchor nodes can be placed randomly anywhere in the field.

Most of the approaches presented above rely on the triangulation of the nodes to get the position estimates but this could only be done once we can find at least three anchor nodes with known positions in the communication radius of all the sensors with unknown positions. Problems also lie with distributed approaches, mainly due to the coherency and synchronization issues. In our approach we rely heavily on the estimates of the sensors at the previous time step since we move iteratively. In distributed environment we need to make sure the availability of the information before we can proceed to the next iteration.

4 Some Graph Theory Basics

A graph is a collection of points and lines connecting some subset of them. A graph G can be defined to be a pair of $(V(G), E(G))$, where $V(G)$ is a non-empty finite set of elements called vertices and $E(G)$ is a finite set of unordered pairs of distinct elements of $V(G)$ called edges. $V(G)$ is sometimes called the vertex set and $E(G)$ is the edge set. The edges of graphs may also be imbued with directedness. A normal graph in which edges are undirected is said to be undirected. Otherwise, if arrows may be placed on one or both endpoints of the edges of a graph to indicate directedness, the graph is said to be directed. Formally, a digraph D, (or a directed graph) is defined to be a pair $(V(D), A(D))$, where $V(D)$ is a non-empty finite set of elements called vertices, and $A(D)$ is a finite family of ordered pairs of elements $V(D)$ called arcs. $A(D)$ is called the arc family of D. An arc whose first element is v and whose second element is w is called an arc form v to w; which is different from an arc from w to v [9]. In this problem, we only deal with undirected graphs.

A tree can be defined to be a graph which contains no circuits or cycles. Given any connected graph G, we can choose a circuit and remove one of its edges, the resulting graph remains connected. We repeat this process with one of the remaining circuits, continuing until there are no circuits left. The graph which remains will be a tree which connects all the vertices of G; it is called a spanning tree. The depth or generation or level of a node n, in its tree T, is the length of the path from n to T's root. The path is the unique shortest sequence of edges from a node n to an ancestor [9, 10].

5 Algorithm

We employ the usual two dimensional Euclidean metric space. The distance d_{xy} between any two points x and y in this space is given by

$$d_{xy} = \sqrt{\sum_{n=1}^{2} (x_n - y_n)^2}.$$

Euclidean distance matrix, D in R_+^{NxN} is an exhaustive table of distances between points taken by pair from a list of any N points $\{x_1, x_2, \ldots, x_N\}$. Each point is labeled ordinarily, hence the row or column index of the distance matrix i or $j = 1 \ldots N$, individually addresses all the points in the list. This distance matrix, D has N^2 entries but only $N(N-1)/2$ pieces of information. The distance matrix, D satisfy all the axiomatic requirements imposed by any metric space.

Let's consider an ideal version of the problem. For now assume there are no sensors but N points in an arbitrary flat rectangular grid, lying in the two dimensional Euclidean space. These N points are placed randomly in this plane. We claim no information about the exact position for all these N points. We further assume M more points in the same setup as above. These points are also placed randomly in this plane. These points are different than the previously defined N points. We claim that for these M points we precisely know their exact location in 2-dimensional Euclidean space. For now assume that somehow we know the distances between all these $N + M$ points and we have a Euclidean distance matrix, D with dimensions $(N + M)$ x $(N + M)$ for these $N + M$ points. There are several references in literature to find this distance information based received signal strength [5], time of arrival, time difference of arrival and direction of arrival [11, 12, 13].

Using the graph theory described above, we consider these $N + M$ points as the vertices of an undirected graph, G. In other words $V(G) = \{x_1, \ldots, x_N, y_1, \ldots, y_M\}$. We further impose one more restriction which takes us one step away from the idealized situation. There is a minimum communication radius, R, which restricts the communication among the nodes. We now say that any point x in the above setup can talk to all those points which lie in a radius, R around it. For x, we place an edge from x to all those points which lie in its communication radius, R. Mathematically, ordered pair $(x, y) \in E(G) iff d_{xy} <= R$, and this holds for all the pair of vertices lying in the vertex set $V(G)$ of G. The edge between x and y is weighted by the distance d_{xy} between x and y. This weight d_{xy} is also the (x, y)th entry in D. This holds true for all the edges lying in $E(G)$.

We are now in a position to describe the problem. We wish to find out the position estimates of all the N points (whose positions are unknown) based on only the graph G and the distance matrix, D.

Let $X = \{x_1, x_2, \ldots, x_n\}$ is a set of sensor nodes and $Y = \{y_1, y_2, \ldots, y_m\}$ be a set of anchor nodes (by anchor nodes we refer to those sensors which are equipped with GPS receivers and hence, they in addition to perform all the tasks of a sensor, also know their exact positions), such that $|X| = N$ and $|Y| = M$. These $(N + M)$ nodes correspond to $(N + M)$ points described in the idealized setup above. So, we can construct a graph, G, and then have $V(G)$, $E(G)$ and D on G, in the same way we did in the idealized setup.

Fixing every element of Y_{p-1} as the root, we grow a spanning tree and add it into S_p. The subscript p refers to the phase. Notice that we would have different roots and hence different spanning trees in every phase. Although, there is an exception to this[2].

Now we would randomly select one element, t_i from S_p at every iteration, i. The subscript i shows the dependance of the spanning tree picked on each iteration(recall we randomly pick a tree at each iteration). The message is propagated on this randomly picked tree. For each node, n_g, in each generation, g_t, for this tree, t_i, we pass the following message to all it's children. If there are J generations, we go until $J - 1$ generations since the Jth generation has no children.

For the node, q in $t_i \in S_p$, the message is

$$p_{new} = argmin||p_{new} - p_{old}||^2 + \left| ||p_{new} - p_{parent}||^2 - d^2_{parent,node} \right|$$

where p_{new} is the new estimate of the position of node q that minimizes the the above equation

p_{old} is the old estimate of the position of node q, i.e., at the last iteration

[2]

p_{parent} is the current position estimate for the parent of node q which was updated at the previous step
$d_{parent,node}$ is the distance between the node q and its parent taken from the D.

We present the pseudo code for the algorithm as an attempt to make it more clear. We define two methods TriProc and CrossValidation which will help us in writing the pseudo code. TriProc is a routine which updates the position of all those sensor nodes which have at least three or more anchor nodes (or the sensors whose position estimates are known) lying in their communication radius, R.

```
——ALGORITHM——
phase, $p = 0$
$Y_p = Y$
TriProc(X, Y)
while $X$ is nonempty do
  $p = p + 1$
  $S_p = \{$spanning trees with elements of $Y_{p-1}$ as roots (one tree per root)$\}$
  for $i = 1 \dots I$ do
    randomly pick a tree, $t_i$ from $S_p$
    propagate message on $t_i$
  end for
  CrossValidation() [2]
  add 'good' sensors to $Y_p$[3]
  $Y = Y \bigcup Y_p$
  $X = X - Y_p$
  TriProc(X, Y)
end while
——ALGORITHM——
```

We continue by randomly picking a tree, t_i, from S_p at every iteration and propagate the message until we see the variance of estimates going down. In this way, we exploit all we can get from the set, S_p of spanning trees. At every step in each iteration, we work down the generation of the trees. We start from a generation and based on the above message update the estimate of the position of all the nodes in the next generation. We keep on working all the generations out and hence all the nodes in each generation.

At the end of phase one, we get some sensor nodes which have converged to their actual position due to message passing. After Cross Validation[1], we add these sensor nodes to our anchor nodes set. In other words, the set X gets smaller and the set Y gets larger, since we add those sensor nodes whose actual position is now known, to Y and remove them from X. We use the TriProc routine to update all the sensor nodes based on the triangulation, since we have a larger set Y and it is more likely that some sensor nodes will now find at least three or more anchor nodes in their communication radius, R. It is observed that at this point we have almost all the network converged to the correct position estimates.

In case, we still have a non-empty X we would proceed to the next phase of the algorithm. We generate S_p from Y_{p-1}, which now contains the roots for the newly converged sensor nodes. The way we select the roots, makes sure that the same anchor node is not picked as the root which was included in S_{p-1}. We can proceed on to more phases in the same manner until we have computed the actual positions of all the sensor nodes. It is conceivable to be able to resolve positions of some sensors with fewer than three neighbors (depending on geometry) but we do not attempt that here.

6 Cross Validation

One of the problems unaddressed in the above algorithm is how to know that a particular sensor node has converged to its original position estimate, converged to a wrong position estimate or hasn't converged at

[2]The CrossValidation procedure is discussed in the next section.
[3]If $Y_p = \phi$ or contains only one element, we would randomly choose some elements from Y_{p-1}, so that we have enough spanning trees for the next phase.
[1]

all. We can answer one of the questions posed above by looking at the variance of the estimate. If the variance is less than some particular threshold we can easily conclude that we have convergence. The choice of the threshold depends on the noise in the environment. The other question is how would we know that a particular node that has converged to some estimate has actually converged to its exact position estimate. We need some sort of Cross Validation procedure to solve this problem.

The Cross Validation procedure would help us to rule out those "bad" sensors which show us that they have converged to some estimate but in fact they converge to a wrong one. The convergence to a wrong estimate can easily be justified by the fact that based solely on the distance information, if we don't have at least three nodes around some node we would end up getting the estimate anywhere on a circle (in case of one node around it) or it could be two points lying opposite to each other, as reflections across the line connecting the two nodes around it. The Cross Validation procedure is as follows:

If we have k unknown nodes converged to some estimate. We can compute a kxk distance matrix, K for these converged nodes based on their converged estimates. This distance matrix, K can then be compared with the original distance matrix, D at the corresponding entries to see which sensor pairs have the unmatched entries. We can remove the one which has most unmatched entries and then compute K again. In this way, all the "bad" sensors are removed one by one. This Cross Validation procedure removes all the "bad" sensors at the expense of at most two "good" sensors.

7 Example

Example 1 *An example demonstrating the algorithm on a network with fifty sensor nodes and only four anchor nodes.*

Note that in Figure 1(a), the anchor nodes are present in such a way that no three of them lie in the communication radius of any sensor node. Hence, the triangulation procedure cannot be carried out. An easy way to find out whether this condition holds is to find the radius of the circumscribed circle of the triangles formed by taking three anchors nodes as the vertices. If, for all the triangles, the radius of the circumscribed circle is greater then the communication radius the triangulation procedure would not be a solution. Figure 1(c) is a spanning tree with '$\diamondsuit$' as its root. Notice that the root is one of the anchor nodes. Figure 1(d) shows the network after the algorithm is run. As we mentioned before there would be some "bad" sensors which would converge to a wrong estimate and the answer to this problem was the Cross Validation procedure. The "bad" sensors which are marked by '$\times$' can easily be seen to be bad since they are not lying over any '$\bigcirc$'. The "good" sensors '$*$' are lying over '$\bigcirc$'. Also note that, the cross validation procedure has ruled out the 'bad' sensors at the expense of only one "good" sensor. With all the '$*$' and '$+$' in hand, we can now use the triangulation procedure to find the position estimates for the remaining sensor nodes. The '∇s' are the result of this triangulation procedure shown in Figure 1(e).

8 References

[1] M. J. Wainwright, T. S. Jaakkola, and A. S. Willsky. Tree-Based Reparameterization Framework for Analysis of Sum-Product and related Algorithms. In *IEEE Transactions on Information Theory*, pages 1120-1146, May 2003.

[2] Y. Shang, W. Ruml, Y. Zhang, and M. P. J. Fromherz. Localization for Mere Connectivity. In *Proceedings of the 4th ACM international symposium on Mobile Ad Hoc Networking and Computing*, pages 201-212. ACM press. 2003.

[3] P. Biswas and Y. Ye. Semidefinite Programming for Ad Hoc Wireless Sensor Network Localization. Technical report, Dept of Management Sciences and Engineering, Stanford University, Oct. 2003.

[4] D. Niculescu and B. Nath. Ad Hoc Positioning System (APS). In *IEEE GLOBECOM (1)*, pages 2926-2931, 2001.

[5] P. Bahl and V. N. Padmanabhan. Radar: An In-Building RF-Based User Location and Tracking System. In *Proc. IEEE INFOCOM 2000*, pages 775-784, Mar. 2000.

[6] http://www.uswcorp.com/USWCmainpages/our.htm

[7] N. Bulusu, J. Heidemann, and D. Estrin. GPS-less low cost Outdoor Localization for very small devices. In *IEEE Personal Communications*, pages 28-34, Oct. 2000.

[8] P. Bergamo and G. Mazzini. Localization in Sensor Networks with Fading and Mobility. In *IEEE PIMRC*, pages 750-754, Sept. 2002.

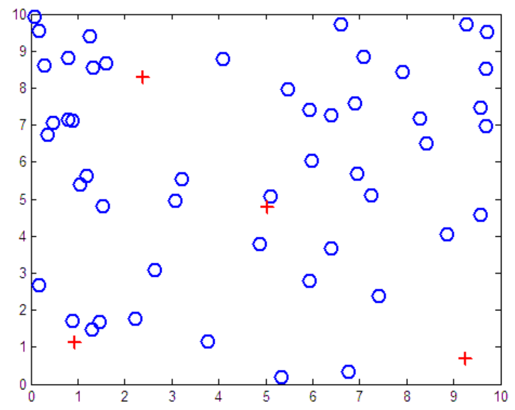
[9] R. J. Wilson, *Introduction to graph theory.* New York Academic Press, 1979.

[10] D. A. Bailey, *JAVA Structures.* McGraw-Hill Education, 1997.

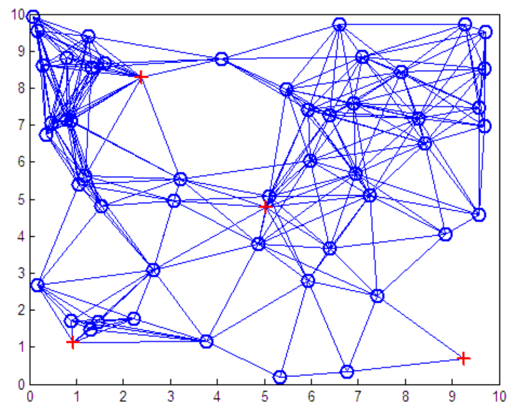
[11] J. Elson and K. Romer. Wireless sensor networks: A new regime for time synchronization. In *Proceedings of the FirstWorkshop on Hot Topics In Networks (HotNets-I)*, Oct. 2002.

[12] J. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. *Tech. Rep. UCLA-CS-020008, University of California Los Angeles*, May 2002.

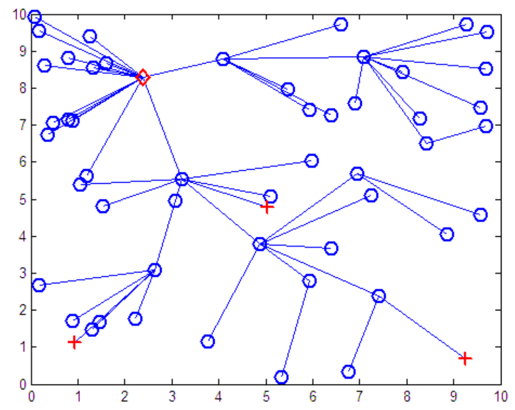
[13] D. Krishnamurthy. Self-calibration techniques for acoustic sensor arrays. *Masters thesis, The Ohio State University*, January 2002.



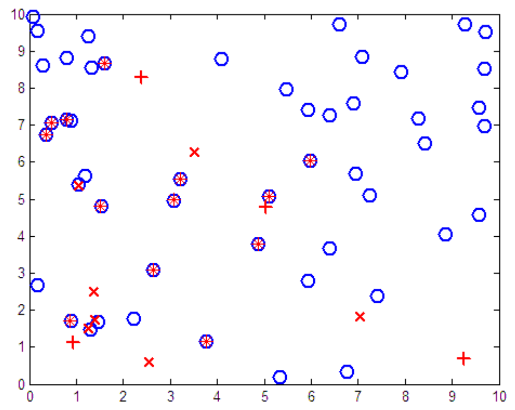
(a)



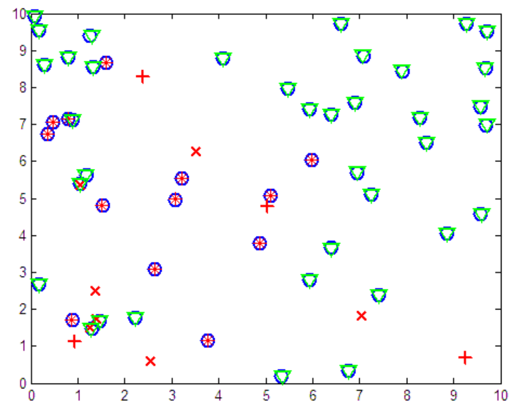
(b)



(c)



(d)



(e)

Figure 1: (a)The Network. (b)Graph of the Network. (c)A Spanning Tree with '$\diamond$' denoting the root of the tree. (d)Network after Algorithm is run. (e)Network after triangulation is performed. For all figures '$\bigcirc$' denote the sensor nodes, '$+$' denote the anchor nodes, '$*$' denotes the "good" sensor when the algorithm is run, '$\times$' denotes the "bad" sensors removed after Cross Validation and '∇' denotes the sensor whose position estimate is known by triangulation.