Refined Anchor Free Localisation for Wireless Sensor Monitoring in Wireless Sensor Networks

M. JAMES STEPHEN
Associate. Professor, Dept. of I.T
ANITS, Visakhapatnam, INDIA

P.V.G.D PRASAD REDDY
Professor, Dept. of CS & SE
Andhra University, Vizag, INDIA

SURESH CHITTINENI
Professor, Dept. of I.T
ANITS, Visakhapatnam, INDIA

Abstract

Localisation is required for many ad-hoc sensor network applications. In this paper we look at the limitations of the existing localisation technique, Anchor Free Localisation (AFL) with regards to coping with non-uniform anchor distributions and errors in ranging information. We present a refined approach called Refined Anchor Free Localisation (R-AFL) that uses a combination of mobile anchor scenarios for anchor information distribution, along with statistical techniques for performing localisation with inaccurate range data. This algorithm is a centralized incremental algorithm. In this algorithm we sue actual distance information instead of hop counts. Since hop counts does not reflect the true distances existing between them. Our algorithm stays in a base station and starts as soon as all the sensors send their local information to its base station.

Simulations with our refined approach have shown significant reductions (in the order of magnitude range) to the required processing for performing statistical localisation over previous attempts, as well as improving the generated location information in situations with non-total anchor information coverage, making possible a wider range of applications.

Keywords**:** *Wireless Sensor Networks, Anchor Free Localisation, Refined Anchor Free Localisation,*

1. Introduction:

Wireless sensor networks [1] are an increasingly attractive means to bridge the gap between the physical and virtual world. A WSN consists of large numbers of cooperating small-scale nodes, each capable of limited computation, wireless communication, and sensing.

Location information is important in many domains, hence various approaches have been proposed, of which some were even constructed and deployed on a large scale (e.g. GPS). Within the WSN community, specialized localisation algorithms have been developed that address the problems associated with the lack of infrastructure (i.e. GPS satellites) and the limited resources leading to incomplete and inaccurate information. A survey of initial approaches is presented by Hightower and Borriello in [5]; recent work includes [6], [7], [8], [9] and [10]. With WSN localisation,

Monitoring applications define an important class of Wireless Sensor Networks (WSNs). In these applications, the network perceives the environment and searches for Events occurrences (phenomena) by sensing different physical properties, such as Temperature, humidity, pressure, ambient light, and movement. In such cases, location in Time and space of both phenomenon and nodes are usually required fro tracking and Correlation purposes. In the former cases, node's clocks are synchronized, while in the Latter, node's physical location (e.g., latitude, longitude) are discovered. The need forSuch information in WSNs and the proximity and similarity of both problems suggest they can be addressed as a single problem, which we call the Localization in Time – Space. In this thesis, we propose a time-space localization system (R-Synapse) for Sensor networks. Simulation results show that the proposed system is able to synchronize The nodes' clocks immediately (within few microseconds) after locating their global Positions. The major advantage of the proposed system is that it reduces the Communication cost by eliminating the GPS receivers. It is completely an Anchor- Free Localization system.

Localization Problem: Give a multihop network, represented by a graph G=(V,E), and a mechanism through which each node can discover its neighbouring nodes by establishing communication with these nodes, and can estimate the range to each of its neighbours, we want to find the global position (x, y) of each and every sensor in the network.

AFL (Anchor-Free Localisation) [2] is a fully decentralized algorithm, where the nodes start from a random initial coordinate assignment and converge to a consistent solution using only local node interactions. The key idea in AFL is fold-freedom where nodes first configure into a topology that resembles a scaled and unfolded version of the true configuration, and then run a force-based relaxation procedure.

An example of localization system is AFL (Anchor-Free Localization) [2], Proposed by Nissanka B.Priyantha, Hari Baladrishnan, Erik Demaine, and Seth Teller. AFL is a fully decentralized Algorithm where nodes start from a random initial Coordinate assignment and converge to a consistent solution using only local node Interactions. Each sensor in the network runs AFL locally, interacting only with Neighbouring nodes, such that after a number of iterations all sensors will reach a Consensus about their coordinates in some coordinates system. By doing this in an Automated manner, large scale sensor networks can eliminate the cumbersome and unscalable process of manually configuring sensor nodes with their location.

But the AFL has so many pitfalls in it.

The following are the some of the pitfalls observed in the AFL algorithm. Unfortunately, these relaxation techniques are quite sensitive to initial starting positions. Bad starting position will result in Local Minima. The insight is that the network gets tangled and that using the spring model style optimization is unable to fully untangle the network.

- The AFL Algorithm proceeds with the consideration of the node with smallest ID. This may not give all the time the centre of the graph.
- Major problem with AFL Algorithm is that it is a concurrent algorithm. All the nodes calculate and refine their coordinate information in parallel. If one node goes wrong direction almost all other nodes goes in that wrong direction.
- The AFL works completely in decentralized manner. That is, each sensor in the network run AFL algorithm locally, interacting only with neighboring nodes and after number of iterations all the sensors will reach consensus about their coordinates in some coordinate system. To run AFL algorithm at each and every sensor in the network we have to incorporate high configured CPUs in each and every sensor of the network, which is a cost effective proposition.
- In the second phase, AFL uses mass-spring based optimization to correct and balance localized errors. But, the fundamental problem with mass-spring optimization is that it has a high probability of converging to local maximum. If the graph obtained in the first phase is globally rigid or fold-free graph then mass-spring optimization work properly. But AFL doesn't assure that it can produce a globally rigid or fold free graph in its first phase.

In the later sections, we propose an anchor-free localization system R-AFL which is a refinement of AFL algorithm, which eliminates almost all limitations that occur in' AFL algorithm.

Synchronization Problem:
Given a multihop network, represented by a graph G = (V,E), and a mechanism through each node can discover its neighbouring nodes by establishing communication with these nodes, and can estimate the range to each of its neighbours we want to find the time $t_u(t) = d_u t + o_u$ for all unsynchronized and unknown nodes u, where d_u is the drift and o_u is the offset.

In many aspects, the characteristics of the synchronization problem are similar to Those of the localization problem. In a sensor network, most of the application that required position information. If we look closer at the solutions to those problems, we can detect a number of similarities. For examples, localization systems can be

divided into components wherein each one is responsible for solving a single piece of the localization problem. Clearly, the synchronization system can also be divided into corresponding components.

The need for both time and space information, and also the similarities between these tow problems has shown the importance of combining them into a single one: the Localization in Time-Space. By doing so, we save energy and network resources, and also have the opportunity to improve time and position estimations in contrast to the scenario in which these problems are solved separately. For instance, synchronization algorithms can take advantage of techniques and resources used by the localization algorithms, and localization algorithms can take advantage of techniques and resources used to synchronize neighbour nodes.

Then, the localization in time-space problem can be stated as follows:

Localization in Time-Space Problem

Given a multihop network, represented by a graph G = (V,E), and a mechanism through which each node can discover its neighbour nodes by establishing communication with these nodes, and can estimated the range to each of its neighbours, we want to find the position $(x_u\ y_u)$ and time $t_u(t) = d_u t + o_u$ for all unsynchronized and unknown nodes u, where d_u is the drift and o_u is the offset.

Recently, the both problems have been discussed in conjunction. Initially, Romer addressed and solved these problems separately. Then Romer and Matter presented both problems as related to each other, but no integrated solution is proposed. To the best of our knowledge there is only one algorithm called Synapse (SYNchronization And Positioning for Sensor networks) [3] proposed by Horacio A.B.F. de Oliveira, Eduardo F.Nakamura, and Antonio A.F.Loureiro, which gives an integrated solution to these two problems. But as we mentioned in the previous section Synapse has so many pitfalls in it. In the later sections, we propose an anchor-free localization system R-Synapse which is a refinement of Synapse algorithm, which eliminated almost all limitations that occur in Synapse algorithm.

2. Complexity of the Problem.

Give an abstract graph with a specified length (positive real number) for each edge, when can the graph be embedded into 2d OR 3D while satisfying the edge lengths? When is such an embedding unique (up to global translation, rotation, and reflection), and therefore a reliable reconstruction of the desired geometry? Both of these questions have received considerable attention in both the discrete geometry and computational geometry communities.

Deciding whether a graph with edge lengths can be embedded is NP-hard in general. Basically, triangles from rigid structures but can be independently flipped (folded), and deciding whether a string of triangle can be folded left and right to make a particular length is equivalent to subset sum. Saxe proved the stronger result that the problem is strongly N P-hard even for embedding into ID.

A graph with specified edge length which has a unique embedding is called globally rigid, a variation on the well-studied concepts in rigid theory. Because global rigidity can be expressed as the uniqueness of a solution to a system of algebraic constraints (specifying distances between some pairs of vertices), global rigidity is almost always a property of the underlying graph, not the specific edge length. A graph (without edge lengths) is generically globally rigid if, for almost any realizable assignment of length to the edges, it is globally rigid.

Hedrikson showed that, for a graph to be generically globally rigid in d dimensions, it must be (d+1)-connected and the removal of any edge must leave the graph “generically rigid.’’. Both of these properties can be checked in polynomial time. Connelly proved that these tow properties are not enough: they do not imply generic global rigidity in 3D. However, Hedrikson conjectures that these two properties are enough, exactly characterizing generic global rigidity in 2D.

sEmbedding a graph with given edge length also arises in the context of reconstructing the geometry of molecular structures in an area called distance geometry. In this context, distance measurements are substantially less accurate, and several techniques have been developed to refine estimates and reduce error bounds by combining several constraints. On the algorithmic side, Berger et al., give efficient algorithms for embedding a graph ;with error-prone edge lengths, even when nearly half of the edges might have completely inaccurate lengths. However, these algorithms rely on every node having a constant fraction of the nodes as neighbours, for a total of $\Omega(n^2)$ links between n nodes, which does not scale in our context.

3. Refined Anchor-Free Localization Algorithm.

In this section we propose a localization algorithm called R-AFL (Refined Anchor-Free Localization). This is a centralized incremental algorithm. In this algorithm we have used RSSI or local node-to-node distance information. The algorithm produces the graph which is a combination of reflection, translation, and rotation of the original embedding.

3.1 R-AFL Algorithm.

The R-AFL Algorithm proceeds in tow phases. In this algorithm we sue actual distance information instead of hop counts. Since hop counts does not reflect the true distances existing between them. Our algorithm stays in a base station and starts as soon as all the sensors send their local information to its base station. By using this local distance information of all these nodes it calculates the global information of the distance matrix for the whole topology. i.e., if form a node n_i to n_j, if there is an edge (if they are neighbours) it puts its distance information as $dist_{ij}$, else it puts a huge number to indicate the infeasibility of the connectivity. At this stage we have a distance matrix which represents the distance information between the neighbours of the whole graph.

After getting this global information the first phase of our algorithm starts by using this information and finds the node coordinates of all the nodes and sends back this information to all the sensors. Base station stores coordinates along with the node ids' and propagate them to the sensors. These sensors store this information by comparing with this node ids.

3.2 PHASE-1

The goal of the first phase of AFL algorithm is to embed the graph structurally similar to the original embedding. More specifically, the algorithm tries to avoid folds in the resulting graph compared to the original graph. We formally define a fold-free embedding of graph; to be one where every cycle of the embedding has the correct clockwise/counter-clockwise orientation of nodes, modulo global reflection, which respect to the original graph. We start with some terminology and assumptions. We assume that each node has a unique identifier; the identifier of node i is denoted by ID_i. The algorithm uses the phrase hop-count between nodes i and j to mean the number of nodes h_{ij} along the shortest radio path between nodes i and j . In practice, this heuristic works on a neighbour graph that assumes only radio connectivity , Without using accurate ranging information from other technologies like ultrasound.

The algorithm first elects five reference nodes. Four of these nodes n_1 n_2 n_3 n_4 are selected such that they are on the periphery of the graph and the pairs (n_1, n_2) is roughly perpendicular to the pair (n_3, n_4). The node n_o is elected such that it is the "middles'' of the graph. By using the distance matrix we shall find the adjacency list of the graph. From adjacency list find the Adjacency matrix of the graph and let it be A. Apply Floyds algorithm on A and the path matrix of the graph which represents the minimum hop counts in between any two nodes of the graph. Find the eccentricity of each and every node in the network. This can be done, by choosing the maximum value in each row.

- Step-1. Select reference node n_o such that the eccentricity of n_o is the least among all others. Any ties are broken using the node's ID.
- Step-2. Select the reference node n_1 to maximize $h_{0,1}$ i.e., n_1 is a node that is the maximum hop-count away form node n_o. Any ties are broken using the node's ID.

- Step-3. Select reference node n_2 to maximize $h_{1,2.}$ Any ties are broken using the node's ID.
- Step-4. Select reference node n_3 to minimize $|h_{1,3} – h_{2,3}|$. In general, several nodes may all have the same minimum value, and the tie-breaking rule is to pick the node that maximizes $h_{1,3} + h_{2,3}$ from the contenders. This step selects a node that is roughly equidistant from nodes n_1 and n_2 and is 'far away'' from n_1 and n_2
- Step-5. As in the previous step, select reference node n_4 to minimize $|h_{1,4} – h_{2,4}|$. Now, break ties differently: from among several potential contender nodes, pick the node that maximizes $h_{3,4.}$ This optimization selects a node roughly equidistant form nodes n_1 and n_2 while being farthest from node $n_{3.}$

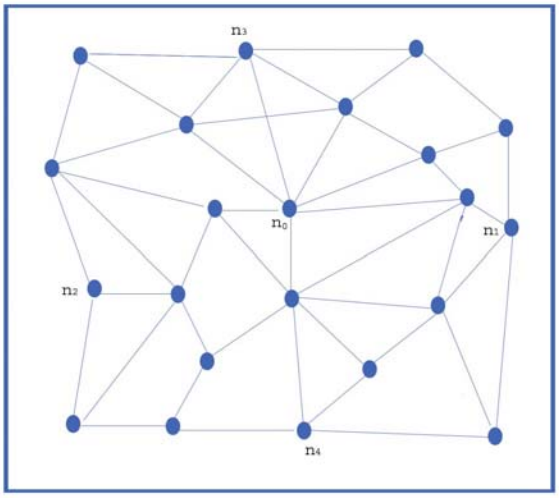

Figure3.1 Finding the Reference points

Then for each node $n_{i,}$ use the hop-counts $h_{o,i}$, $h_{1,i,}$ $h_{2,i,}$ $h_{3,i}$ and $h_{4,i}$ from the chosen reference nodes to approximate the polar coordinates (P_i , θ_i).

$$P_i = h_{o,i} * R$$

$$\Theta_i = \tan^{-1} (h_{1,i} – h_{2,i} / h_{3,I} – h_{4,i})$$

Here R is the maximum radio range. This coordinate assignment roughly approximates the rule layout of the graph, especially for graphs that ' radiate out'' from a central point. When calculating $P_{i,}$ the use of range R to represent one hope-count result in a graph which is physically larger than the original graph; this property of the graph helps avoid local minima during the optimization phase.

3.3 PHASE-11

This phase is an optimization phase. In this phase we optimize the node coordinates incrementally, so that all the nodes are configured to the positions so that they satisfy inter node distances to their respective neighbours. In this phase the algorithm proceeds as follows:

- Step-1. We maintain a neighbourhood queue Q_1 in which we store the centre of the graph n_o in the queue. After that we put neighbours of the centre in the queue, and then their neighbours and we proceed so on until all the nodes' are exhausted in the graph
- Step-2. Repeat Step-3 until all the elements of Q_1 are exhausted.
- Step-3. Delete nodes from $Q_{1,}$ and for each such element, take another queue Q_2 and insert all the neighbours of the element that popped from $Q_{1.}$

- Step-4. Repeat Step-5 until all elements of Q_2 are exhausted.
- Step-5. Delete each node from Q_2 and move it infinitesimally and compare the actual distance to the obtained distance by its position. The movement of the node should be in such a way that the difference should decrease in the subsequent moves

The topology we get is unique up to its translation, rotation and flipping. As the algorithm is incremental the error propagates and due to that the error between the nodes at the outer periphery of the graph is relatively more when compared to the nodes around the centre.

4. Conclusion

Many sensor network applications require that each node's sensor stream be annotated with its physical location in some common coordinate system. Manual measurement and configuration methods for obtaining location don't scale and are error-prone, and equipping sensors with GPS is often expensive and does not work in indoor and urban deployments. Sensor networks can therefore benefit from a self-configuring method where nodes cooperate with each other, estimate local distances to their neighbors, and converge to a consistent coordinate assignment.

In this paper we looked at the limitations of the existing localisation technique, Anchor Free Localisation (AFL) and we presented a refined approach called Refined Anchor Free Localisation (R-AFL) that overcome almost all the limitations of the existing AFL technique for localisation. This algorithm is a centralized incremental algorithm and we have used RSSI or local node-to-node distance information. The algorithm produces the graph which is a combination of reflection, translation, and rotation of the original embedding.

References

[1] I. F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. Wireless Sensor Networks: A Survey. *Computer Networks*, 38(4):393–422, March 2002.

[2] NISSANKA, B., PRIYANTHA, HARI BALAKRISHNAN, ERIK DEMAINE, AND SETH TELLER. Anchor-Free Distributed Localisation in Sensor Networks. Tech Report 892, MIT Laboratory for Computer Science April 15, 2003.

[3] Horacio A.B.F de Oliveira, Edurado F. Nakamura, Antonio A.F. Loureiro and Azzedine Boukerche, Localization in Time and Space for Sensor Networks, 21st International Conference on Advanced Networking and Applications (AINA'07), 2007 IEEE.

[4] Tom Parker_ Koen Langendoen Faculty of Electrical Engineering, Mathematics, and Computer Science Delft University of Technology, The Netherlands "Refined Statistic-based Localisation for Ad-Hoc Sensor Networks"

[5] J. Hightower and G. Borriello, "Location systems for ubiquitous computing," *IEEE Computer*, vol. 34, pp. 57–66, Aug. 2001.

[6] N. Bulusu, J. Heidemann, V. Bychkovskiy, and D. Estrin, "Densityadaptive beacon placement algorithms for localization in ad hoc wireless networks," in *IEEE Infocom 2002*, (New York, NY), June 2002.

[7] D. Niculescu and B. Nath, "Ad-hoc positioning system," in *IEEE Globe- Com*, pp. 2926–2931, Nov. 2001.

[8] C. Savarese, K. Langendoen, and J. Rabaey, "Robust positioning algorithms for distributed ad-hoc wireless sensor networks," in *USENIX technical annual conference*, (Monterey, CA), pp. 317–328, June 2002.

[9] A. Savvides, H. Park, and M. Srivastava, "The bits and flops of the n-hop multilateration primitive for node localization problems," in *First ACM Int. Workshop on Wireless Sensor Networks and Application (WSNA)*,(Atlanta, GA), pp. 112–121, Sept. 2002.

[10] M. Sichitiu and V. Ramadurai, "Localization of Wireless Sensor Networks with a Mobile Beacon," Tech. Rep. TR-03/06, Center for Advances Computing and Communications (CACC), Raleigh, NC, July 2003.