# Adumbrate: Motion Detection with Unreliable Range Data

Tom Parker* Koen Langendoen
{T.E.V.Parker, K.G.Langendoen}@tudelft.nl
Faculty of Electrical Engineering, Mathematics, and Computer Science,
Delft University of Technology, The Netherlands
(* Supported by the Dutch Organisation for Applied Scientific Research (TNO))

***Abstract*— Most current work in Wireless Sensor Networks deals with applications running on static networks, along with some localisation requirements, but without any motion detection hardware. However, many of these applications require some level of motion detection, if only to notice the cases when a network ceases to be statically located and starts to have moving nodes. As most of the currently used application scenarios rely on the assumption that motion will not happen, if a node does move it will cause significant amounts of damage to any protocols relying on this static assumption e.g. routing, localisation, aggregation, etc.**

**In this paper we look at methods for detecting moving nodes, using only RSSI data, including an anchor-less solution to ensure that we can always detect motion. Our methods are intended to work in co-operation with existing static network localisation algorithms.**

## I. Introduction

Many possible applications have now been thought of for Wireless Sensor Networks (WSNs), and a significant number of them rely on location information in order to perform their designated function. This is mainly because the main purpose of a WSN is information gathering, and gathered data is only useful if you know what it applies to. For example, the data "the temperature has gone up by 10 degrees" is not very useful, but the information "the temperature has gone up by 10 degrees in room 3C" is a lot more interesting. Location information gives us a context, which allows us to actually use our gathered data. For example, monitoring room temperature can be used to control when to switch air-conditioning systems on and off. When detailed location information is present, it might even be possible to personalise working conditions within a shared office (i.e. individual settings per cubicle). 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).

The range of viable localisation techniques depends heavily on what node hardware is available. At one end of the scale, if every node has accelerometers, GPS, and an array of accurate ultrasound sensors, then localisation is quite simple. Alternately, nodes can have no hardware designed for motion detection or localisation at all, and only RSSI data from a radio to give limited ranging information. Unfortunately, most node hardware is of the latter type. Of course, more hardware can always be added to a node, but cost factors (both in monetary price/node and energy costs) will tend to reduce the likelihood of fully equipped nodes. Within the WSN community, specialised localisation algorithms have been developed that address the problems associated with little to no infrastructure (i.e. access to GPS satellites) and limited resources leading to incomplete and inaccurate information. A survey of initial approaches is presented in [4]; recent work includes [1], [8], [12], [13] and [15]. This work however generally deals with static networks, and detecting when a network is no longer static with minimal additional hardware requirements would be of considerable use.

With WSN localisation, nodes with additional hardware are referred to as "anchor" nodes i.e. they have a reliable source of information about their location. Many localisation techniques rely on anchors, and on the assumption that anchor nodes are uniformly distributed among a uniform distribution of non-anchor nodes. Given the small percentage (<10% in most scenarios currently postulated) of anchors within a large collection of non-anchors, and the aim that sensor networks are eventually intended to be easy to distribute for non-computer scientists, this assumption can not be relied on for many application scenarios. As we cannot rely on the existance of anchor nodes, we need to be able to detect motion even without anchors.

Another major problem within WSN localisation techniques is acquiring accurate range information between pairs of sensor nodes. This can be done in a variety of ways, ranging from simple techniques like Radio Signal Strength Indication (RSSI), time of flight data for various sensor types (e.g. ultrasound), to more complex ideas like time of flight difference (which measures the difference between two incoming signals travelling at different speeds). In each case, there is generally some error in the ranging information, which motion detection algorithms must be aware of and be able to work with.

In this paper we focus on two forms of scenario - what can be done with just basic nodes (no localisation hardware; just RSSI); and what can be done with minimal quantities of additional hardware on a limited set of nodes (anchor nodes). With only basic nodes we are limited as to what we can do, but some information can still be gathered. In the situation with a limited set of anchor nodes, we still may well have the same problem as with just basic nodes, as with low percentages of anchor nodes, a given basic node may well have no communication with anchor nodes. One solution to the lack of additional anchor nodes is that the anchors may well be mobile ([9], [15]), and so even if a basic node has no current communication with anchor nodes, gathering some information before communication is established with anchor nodes may help determine earlier location data.

One piece of information that would be very useful is motion information - if a node has not moved between its

initial deployment and the time it is fully localised, then we know that all data gathered up until that point was from a particular location. If it has moved, information on the approximate amount of motion may help decide whether the data can still be treated as located at a particular point (with a particular level of location accuracy). Our particular focus here is on allowing smarter decisions in limited motion scenarios for localisation algorithms designed for static networks, and limiting the problems that moving non-anchor nodes can cause to stateful localisation techniques.

## II. Non-anchor node motion

If a node has been localised, and then moves without being aware of its movement, then the node will be somewhere other than where it thinks it is. If it then broadcasts its old location data, while being at the new location, then other nodes in the network will have inconsistent information. This is only a problem with non-anchor nodes, as when anchor nodes move to a new location, they will have new location data, and in both cases their true and calculated locations are the same (to within a known degree of accuracy).

In order to help formulate solutions to this problem, we will firstly examine what can be done in networks with anchor nodes in order to detect motion. Secondly, in the event that a node currently has not received any anchor information from the network, because of a current local lack of anchors, then we need to be able to find alternate ways to do motion detection. We need to be able to do this because there may be data that the sensors need to gather before anchor information is available, and so we need to be able to work out whether they have moved since the data was gathered. Anchor-less situations are likely in the early stages of some mobile anchor scenarios, especially when the placing object is far away from the locations where the nodes are being placed.

Unfortunately, most methods for detecting movement of nodes can not tell the difference between moving nodes and malicious nodes (nodes that are sending bad data). Malicious nodes are hard to deal with - with a large enough amount of effort and/or nodes, a malicious intruder can potentially break an entire network. However, for most non-military sensor network scenarios, the chances of a malicious intruder are very low, whereas motion is likely. We are therefore going to concentrate our efforts on detecting motion, and leave the problem of dealing with malicious intruders for more advanced systems.

## III. Bounding boxes

Anchor nodes periodically broadcast their locations, and if a node has received location data from an anchor then it knows it is in radio contact with that anchor, and so therefore it must be within radio range (where "radio range" is a maximum possible value including "gray area" [18] effects) of that anchor. Thus we can limit the space of possible locations for that node to a circle centred on the anchor's location with radius equal to the radio range. Bounding region information can therefore be used to sanity-check information from localisation algorithms.

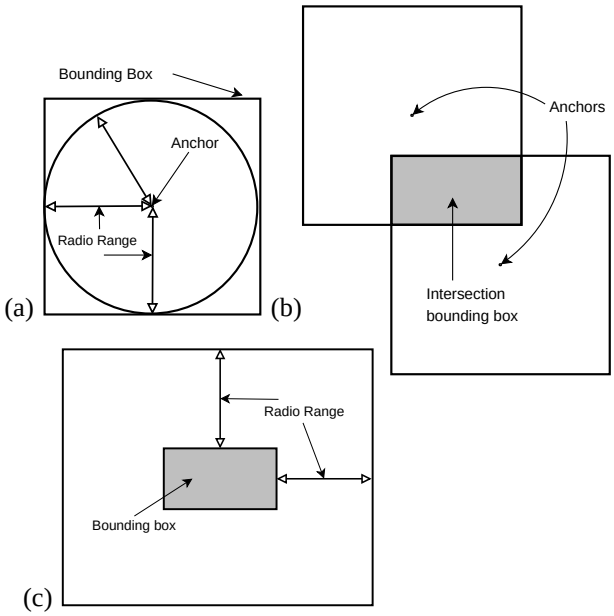


Fig. 1. Bounding Boxes

For practical purposes (significant speed improvements) we use a bounding box rather than a circle, with each side equal to $2 * radio\,range$, and the anchor in the centre (Figure 1a). (The basic concept of bounding boxes has previously been analysed in earlier work [9], [16]). This results in a larger region, but we still have the guarantee that all possible locations for the node are located within the box, while keeping the box size to a minimum. This currently assumes a circular radio model, but for radios with non-circular transmission spaces, we can calculate the minimum box that contains the entire possible transmission space, and so be still able to use this methodology.

When a node receives location information from an additional anchor, it knows that it must be within the bounding boxes for both anchors. Therefore, we can reduce the bounding box for the node to the intersection of both of these boxes (Figure 1b, Algorithm 1 on the facing page). A bounding box is defined by two points, its Top-Left and Bottom-Right corners.

An additional source of bounding boxes that we can use are from "pseudo-anchor" nodes. A pseudo-anchor node is a non-anchor node that has a reasonable level of confidence in its location (for example, because it has a bounding box that is no larger than a suitable threshold), and therefore the cost of transmitting its bounding box is worthwhile given the likely improvements to other nodes' bounding boxes. The transmitted bounding box for a pseudo-anchor node is its bounding box expanded by $radio\,range$ in each direction (Figure 1c). This creates a larger box than for anchor nodes, but especially in low anchor node density scenarios, pseudo-anchor boxes provide another useful source of bounding box information, while maintaining the guarantees regarding the node location always being within the bounding box.

### A. Breaking the Boxes

As a consequence of the sanity condition that a node's bounding box will always contain its true location, and that any two nodes that are in communication must be

**Algorithm 1** Bounding boxes

Abbreviations used here: TL = Top-Left corner of a bounding box, BR = Bottom-Right corner, R = Maximum possible radio range between a pair of nodes, including "gray area" [18] effects.

1) Initially, the bounding box for a node is set to $[(-\infty,\infty)\times(-\infty,\infty)]$.
2) As anchor information comes in, the bounding box for this node is intersected with the existing bounding box (see Figure 1 for examples of bounding boxes, including a diagram of this step in Figure 1b)
$NewBox(TL,BR) = Max(Anchor_{TL_y} - R, OldBox_{TL_y}))\times$
$(Min(Anchor_{BR_x} + R, OldBox_{BR_x}),\ Min(Anchor_{BR_y} + R, OldBox_{BR_y}))]$

within radio range of each other, bounding boxes assume another sanity condition - that the current bounding box of a node and another bounding box that it has received, and therefore wishes to intersect with, will always have a non-empty union.

Figure 2 is an example of how motion of a node can break bounding box sanity. $A1$ and $A2$ are the locations of a moving node $A$ before and after it moves, and $B$ is a stationary node. The inner and outer boxes around the nodes represent their bounding boxes and bounding boxes expanded by radio range, respectively. If $A$ talks to $B$ when it is at $A2$, and $A$ thinks it is still located at position $A1$, then there will be an inconsistency between $A$'s bounding box and the bounding box of $B$, which means that one of the two nodes must have moved. In a number of cases we will not be able to detect motion (the maximum allowable motion before we can detect motion with absolute certainity is proportional to the size of the bounding box of a node), but in these cases we do maintain bounding box consistency, so we can still generate valid bounding boxes, although with a reduced accuracy due to the size of the boxes.

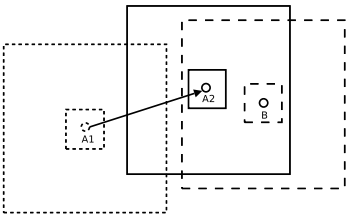

Fig. 2. Motion example

When we do detect bounding box inconsistencies, we can work to correct the problem. If a node $N$ receives a new bounding box from a neighbour $M$ that would create an inconsistent situation ($Box_N \cap Box_M = \oslash$), then this tells us that either that $N$ or $M$ has a problem. Both nodes then check how many of their neighbours currently consider them inconsistent. If two neighbours (including either $N$ or $M$) consider one of $N$ or $M$ currently inconsistent, then that node should recalculate its bounding box information. This is done by discarding all current bounding box data (i.e. returning the node to Step 1 of Algorithm 1), and sending a control packet to all of the neighbouring nodes saying that any currently used bounding box information from that node should be discarded, and requesting their current bounding boxes.

Figure 3 shows how this could work for a node $A$ moving from $A1$ to $A2$. It starts to communicate with nodes $B$ and $C$, and there is an inconsistency between the box $A1$ and the boxes for $B$ and $C$, so there is an inconsistency "link" from $A \leftrightarrow B$ and from $A \leftrightarrow C$. As two of $A$'s neighbours consider it inconsistent, it resets its bounding box data back to the startup configuration, and sends a control packet to $B$ and $C$ invalidating any bounding box data they have gained from $A$, and requesting their bounding boxes. This would then result in a new and valid bounding box for $A$. Any localisation algorithms being run on the node should also possibly be notified at this point if the previously determined location for the node is now outside the new bounding box.

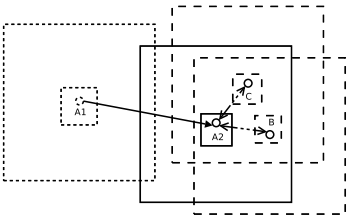

Fig. 3. Inconsistency

In many of the possible scenarios for bounding box inconsistency, the problems will now be resolved, and the node will have a new bounding box. If however, this fails, then the node should send a message to its neighbours declaring that it currently considers them inconsistent, and remain in an inconsistent state. The inconsistent node should now stay in that state until there is a change in any of its neighbours' bounding boxes, in which case the bounding box for this node should be re-evaluated to check for the resumption of consistency.

One problem here is that $B$ and $C$ may have previously integrated $A$'s information into their bounding box configurations, and if $A$'s information is later found to be invalid, then $B$ and $C$ need to be able to work out what parts of their bounding boxes are due to $A$ and what are due to other nodes. In order to counter this, each node can keep a record of the bounding box for each other node, in order to be able to rebuild an accurate bounding box when one node's information is found to be invalid. If a node resets its bounding box information due to detected inaccuracies, then the node also discards the list of bounding boxes that it had stored as well. To deal with mobile situations where there are many various sources of anchor information, and the storage of every other recieved node would be impractical, then only a limited set ($N$ most recent recieved boxes) are stored, in addition to the calculated box for the node in question. Discarding some recieved boxes after they have been used to improve the local box does reduce our capability to handle inconsistent boxes due to motion, but given the limited storage available to WSN nodes, this is a reasonable trade-off.

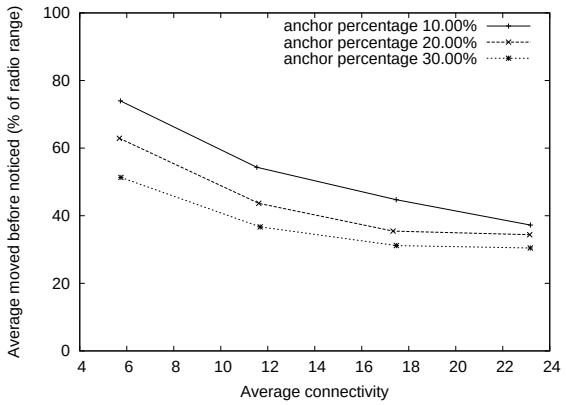

Fig. 4. Bounding box testing

### B. Results

We performed a series of experiments, testing how much motion was necessary before bounding box inconsistencies were noticed. The nodes were scattered in a 200 x 200 box, with radio range set to 14. We varied the number of nodes to get different levels of average connectivity in the network, as well as designating a percentage of the nodes as anchor nodes. For each simulation run, we allowed the box sizes to stabilise, and then started to move one of the nodes in a random direction. Each experiment was run 20 times, with varying random seeds for each configuration, and the results given here are an average of the 20 sets for each configuration.

The graph in Figure 4 shows the minimum motion necessary before inconsistency checking notices motion. The minimum motion necessary for detection reduces with higher connectivity networks, as well as with increasing anchor percentages. For most scenarios, the amount of motion necessary does not in general exceed *radio range*. Additionally, all of the experiments reported a zero false-positive rate i.e. no node reported as moving was in fact stationary.

## IV. Anchor-free mass-spring motion detection

For the problematic case where we have not yet received any information from anchors, localisation becomes much more difficult. We can however use anchor-free localisation to build a local co-ordinate system, which can be used to detect moving non-anchor nodes and record their relative motion. The motion information can later be translated from the local co-ordinate system to a global system once anchor information is available.

For motion detection to be possible however, we need a way to build local co-ordinate systems in the absence of accurate range information. We cannot build bounding boxes due to the lack of anchors to initialise the algorithm, and so we turn to mass-spring models (Appendix A) for the node locations instead. Mass-spring models require more calculations than bounding boxes, but in the absence of anchors, mass-spring models are still a viable option.

### A. Motion detection

This is a simplified overview of our motion detection algorithm. For full details see the appendices. The node that is running this algorithm is referred to as the "root" node. In order to do motion detection, we first need a method to build local co-ordinate systems:

1) Gather range data (estimated values and variances from the radio model) from the root node to its neighbours, and also query the root's neighbours for range data to their neighbours, giving us a topological map for all of the root's 1- and 2-hop neighbours. We can then place the root node, and one of the root node's neighbours (Appendix C).
2) Working from these initial two nodes, we can now start to find initial locations for the other nodes. We can place all nodes that have two neighbours in the already placed set of nodes, using those two neighbours (*A* and *B*, referred to as the "parent" nodes of our new node) and the ranges between them to place our new node *C* (Appendix D). In some cases, we will have chosen parent nodes that are unsuitable for placing *C*, and in these cases the algorithm will fail the sanity tests specified in Appendix A. If this is the case, we then proceed to check other possible parent node pairs for suitability as per Appendices D and E.
3) The locations for the nodes are now further refined (Appendix F). Refinement is necessary because our initial configuration does not take into account all of the links (Appendix B) between nodes when we are placing them.

Now that we can build a local co-ordinate system, motion detection is possible by comparing a local co-ordinate system generated at one moment in time ($LCS_1$) by a node, to another generated system by the same node at a later point in time ($LCS_2$). We require at least 2 nodes common to both systems (which may or may not be neighbours), in order to be able to use this information, otherwise we cannot work out which way the node moved.

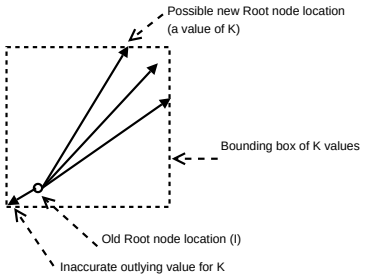

Fig. 5. Calculating values for K

For each pair of nodes common to both co-ordinate systems (*A* and *B*), and using the $LCS_1$ system co-ordinates for *A*, *B* and our root node (marked as *I*), as well as range data from $LCS_2$ for our root node relative to *A* and *B* we can calculate the set of possibilities for the location of the $LCS_2$ root in $LCS_1$ (Appendix G, Figure 5).

We now have a set of up to 4 possible locations for *K* which are checked against the measured $R_{K,A}$ and $R_{K,B}$ values. The values that have correct ranges (at most two of them, by standard geometrical theory regarding the intersection of two circles [17]) are valid locations for *K*, and we choose the closest to the existing root node, as the movement between separate invocations of this algorithm should be minimal.

Each of the valid $K$ locations represents a "motion vector" ($MV$) for our root node. We can calculate values for $MV$ using the locations of $K$ as the vector between $I$ and $K$, as in the event of no changes, $K = I$, and $I$ is at $(0,0)$ by the definition of $I$ being the origin of the local co-ordinate system. The average of the values for $MV$ is the assumed motion, and the maximum $K$ values in each direction gives us a bounding box whose area is proportional to the inaccuracy in our $K$ measurement.

### B. Results

We performed a series of experiments to test anchor-free mass-spring motion detection, starting from a randomly generated set of "true" node locations, using 226 nodes in a 100x100m area, with a radio range of 14m, giving an average connectivity of approximately 12.

Experimental tests [11] have shown that the change in the error between consecutive measurements for the range between a pair of static nodes, will be significantly smaller than the error between the measured ranges and the true range. This is because many of the sources of range inaccuracy (reflections, batteries running down, low-quality radios, etc) should be relatively stable between one range measurement and the next. We therefore setup our experiments to mimic this, by taking the topology and ranging information from the "true" locations, and adding some gaussian distributed noise to the ranging data (mean equal to the "true" range, variance at different levels for different experiments). This "noisy" ranging information was then used to generate a local co-ordinate system (Appendices C-F). We then moved the root node by a random amount (uniformly random direction, distance depending on the experiment). For all the links not connected to the root node, we changed their "noisy" ranges by a small random value (mean equal to the original "noisy" range, variance at different levels for different experiments), and for the links connected to the root node, we re-generated new "noisy" ranges according to the true ranges for the new root node location (noise generated with the same parameters as the first local co-ordinate system). This second set of "noisy" data was then used to generate another local co-ordinate system, and the two were compared as per Appendix G.

For all of the experiments, the results are specified as percentages of the radio range, and are averages of 20 runs of a particular set of parameters, using a different random seed each time. Figures 6, 7 and 8 show the results with inaccuracy for the non-moved links set to 0%, 5% and 10% of the original variance. The 6 lines on each of the graphs represent a variety of movements of the root node between the first and second sets of data. At 10% and 20% motion, neither altering the original error nor the second measurement inaccuracy significantly changes the results, and the percieved motion is reasonably accurate ($\pm$3%). However with greater motion (>20%), the percieved motion becomes increasingly inaccurate. Note that this is the motion between successive tests of the motion detector, and so if we run the algorithm frequently

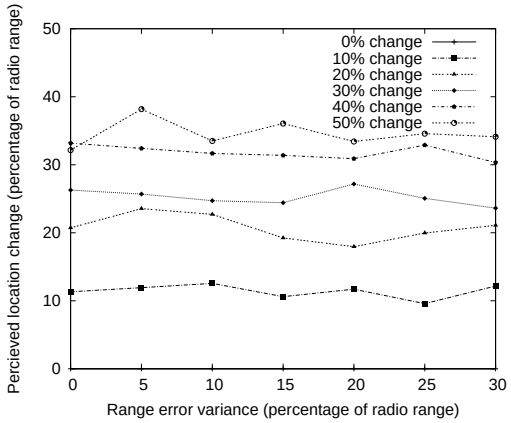


Fig. 6. 0% inter-measurement inaccuracy

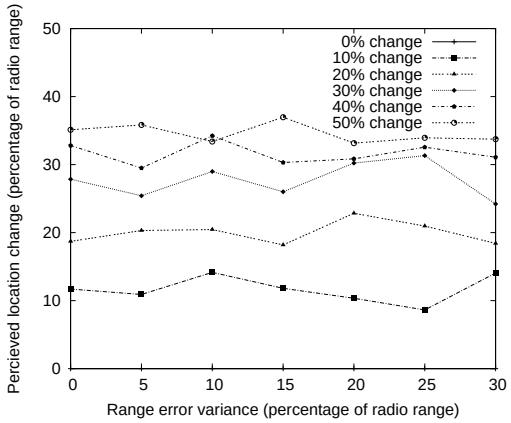


Fig. 7. 5% inter-measurement inaccuracy

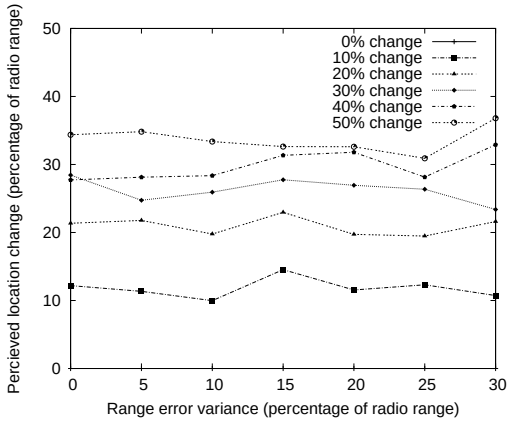


Fig. 8. 10% inter-measurement inaccuracy

enough (depending on the average rate of motion of the root node) these more difficult cases can be avoided.

The curves in all cases are relatively flat - a first guess at expected results for these experiments would assume an upwards curve in perceived motion as the error between true and measured distances increases. However, the motion detection algorithm that we are using here works with the differences between two measured distances, and as the errors for each of the two measured distances are similar, increasing the error from true distances does not significantly alter the algorithm's results.

Increasing the change in the error between the two measured distances does not change the results that much either, and this also applies with additional tests that we have done for higher values of the error change. The error values that we have used here are similar to values shown

in experimental testing [11].

## V. Related Work

Galystan et al. [3] did some earlier work using bounding boxes, with additional optimisations attempted in the area of "negative information" i.e. if two nodes can not communicate with each other, they are assumed to be out of range with each other. Bounding boxes have the assumption that a node is certain to be somewhere within them, but given the significant likelihood of bad links (two nodes that are in radio range but cannot communicate) in the real world due to a variety of possible problems (e.g. objects in the way), this will cease to be the case if we use negative information. Results from Zhao and Govindan [18], and from Zhou et al. [19] indicate that even without such obstacles, bad links still occur in a significant percentage of cases.

Capkun et al. [2] created an algorithm to create local coordinate systems, and a method for translating from one system to another. They then proceeded to attempt to use a network of co-operating nodes to build a Network Coordinate system (a form of local co-ordinate system where all of the nodes in a network use the same local co-ordinate system), using a Location Reference Group (LRG) of semi-stable (i.e. minimal movement) nodes as a centre for the topology. We have used an LRG-like system here, but using information from a local neighbourhood rather than the entire network. Network Coordinate systems result in a significantly increased amount of traffic required to setup and maintain the system over local coordinate systems, and that cost rises with the size of the network. The benefits gained via the use of this are minimal, and in most mobile anchor scenarios the situation where you have no anchor information is for a limited time only, and so cross-network protocols that could utilise a network coordinate system (e.g. node→sink message routing) would be better off storing data locally and waiting for anchor information before transmitting.

Priyantha et al. [10], as well as Shang and Ruml [14] also looked at anchor-free localisation, but using global rather than local knowledge, with the accompanying increases in network traffic and storage required for that class of solution.

Krumm and Horvitz [7] did some earlier work with motion detection using RSSI. Their method used smoothed histograms of varying signal strength from APs in an 802.11 network to determine whether a particular node was moving. The motion detection algorithm did not explicitly use location data, but the requirement for the APs to be static allows them to use them as reference points. Our work here requires that a subset of the nodes being measured are relatively static (such that comparisms between the different local co-ordinate systems can be made), but without requiring that the static nodes remain permenantly static. If we had permenantly static infrastructure nodes (e.g. an urban 802.11 network), other more efficient algorithms would be possible, but this cannot be guaranteed for WSNs.

## VI. Conclusions and Future Work

We have shown here that even in difficult localisation scenarios (such as anchor-less scenarios), where only very limited information can be used, that motion can be detected without knowing exactly where you are, and all of this can be done without additional hardware. If however we have location information from an existing localisation method, then we have also shown how we can also detect motion using more simple methods.

Getting rid of the errors in range measurements is hard to do, but that is the price of gathering data from the real world. With mass-spring anchor-free approaches, we have shown that it is possible to work around these errors, and derive good motion information. Mass-spring approaches are somewhat more computationally expensive, but given the significant improvements in the motion information, and that this enables motion detection without requiring anchors, we believe that this is worth it. Mass-spring approaches also have the advantage of being able to more rapidly detect motion, but at the cost of introducing the chance of false positives, as opposed to the zero false positives approach of bounding box methodologies.

In the future, we hope to expand on our work here to attempt to further improve the motion information that can be gathered, by integrating more accurate models of various ranging sensors, and also testing to see whether a combined model from several sensors may improve accuracy. Additionally, we would like to explore integrating together data from both mass-spring models and bounding box methods.

## References


[1] N. Bulusu, J. Heidemann, V. Bychkovskiy, and D. Estrin. Density-adaptive beacon placement algorithms for localization in ad hoc wireless networks. In *IEEE Infocom 2002*, New York, NY, June 2002.

[2] S. Capkun, M. Hamdi, and J.-P. Hubaux. GPS-free positioning in mobile ad-hoc networks. *Cluster Computing*, 5(2):157–167, April 2002.

[3] A. Galstyan, B. Krishnamachari, K. Lerman, and S. Pattem. Distributed online localization in sensor networks using a moving target. In *Proceedings of the third international symposium on Information processing in sensor networks*, pages 61–70. ACM Press, 2004.

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

[5] Andrew Howard, Maja J. Matarić, and Gaurav S. Sukhatme. Relaxation on a mesh: a formalism for generalized localization. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1055 – 1060, Wailea, Hawaii, Oct 2001.

[6] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science, Number 4598, 13 May 1983*, 220, 4598:671–680, 1983.

[7] John Krumm and Eric Horvitz. Locadio: Inferring motion and location from wi-fi signal strengths. *mobiquitous*, 00:4–13, 2004.

[8] D. Niculescu and B. Nath. Ad-hoc positioning system. In *IEEE GlobeCom*, pages 2926–2931, November 2001.

[9] T. Parker and K. Langendoen. Refined statistic-based localisation for ad-hoc sensor networks. In *IEEE Workshop on Wireless Ad Hoc and Sensor Networks (associated with Globecom 2004)*, Dallas, TX, November 2004.

[10] N. Priyantha, H. Balakrishnan, E. Demaine, and S. Teller. Anchor-Free Distributed Localization in Sensor Networks. Technical Report #892, MIT Laboratory for Computer Science, April 2003.

[11] Niels Reijers, Gertjan Halkes, and Koen Langendoen. Link layer measurements in sensor networks. In *1st IEEE Int. Conf. on Mobile Ad-hoc and Sensor Systems*, Fort Lauderdale, Florida, USA, October 2004.

[12] C. Savarese, K. Langendoen, and J. Rabaey. Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In *USENIX technical annual conference*, pages 317–328, Monterey, CA, June 2002.
[13] 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)*, pages 112–121, Atlanta, GA, September 2002.
[14] Yi Shang and Wheeler Ruml. Improved MDS-Based Localization, 2004.
[15] M. Sichitiu and V. Ramadurai. Localization of Wireless Sensor Networks with a Mobile Beacon. Technical Report TR-03/06, Center for Advances Computing and Communications (CACC), Raleigh, NC, July 2003.
[16] S. Simic and S. Sastry. Distributed localization in wireless ad hoc networks. Technical Report UCB/ERL M02/26, UC Berkeley, 2002.
[17] Eric W. Weisstein. Circle-Circle Intersection. http://mathworld.wolfram.com/Circle-CircleIntersection.html. From MathWorld - A Wolfram Web Resource.
[18] Jerry Zhao and R Govindan. Understanding Packet Delivery Performance In Dense Wireless Sensor Networks. In *Proceedings of the ACM Conference on Embedded Networked Sensor Systems*, November 2003.
[19] Gang Zhou, Tian He, Sudha Krishnamurthy, and John A. Stankovic. Impact of radio irregularity on wireless sensor networks. In *Proceedings of the 2nd international conference on Mobile systems, applications, and services*, pages 125–138. ACM Press, 2004.

## APPENDIX

When we refer to radio range in this appendix, we are using the maximum possible radio range between a pair of nodes, including any "gray area" [18] effects. The techniques here have been influenced by [5].

### A. Mass-spring model

In our mass-spring model, the range between a pair of nodes is modelled as a spring, with a known relaxation state and a spring constant. For a pair of nodes $A$ and $B$, with a range $\sim N(m,v)$, the relaxation state is equal to $m$ and the spring constant is $v$ multiplied by a scaling constant $k$. The energy $U_{A,B}$ of the spring between a pair of nodes is given by

$$U_{A,B} = \frac{|R_{A,B} - m|}{kv} \quad (1)$$

### B. Links

A link between a pair of nodes is defined as one of two possibilities, either

1) $A$ and $B$ can communicate directly i.e. $A$ and $B$ have a known value for the measured radio range between them. $A$ is therefore a neighbour of $B$ and vice versa.
2) $R_{A,B} < radio\ range$, but $A$ and $B$ are not connected using the previous rule. In this case the link distance is defined as the radio range, and the $U_{A,B}$ result is scaled by the probability of a broken link (i.e. $U_{A,B}^{broken\,link} = U_{A,B} * BrokenLinkProbability$) as given from experimental data. Values for the broken link probability will be approximately in the 0.1-0.2 range. $A$ and $B$ in this case are not neighbours, but they are linked.

A link creates a "force" that pushes the node towards a more accurate location. For a given node $A$, we can calculate the force $F_A$ on that node using

$$F_A = \sum_B F_{A,B} = -\sum_B (A \hat{\rightarrow} B) U_{A,B} \quad (2)$$

where $A \hat{\rightarrow} B$ is the unit vector from $A$ to $B$ and $A$ and $B$ are linked.

### C. Reference node placement

In order to define a local co-ordinate system, we need reference points. The root node is declared as being located at $(0,0)$, and we also require a second "reference" node to define the x-axis for this system.

We need a node that is highly connected to the root node's immediate neighbours, in order to reduce the amount of calculations we need to do later on. Therefore, the selected reference node will be one of the 1-hop neighbours of the root node, and we select it using the following rules in order

1) Highest number of root-transitive links (i.e. for a given node, the number of its neighbours that are also neighbours of the root node).
2) Highest number of neighbours.
3) If we still have >1 possible nodes, pick one randomly (lowest node id is a suggested method).

We now also declare this selected neighbour as being initially located at $(m,0)$ where $m$ is the measured distance to the neighbour. As this always makes $U_{Root,Neighbour} = 0$, this is currently a minimum energy configuration of the positioned nodes.

Once we have the reference node and root node placed, we then move onto the other nodes.

### D. Initial placement

For a node $C$ with already placed neighbour nodes $A$ and $B$, and $A$ and $B$ are neighbours of one another, we may be able to calculate an inital location. Using the measured values for all of the inter-node distances, we start by calculating $\angle BAC$ from the law of cosines.

$v = \frac{R_{A,C}^2 + R_{A,B}^2 - R_{B,C}^2}{2R_{A,C}R_{A,B}}$, $\angle BAC = cos^{-1}(v)$
Sanity assumption: $|v| \leq 1$

Using a line $D$, parallel to the x-axis but through $A$, we then calculate the angle of $A \hat{\rightarrow} B$ to $D$

$n = \frac{A_x - B_x}{R_{A,B}}$, $z = sin^{-1}(n)$ where $z$ is the angle of $A \hat{\rightarrow} B$ to $D$
Sanity assumption: $|n| \leq 1$

We can now calculate two possible values of $\theta$ (= angle of $A \hat{\rightarrow} C$ to $D$), using $\theta = z \pm \angle BAC$. We then have two possibilities for $C$'s co-ordinates using the two values of $\theta$ and $C = (A_x + R_{A,C}cos(\theta), A_Y + R_{A,C}sin(\theta))$. These are shown on Figure 9 as $C$ and $C'$. We choose the initial location of a node with the minimum amount of force (as defined in B) given the current set of placed nodes.

In some cases we will fail the sanity assumptions, and have to test with other pairs of neighbour nodes. Once we have placed all of the nodes that have a valid pair of placed neighbours, we then work on the remaining nodes.

### E. Placing remaining nodes

If we have remaining unplaced 1-hop neighbours of the root that do not have 2 neighbours in the set of already placed nodes, then we can repeat the process for selecting a reference node (C, but using only non-positioned nodes as possibilities), and place this newly selected neighbour at $(-\frac{\sum_p^{placed} p_x}{n}, -\frac{\sum_p^{placed} p_y}{n})$ i.e. an averaged location directly

opposite the current set of placed nodes, which is the most likely location for this remaining unplaced node. We now return to the process of placing additional nodes that have two neighbours in the "already placed" set, and if necessary keep repeating this sequence of processes until all the 1-hop neighbour nodes are placed.

After placing all of the 1-hop neighbours, if we still have unplaced 2-hop nodes with 2 placed neighbours, but for all possible pairs of placed neighbours *A* and *B*, *A* and *B* are not neighbours of each other, then we use the calculated locations for a pair of neighbours to work out the distance between them. The calculated distance is then used temporarily for the placement steps in Appendix D. This is less accurate, but will still give us a reasonable first guess for the location of a node.

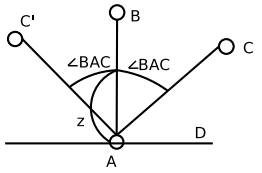

Fig. 9. Placing C

If there are still unplaced 2-hop nodes, without at least 2 placed neighbours then these 2-hop nodes must have 1 placed 1-hop neighbour (by the definition of a 2-hop node as being connected to a 1-hop node, all of which have now been placed), then we place the 2-hop neighbour at $(\frac{p_1^x(r_1+r_2)}{r_1}, \frac{p_1^y(r_1+r_2)}{r_1})$ where $r_{\{1,2\}}$ is the root→1-hop and 1-hop→2-hop measured ranges respectively, and $p_1^{\{x,y\}}$ is the x- and y-coordinates of the 1-hop neighbour.

Placing the 2-hop neighbour further along the line of the 1-hop neighbour provides a reasonably likely initial position, without the need for extensive calculations on the full set of placed nodes.

### *F. Topology optimsation*

The total energy of the system in a particular configuration is

$$Energy = \sum_{A,B} U_{A,B} \quad A,B \in placed\,nodes$$
and there exists a link between *A* and *B*

An optimal topology for a mass-spring system is when the total energy of the system reaches a pre-defined minimum value (ideally zero, but in practice this will often not be possible to achieve). We may not be in this state after the inital placing, as we did not take all of the link information into consideration initially. We therefore need to further refine our location data.

The location of each node *A* is refined, firstly for the 1-hop neighbours, then the 2-hop neighbours. For 2-hop networks, this makes sure that a node's parents will always be evaluated before the node itself. *A* is refined as follows:

1) If *A* has an ancestor node (parent, parent of parents, etc) that switched to its alternate location during this round of the algorithm, then recalculate *A*'s location and alternate location according to the previously specified initial placing algorithm (D).
2) Otherwise
   a) Calculate *A*'s current force $F_A$, with Equation 2.
   b) If *A* has an alternate location, which is a valid location given the communication links to this node i.e. all direct links to *A* are within radio range of the alternate location, calculate the force for the alternate location as well, and if the magnitude of that force is smaller, *A* is moved to the alternate location.
   c) Update *A*'s current estimated location
   $A \leftarrow A + F_A T$
   where $T$ is an arbitrary constant controlling the rate of convergence.

These steps are repeated until a minimum energy state is reached, or until the reduction in energy from one state to the next drops below a pre-defined limit (or the energy increases!). One possibility for improving the speed and accuracy of this process is to choose a value for $T$ that is proportional to *Energy*, allowing for rapid reductions initially, reducing the motion as we progress towards the minimum energy state. Other techniques such as simulated annealing [6] could also be applied to select suitable values for $T$.

### *G. Motion detection*

Using anchor-free co-ordinate systems, motion detection is possible by comparing a generated local co-ordinate system at one moment in time ($LCS_1$) to another generated system by the same node at a later point in time ($LCS_2$).

For each pair of nodes which we will designate *A* and *B*, using the $LCS_1$ system co-ordinates for *A*,*B* and our root node (marked as *I*), as well as range data from $LCS_2$ for our root node relative to *A* and *B*, we can calculate the possibilities for the location of the $LCS_2$ root in $LCS_1$ (designated as *K*) using

$$(K_x - A_x)^2 + (K_y - A_y)^2 = R_{K,A}^2 \quad (3)$$
$$(K_x - B_x)^2 + (K_y - B_y)^2 = R_{K,B}^2 \quad (4)$$

Solving for $K_x$ in terms of $K_y$, *A* and *B*, gives us

$$e = R_{K,A}^2 - A_y^2 - A_x^2 \quad (5)$$
$$m = \frac{e - (B_y^2 + B_x^2 - R_{K,B}^2)}{2(B_x - A_x)} \quad (6)$$
$$n = \frac{2(B_y - A_y)K_y}{2(B_x - A_x)} \quad (7)$$
$$K_x = m - nK_y \quad (8)$$

Using Equations 3 and 5-8 we can then solve for $K_y$

$$o = (A_x n - A_y - mn)^2 (m^2 + 2A_x m - e) \quad (9)$$
$$K_y = \frac{-2(A_x n - A_y - mn) \pm 2\sqrt{o(n^2+1)}}{2n^2+1} \quad (10)$$

Equation 10 gives us two values for $K_y$ which we can then substitute back into Equation 4 to get values for $K_x$.

$$h = R_{K,B}^2 - B_y^2 - B_x^2 \quad (11)$$
$$K_x = -B_x \pm \sqrt{2B_y K_y + h - B_x^2 - K_y^2} \quad (12)$$

This gives us up to 4 $(K_x, K_y)$ pairs that represent potential values for *K*. Results involving imaginary numbers are discarded, as they do not represent valid solutions.