Delft University of Technology
Parallel and Distributed Systems Report Series

Localisation in Mobile Anchor Networks

Tom Parker **Koen Langendoen**

report number PDS-2005-001

PDS

ISSN 1387-2109

Published and produced by:
Parallel and Distributed Systems Section
Department of Information Systems and Algorithms
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Zuidplantsoen 4
2628 BZ Delft
The Netherlands

Information about Parallel and Distributed Systems Report Series:
reports@pds.twi.tudelft.nl

Information about Parallel and Distributed Systems Section:
http://pds.twi.tudelft.nl/

Localisation in Mobile Anchor Networks

Tom Parker* Koen Langendoen
{T.E.V.Parker, K.G.Langendoen}@ewi.tudelft.nl

Abstract

Localisation is required for many ad-hoc sensor network applications. Therefore, much work has been done regarding techniques for localisation, mainly using anchors (nodes with known locations). However, there has been little study of how anchors are likely to be distributed in applications, and how to perform localisation with more realistic anchor distributions.

In this paper we look at the limitations of many of the existing proposed localisation techniques with regards to coping with non-uniform anchor distributions and errors in ranging information. We present a refined approach using mobile anchor scenarios for anchor information distribution, combined with statistical techniques for performing localisation with inaccurate range data. We also show methods for dealing with motion in nodes using related techniques, including an anchor-less solution to ensure that we can always detect motion.

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.

1 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). 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 [5]; recent work includes [1], [8], [11], [12] and [14].

With WSN localisation, some nodes 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. Instead, in this paper we look at mobile anchors to provide usable anchor distributions. Mobile anchors require a minimum of additional effort on the part of the individuals setting up the sensor network, and also reduce the hardware costs for a sensor network compared to static anchor networks.

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 localisation algorithms must

* Supported by the Dutch Organisation for Applied Scientific Research (TNO), Physics and Electronics Laboratory.

be aware of and be able to work with.

In this paper we will look at a number of the limitations with many of the existing proposed localisation techniques, show how they are unlikely to work well when the uniform anchor distribution assumption breaks down, and present a refined approach. This approach uses mobile anchor scenarios, along with statistical techniques for performing localisation with inaccurate range data. Simulations with our improved approach have shown significant reductions (in the order of magnitude range) to the required processing for performing statistic-based 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. We also look at what can be done to cope with the possibility of non-anchor nodes moving.

2 Mobile anchors

In this section we look at how anchor information can be distributed across an ad-hoc sensor network, and how mobile anchor scenarios have several advantages over other methods.

2.1 Anchor distribution

Most methods for providing location information to a sensor network start with adding additional localisation hardware (e.g. GPS) to a small percentage of the nodes in the target area. These anchor nodes will initially gather accurate location information on their own, and then transmit this information to their neighbouring nodes. This approach has a number of major faults:

- Most localisation algorithms based on "spread anchor" scenarios rely on the anchors being uniformly distributed across the sensor network. Unless special care is taken to make sure of this, or a very large percentage of the nodes are anchors, then this is unlikely. Given a small anchor percentage (as in most proposed applications), there is a high probability that there will be regions of the sensor grid that have insufficient anchors, leading to problems in attempting to localise any nodes in that region.

- Anchor nodes are generally more expensive (because of the additional hardware requirements), which creates a difficult decision regarding the balancing of the application requirements between having improved accuracy (lots of anchors) and reducing the overall cost of the network (few anchors).

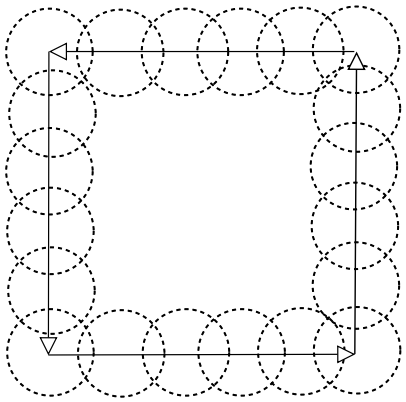


Figure 1: Example mobile anchor scenario

- The additional anchor hardware is often only useful during the initial phase of the network setup, and is then mostly surplus to requirements. An anchor node may also have a reduced operational lifespan due to the power drains of the localisation hardware.

- There have been some attempts to fix these problems (Adaptive Beacon Placement [1] for example), and there are partial fixes, but a better approach is to look at other ways that location information can be distributed rather than the use of static anchor nodes.

2.2 Mobile anchor scenarios

Mobile anchors [14] are an alternate approach, resolving a number of the problems with the spread anchor scenarios. This approach uses a single, large anchor capable of moving along a path. This large anchor could be carried by a car or a person for example. The intention is that this larger anchor will have effectively unlimited power (i.e. can transmit as many messages as needed) because it is intended to be more easily accessible than the individual sensors, and so replacing the anchor node's batteries is less of a problem than replacing batteries in the sensor nodes.

As the mobile anchor moves, it broadcasts its location at regular intervals (either every few seconds, or after it has moved a short distance from its last broadcast location), thus creating a series of "virtual" anchors, as in Figure 1. Each circle represents a position where the mobile anchor broadcasted its location. This creates a very dense anchor distribution, but only in limited areas.

2.3 Real-world applications

To see how mobile anchor scenarios would work for various applications, we looked at the structure of these applications, and saw how we could better utilise the already available resources. The main area of interest regards the method for the distribution of the sensors. A number of different methods have been proposed, varying from the manual placing of individual nodes, through to the dropping of nodes from a plane. These can be grouped into two main categories depending on the distance from the object that is placing the nodes to the location that the node is being placed at.

The simplest scenarios are when the distance is less than the node's radio range (ideally much less). In this case, the placing object itself (be it a person or a car) is the mobile anchor. This can be achieved by combining an anchor node with the placing object (either carried by the person, or attached to the car). It can then broadcast its location information as it places the nodes, thus providing a path that passes near all of the nodes.

More complicated are the situations where the nodes are far away from the placing object, for example when dropping nodes from a plane (especially from a high altitude, or when trees or other obstacles are likely to block radio signals from the placing object). One solution to this problem is that the plane could drop one or more small robots fitted with localisation equipment, in addition to the sensor nodes. These robots could travel along a semi-random path around the sensor grid (with constraints to keep them near the grid), and provide location information to the sensor nodes as they move around.

Additionally, there will also be scenarios where nodes are initially placed far away from any mobile anchors, but mobile anchors later come in contact with part of the network. One example would be of a mine field with attached nodes. Groups of soldiers moving near to the mine field could act as mobile anchors, but would not be able to go across the mine field, resulting in a mobile anchor scenario with a path along one side of the network. In these scenarios, the anchor information distribution in the network improves over time, resulting in increasingly accurate location data for the sensor nodes.

2.4 Advantages

There are several advantages of the mobile anchor approach

- Instead of many anchor nodes (and having to make the trade-offs regarding how many) we have effectively many anchor nodes, but for the cost of only a few anchor nodes (one per placing object). The anchor infrastructure is therefore "there when you need it, not when you don't". All of the sensor nodes should have similar lifetimes, without the additional power drains that would occur if some of them were also anchors for the network.
- In the complicated scenario with the use of mobile anchor robots, the cost of the scenario does go up from what would be possible with more simple scenarios. However, the robots could also be fitted with additional sensors (above and beyond what would be fitted to normal nodes), so that once they have finished providing location information to the network, they can be moved to locations where events are happening to gather more detailed information. The possibility of very simple (and cheap!) sensor nodes coupled with larger robot-mounted sensor arrays would provide a cost-effective methodology for detailed data gathering without requiring every node to have a large sensor array.
- In the event that the initial anchor path is not sufficient to provide good location information for all of the sensor nodes, we may (depending on the application) be able to do on-the-fly improvements in bad areas. The equivalent solution [1] for standard anchor scenarios would involve placing additional anchor nodes, at additional cost, but with mobile anchors we can simply move the mobile anchor near the inaccurately located nodes. As we do not currently know where the inaccurate nodes are, we have to search for them, but this can be done by starting from their neighbouring nodes (as discovered from the radio topology) and searching a circle with a radius equal to the radio range.

3 Existing localisation methods

In this section we have a brief look at existing localisation algorithms, with an emphasis on their capabilities regarding the handling of inaccurate distance information, and their ability to handle the non-uniform anchor distributions which occur in many mobile anchor scenarios.

3.1 Deterministic methods

Langendoen and Reijers [7] studied three localisation algorithms (Euclidean, Hop-Terrain, Multilateration) that can handle low numbers of anchors, and identified a common three-phase structure. First, information about the anchors is flooded through the network to determine the

(multi-hop) distances between anchors and nodes. Second, each node calculates its position using the known positions and estimated distances of the anchors, for example, by performing a lateration procedure (as with GPS systems). Third, nodes refine their positions by exchanging their position estimates and using the one-hop internode distances. After these three stages, a subset of the nodes have location information that is considered "good" (i.e. reliable).

Euclidean [8] uses basic geometric reasoning (triangles) to progress distance information from the anchors to the nodes in the network, and uses lateration to calculate the position estimates; no refinement is included in the algorithm. Euclidean's basic safety measure against inaccurate range information is to discard "impossible" triangles generated in phase 1. Unfortunately, this happens quite often, leaving many nodes in phase 2 without enough information to calculate their position (distances to at least 3 anchors are required). The end result is that Euclidean is only able to derive an accurate position for a small fraction of the nodes in the network.

Hop-Terrain [8, 11] avoids the range error problem to a large extent by using only topological information in phase 1. The distance to an anchor is determined by counting the number of hops to it, and multiplying that by an average-hop distance (calculated by the anchor nodes during an initial anchor information flood). Next, the node positions are estimated by means of a lateration procedure. In the refinement phase, Hop-Terrain switches to using the measured (inaccurate) ranges to neighbouring nodes. To avoid erroneous position estimates affecting neighbours too much, the refinement phase uses confidence values derived from the lateration procedure (dilution of precision and residue). Hop-terrain works reasonably well for a regular network topology in which nodes are evenly distributed. This however is not the case for a significant number of WSN scenarios, resulting in the algorithm becoming increasingly less accurate as the regularity assumption starts to break down.

Multilateration [12] starts by summing the distances along each multi-hop path in phase 1. To account for the accumulated inaccuracies it does not perform a lateration procedure, but instead uses each distance to specify a bounding box centred around the associated anchor, in which the node may be located. In phase 2, these bounding boxes are simply intersected and the position estimate is set to the centre of the intersection box, followed by a refinement procedure in phase 3. Multilateration's effectiveness with varying errors in range measurements will depend on the exact nature of the errors. If many of the measured distances are larger than the true distances, then Multilateration should be able to cope with the problem (as the true distance will still fall within the bounding box). However, in general, ranges are likely to be both under- and over-estimated (our current experimental model treats both as equally likely), and Multilateration is less likely to be able to cope with under-estimation, and so will result in possible location information being discarded due to contradictions between ranges with varying errors.

3.2 Statistic-based Localisation

One technique that attempts to do more with inaccurate ranging information is Statistic-based Localisation. The initial work on this was performed by Sichitiu and Ramadurai [14]. It assumes that while the incoming range data has errors, these errors can be modelled with a probability distribution based on the incoming data - either from a sensor or the Radio Signal Strength Information (RSSI) from the radio. This model can be worked out either from manufacturer-supplied data for the sensor providing range data, or from experimental data [18]. Sichitiu and Ramadurai focused on RSSI, but other sensors could be used if there are error models for them - it would not be necessary to re-write the algorithm to do this.

Given that a node has a series of distances to anchors, and that for each distance you have an error model, these models can be combined to calculate a "map" of the most likely locations for this node, by calculating probabilities for each location at discrete intervals across the sensor grid. Figure 2 shows a visualisation of an example map, and Algorithm 1 provides more details about how the maps are generated.

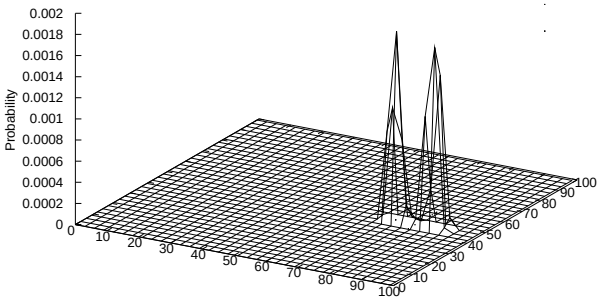


Figure 2: Visualisation of a node's localisation probability map

Sichitiu and Ramadurai's technique has two major problems however. Firstly, the large quantities of calculations required to create the maps and secondly, the requirement for the nodes to be at one-hop distances from the anchors (achieved in [14] by using a mobile anchor with a very dense path) - this is required because they do not have a method for distributing anchor information re-

ceived by one node to another, and so only anchor nodes could send their distance information.

4 Refined Statistic-based Localisation (RSL)

Given the problems with existing algorithms, we propose a set of refinements to the earlier statistic-based localisation algorithm. We have chosen this because a combination of statistic-based localisation and mobile anchor scenarios works well together, as the dense topologies of mobile anchors can be exploited for significant accuracy increases. This is a different approach to most other localisation algorithms, which are designed towards sparse anchor topologies.

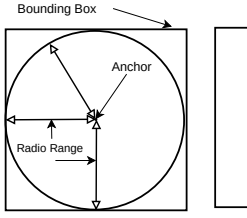
In this section, and in Algorithm 2 we explain Refined Statistic-based Localisation (RSL). In this improved algorithm, we can use dense topologies in limited regions of the network (as generated by mobile anchors) to create "pseudo-anchors" which will be able to spread the anchor information further into the network. Additionally, we have made changes to reduce the computation load on individual nodes, creating a more useful algorithm for applications with limited resources (i.e. most proposed sensor node applications). We also look at some additional future improvements that could be made to further improve on our algorithm.

4.1 Bounding boxes

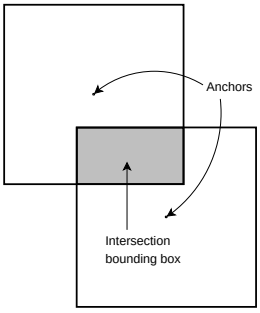
If a node has received a position estimate from an anchor then it knows it is in radio contact with that anchor, and so therefore it must be within radio range of that anchor. So, we can limit the space of possible locations for that node to a circle centred on the anchor's location with radius equal to the radio range. 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 3a). (The basic concept of bounding boxes has previously been analysed in [15], but not in combination with any form of statistic-based localisation.) This results in a larger region, but we still have the guarantee that all feasible locations for the node are located within the box, while keeping the box size to a minimum. For radios with non-circular transmission spaces, we can also similarly calculate the minimum box that contains the entire possible transmission space.

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 3b, and Algorithm 2, step 2). A bounding box is defined by two points, its Top-Left and Bottom-Right corners. Note that the probability visualisation in Figure 4 on page 7 only shows a partial grid (as opposed to Figure 2 which shows basic Statistic-based localisation, and uses a complete grid) - this partial grid is the section of the complete sensor grid corresponding to the bounding box for this particular node.

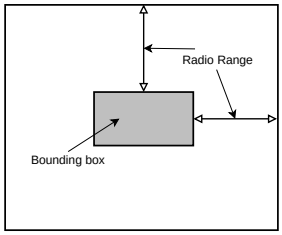
Experimental results for testing the reduction in the size of the calculated sensor grid, show an average reduction in the number of required calculations by a factor of 8 when we use bounding boxes. Also, with the additional optimisation of not doing calculations for the nodes with the largest bounding boxes and simply assuming they have an unknown location, we could improve this result further. For example, by not performing any calculation for nodes with bounding boxes where $width * height > 0.75 * (2 * RadioRange)^2$, we reduce the overall calculation load by an additional factor of 3. This does reduce the overall accuracy, but nodes with large bounding boxes would have had high error values if we did attempt to localise them, and so getting rid of the most hopeless cases saves significant amount of computational time, while only resulting in a small difference in the overall average accuracy.



(a) Anchor bounding box



(b) Union of two bounding boxes



(c) Non-anchor bounding box example

Figure 3: Bounding Boxes

4.2 Limited broadcast

To get around the problem of needing anchors within one-hop of the sensor nodes, we perform a limited broadcast of calculated node location information - limited by only

Algorithm 1 Statistic-based localisation [14]

1. Initially, the local probability "map" is set to a constant value across the entire sensor grid, as all locations are considered to be equally likely at the start of the algorithm.
$$PosEst(x,y) = c \quad \forall (x,y) \in [(x_{min}, x_{max}) \times (y_{min}, y_{max})]$$

2. Incoming anchor information is processed as follows:

   (a) The incoming anchor location is used to create a refinement function on the possible locations of the node
   $$PDF_{rssi} = N \sim (EstimatedDistance_{anchor}, RadioRangingVariance)$$
   $$Refine(x,y) = PDF_{rssi}(distance((x,y),(x_{anchor}, y_{anchor}))) \quad \forall (x,y) \in [(x_{min}, x_{max}) \times (y_{min}, y_{max})]$$

   (b) The node applies Bayesian inference to its current map to generate an improved map
   $$NewPosEst(x,y) = \frac{OldPosEst(x,y) \times Refine(x,y)}{\sum_{x_{min}}^{x_{max}} \sum_{y_{min}}^{y_{max}} OldPosEst(x,y) \times Refine(x,y)} \quad \forall (x,y) \in [(x_{min}, x_{max}) \times (y_{min}, y_{max})]$$

3. Finally, the weighted average of all of the data in the map is used to calculate the estimated position of this node
$$(\hat{x}, \hat{y}) = (\sum_{x_{min}}^{x_{max}} \sum_{y_{min}}^{y_{max}} x \times PosEst(x,y), \sum_{x_{min}}^{x_{max}} \sum_{y_{min}}^{y_{max}} y \times PosEst(x,y))$$

Algorithm 2 Refined Statistic-based Localisation (RSL)

Abbreviations used here:
TL = Top-Left corner of a bounding box, BR = Bottom-Right corner of a bounding box, R = Radio Range of the nodes

1. Initially, the bounding box for a node is set to $[(-\infty, \infty) \times (-\infty, \infty)]$.

2. As (pseudo-)anchor information comes in, the bounding box for this node is intersected with the existing bounding box (see Figure 3 for examples of bounding boxes, including a diagram of this step in Figure 3b)
$$NewBox(TL, BR) = [(Max(Anchor_{TL_x} - R, OldBox_{TL_x}), Max(Anchor_{TL_y} - R, OldBox_{TL_y})) \times (Min(Anchor_{BR_x} + R, OldBox_{BR_x}), Min(Anchor_{BR_y} + R, OldBox_{BR_y}))]$$

3. Once information from at least two (pseudo-)anchors have been received, and the minimum waiting period since the last incoming anchor has passed, then we initialise the local map to a constant value
$$PosEst(x,y) = c \quad \forall (x,y) \in BoundingBox$$
and then each of the incoming (pseudo-)anchors that we have received so far is processed as follows:

   (a) The incoming anchor information is used to create a refinement function on the possible locations of the node
   $$PDF_{rssi} = N \sim (EstimatedDistance_{anchor}, RadioRangingVariance / Confidence_{Anchor})$$
   $$Refine(x,y) = PDF_{RSSI}(distance((x,y),(x_{anchor}, y_{anchor}))) \quad \forall (x,y) \in BoundingBox$$

   (b) The node then multiplies each value in the map by the refinement function to generate an improved map
   $$NewPosEst(x,y) = OldPosEst(x,y) \times Refine(x,y) \quad \forall (x,y) \in BoundingBox$$

4. The location on the map with the highest probability is determined (this is the most-likely location for this node)
$$(\hat{x}, \hat{y}) = maxarg\{PosEst(x,y) \,|\, (x,y) \in BoundingBox\}$$

5. Finally, the map is normalised to provide an externally-usable probability value
$$NormConstant = \sum_{BoundingBox_{TL_x}}^{BoundingBox_{BR_x}} \sum_{BoundingBox_{TL_y}}^{BoundingBox_{BR_y}} PosEst(x,y)$$
$$FinalPosEst(x,y) = PosEst(x,y) / NormConstant \quad \forall (x,y) \in BoundingBox$$
(this works because the bounding box always has the property that the probability that the current node is within the bounding box is 1, and so therefore we can normalise the data)

broadcasting if we exceed a minimum probability threshold for the quality of our location information (currently set in our implementation to 0.003). The node effectively acts as an additional "pseudo" anchor, but with two changes from normal anchors.

Firstly, the location information is broadcast with a confidence value (gained from the local probability map), and the error model used by nodes receiving this information will be scaled accordingly, as shown in Algorithm 2, step 3a with the use of $Confidence_{anchor}$ in the generation of PDF_{rssi}. This confidence value is a weighting value used in building the statistical models i.e. a node with confidence 1.0 (an anchor node) will have twice the effect of a node with confidence 0.5.

Secondly, the bounding box for this node is broadcast as well, and the box used by receiving nodes is not just a square centred on the node (as for anchors), but a rectangle equal to the bounding box size, plus radio range in each direction (Figure 3c). This is because the bounding box contains all locations the node could possibly be in, and so increasing it by the radio range creates a box in which nodes that can hear this node could possibly be located. This box will be larger than a box generated from an anchor node, because the location information is less accurate. However, this larger box may still be useful to other nodes in reducing their bounding boxes, and hence reducing the amount of computation that they need to perform.

Nodes that have position information, but do not exceed the probability threshold are considered "bad". These nodes have some position information, but either the information is insufficient, or it is of too low a quality to be fully usable. These do not broadcast their location information to other nodes.

One additional scenario that uses pseudo-anchors is when we have location information from another system (e.g. GPS) and this data is inaccurate. We can then treat this inaccurate anchor as a pseudo-anchor, with an appropriate confidence value and bounding box depending on the incoming data. RSL does not actually specifically require accurate anchors, but simply some sources of initial localisation data to initialise the algorithm.

In our experiments comparing performance with and without the limited broadcast method, we see a similar average error in the locations of the good nodes, but a 38% average increase in the number of good nodes when limited broadcast is used.

4.3 Symmetry problem

There are a number of situations where we will have multiple points that have equally high probabilities (or certainly very similar, and within the bounds of statistical error). One of the most likely instances of this problem is when the mobile anchor is travelling in a straight line. As the distributions of the broadcast anchors cross over equally on both sides of the line, a pair of possible good points will be created, each one equally far away from the line, but on opposite sides. Figure 4 is an example of this, showing the local probability map for a node with this particular problem.

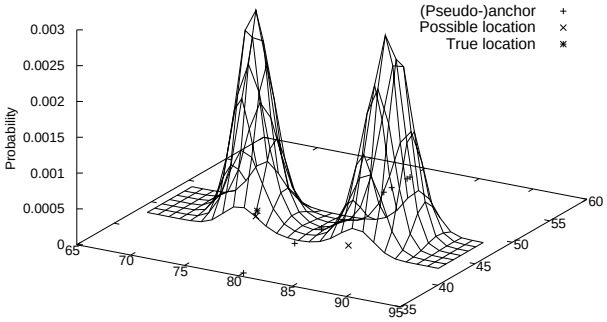


Figure 4: Equal points

The broadcasting of pseudo-anchor data from nodes with good location data reduces the symmetry problem, as this allows the possible locations for the node to be "pulled" in the direction of the correct point. However, in the event that the mobile anchor's path is a straight line, or that there are insufficient nodes in the local area to broadcast pseudo-anchor information, then the problem still occurs. A solution to this is to avoid choosing straight paths for the mobile anchor - simple possibilities include curved paths, or using a "wobbly" path rather than a perfect straight line. Corners and curved sections on the mobile anchor's path reduces the chances of the symmetry problem considerably.

To attempt to combat the symmetry problem, we have devised the *equal pairs heuristic* to test for and compensate against this problem. A node can determine whether or not there is likely to be multiple possible positions, based on its local probability map, by calculating the average of all of the anchor locations the node knows about (weighted according to their confidence values), and seeing how much each anchor's location differs from their average location in each separate axis. This tests to see whether the known anchors are mostly arranged along a straight line, or whether they have a more varied path. If there is a significantly greater total difference from the average point in one axis than another (indicating a mostly straight path), then the node will also test the other possible good points. These can be found by taking the averaged anchor location, then looking at the points that are on the opposite side of the average point from the cal-

culated most-likely location for this node. An average of the most-likely location and the other possible points (weighted according to their individual confidences) will become the node's estimate of its true location.

If the sum of the confidences for the most-likely point and the best of the other candidate points, divided by a scaling factor, is above the standard threshold for transmission of the calculated location, then we transmit both locations. The scaling factor varies according to the degree of difference between the two confidences i.e. how good the second confidence is compared to the first.

$$ScalingFactor = 2 \times \frac{confidence(Best) + confidence(SecondBest)}{confidence(Best)}$$

If the second point is similarly confident to the first, then the scaling factor will be proportionally greater, but it always satisfies the condition $2 \leq ScalingFactor \leq 4$.

If the node decides to transmit its current guesses, then the confidences for both points are transmitted, and the node is treated as two separate nodes by its neighbours, one at each of the two possible points, but each with a reduced confidence (compared to the calculated confidence for the point).

In our experiments testing the use of the equal pairs heuristic on a typical path using straight lines (a square), we saw up to a 30% decrease in the average error compared to tests without the heuristic.

4.4 Heavy data-processing

One downside of statistic-based methods is the amount of data processing required to calculate the local maps. The bounding boxes reduce this significantly (a factor of 8), by eliminating many regions that this node can not be located at. For best results, there should be a waiting period for a short amount of time (e.g. 5 seconds) after the last piece of anchor information has been received, before calculating the local map, in order to work with the smallest possible bounding box. This will slow down the calculation of this node's location, but given that it is necessary to recalculate the data if we receive more anchor information, this can reduce the amount of redundant calculations significantly. The waiting period should be calibrated such that if we have not seen a new anchor for that amount of time, then we are unlikely to receive more anchor information in the near future. Good values for this would be at least as large as the interval between broadcasts of the mobile anchor.

The energy costs associated with RSL are higher than for most localisation techniques, due to the large number of probability calculations required (Lateration being a notable exception, due to the use of least-squares calculations), but this additional cost is in most cases a one-off initialisation cost. Simulation results show an increase in processing time for RSL over deterministic techniques ([8, 11, 12]) by approximately a factor of 2. For a typical node CPU [16] and an average bounding box taken from experimental testing, this increases the localisation runtime from ~5 to ~10 seconds. This order of processing time is not an unacceptable start-up cost for a long running application, given the significant improvements in the derived location information. This can also be performed by many nodes at the same time without additional costs, as opposed to other localisation techniques requiring large numbers of radio messages (which would exhibit increased numbers of packet collisions if several nodes are transmitting radio messages at the same time). RSL has been deliberately optimised towards reduced radio messages with this aim in mind.

An additional optimisation that could reduce the data processing cost is the alteration of the grid size of the calculations. When the probabilities for a region are calculated, this is done at discrete intervals, resulting in a grid of probabilities for a region. The distance between points in the current implementation is fixed in size, but this could be varied on a per-node basis, depending on the size of the bounding box. Larger bounding boxes would increase the point distance, and smaller boxes would decrease it, creating an approximately equal point count (and therefore processing time) for all nodes, while allowing nodes with small bounding boxes to have more accurate estimates than they achieve currently. The point count could also be varied at the application level, to allow for application-specific accuracy requirements.

RSL could also be further improved by the limiting of Step 5 in Algorithm 2. Currently, we generate the normalised map for all locations in a node's bounding box. However, we then only use a small subset of this data. An additional reduction in processing time could be gained by only calculating normalised probabilities for the most-likely location and for the additional possible locations needed for the solving of the symmetry problem (Section 4.3).

5 Results

Using the Positif simulation framework for localisation algorithm testing [7], we have performed a series of comparison tests between RSL, and three deterministic localisation techniques (Euclidean [8], Hop-Terrain [11] and Multilateration [12]), using a mobile anchor scenario in all cases, and with a variety of ranging errors between

nodes. In each case, all of the algorithms have been tested with the same set of data, and each result is the average of 20 runs of the simulation with varying random-number seeds.

The ranging error is modelled as a Gaussian distribution, with the mean as the actual range, and the range variance as a percentage of the radio range. The internal model of RSL in all cases is set to a Gaussian distribution with the mean as the incoming range information, and the estimated range variance set to 20% of the radio range. In all scenarios, there are 226 sensor nodes randomly placed, with a uniform distribution, within a square area. The mobile anchor is modelled as a formation of 111 "virtual" anchors within this sensor grid. The grid has a size of 100x100, and the radio range is set to 14 providing the nodes with an average connectivity of 19 including the anchors, or 12 with only the sensor nodes.

There are three different mobile anchor scenarios being considered here. The first is a "square" formation, with a mobile anchor moving along a square path situated approximately 1/5th of the sensor grid width from the edge of the grid at all times. The second is a "cross" formation, testing what might happen with two separate mobile anchors, one moving from the the top-left to bottom-right, and the other moving from bottom-left to top-right. In both cases, the start points are situated 1/5th of the sensor grid width from the edges of the grid. The straight lines of these two topologies have been deliberately chosen to cause difficulties to RSL. The third topology is a "wobbly" square, taking the square formation locations as a base, and then moving the anchors by a random amount (maximum distance of 2, uniformly distributed) away from the initial location to provide a less straight-line path.

Figures 5 (square), 6 (cross) and 7 (wobbly) are visualisations of the individual node locations for a set of example experiments that we have performed using RSL. The nodes marked with a "•" are anchor nodes, the others are sensor nodes; the ones marked with a "*" are good nodes, nodes marked with a "+" are bad nodes, and the "△" nodes have no position data at all. Lines attached to nodes show the path from a node's true position to where it thinks it is. The longer the line, the less accurate the estimated position. Note that, in general, RSL does a good job of classifying the nodes into good and bad ones, but occasionally generates both false positives (good nodes with long lines) and false negatives (bad nodes with short lines). These anomalies generally occur outside the area directly covered by the mobile anchor. Since the node classification is largely correct, applications should be able to exploit that knowledge to their advantage.

In figures 8, 9 and 10, we show the average accuracy of the good nodes for all of the algorithms. For RSL, we also have bad nodes, so we also show the accuracy for a weighted average of both good and bad nodes. Figures 11, 12 and 13 show the average percentages of positioned nodes in each of these cases. Note the poor coverage (generally less than 50% of the nodes obtain a position estimate) for the square and cross topologies, which shows the problems induced by non-uniform anchor distributions in combination with the symmetry problem for straight line topologies.

In most cases RSL has the lowest percentage error in its "good" positions. Euclidean only outperforms it under ideal circumstances (i.e. no range errors); in all other cases (error variance > 5%) RSL provides (much) more accurate position estimates. In general, localisation algorithms can trade-off accuracy for coverage [7]. RSL, however, combines high accuracy with reasonable coverage. For low error variances, RSL has similar numbers of good nodes as the Hop-Terrain and Multilateration algorithms, only at higher values RSL starts to classify more nodes as being bad. The combination of the RSL good and bad nodes however, gives a comparable level of error to the other algorithms, but with up to a doubled number of positioned nodes.

The "square" topology was chosen as a typical example of a simple mobile anchor scenario, which could have been implemented by for example a mobile node attached to a car driving around a square-shaped building. These x/y-axis aligned paths can be detected by the equal pairs heuristic (Section 4.3) in some cases and compensated for accordingly. Despite the problems still occurring due to the straight paths, RSL is still capable of getting reasonable results.

The "cross" topology was designed to attempt to break the current implementation of RSL, as the equal pairs heuristic does not work as well with diagonal paths. However, although RSL has less accurate results for the cross topology, all of the other algorithms also do badly as well. We would therefore not recommend the use of the "cross" topology for use in mobile anchor applications.

The "wobbly" square topology is an example of a topology that should be easier for the localisation algorithms, as the significantly lower errors for this topology shows. One example case where we would expect to see this sort of topology is where the mobile anchor is attached to a soldier patrolling the perimeter of a base. The significantly better results with this topology over the straight-line topologies is why it is recommended to avoid straight lines with the mobile anchor paths.

In all of the experiments the internal model of RSL has

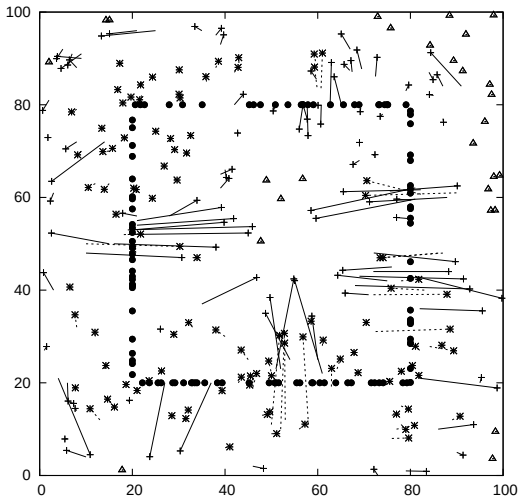


Figure 5: Square topology, 20% range error variance

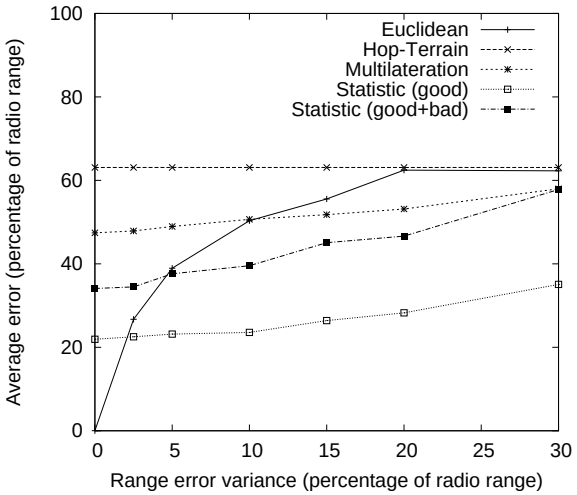


Figure 8: Square topology accuracy

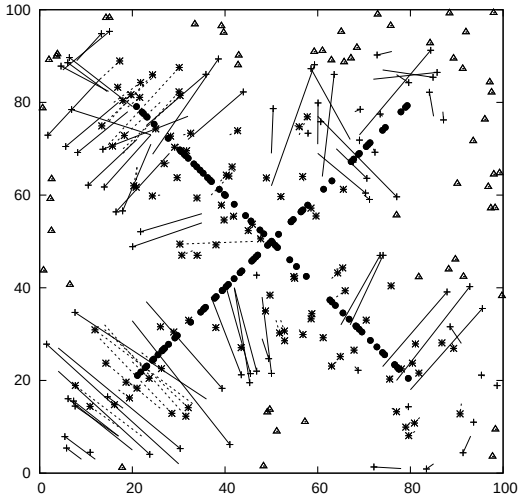


Figure 6: Cross topology, 20% range error variance

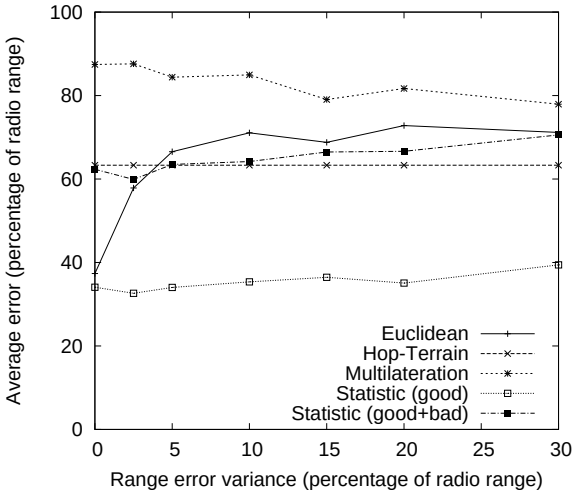


Figure 9: Cross topology accuracy

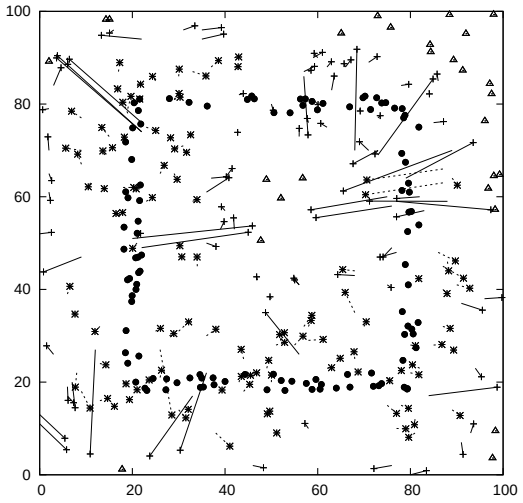


Figure 7: ”Wobbly” square topology, 20% range error variance

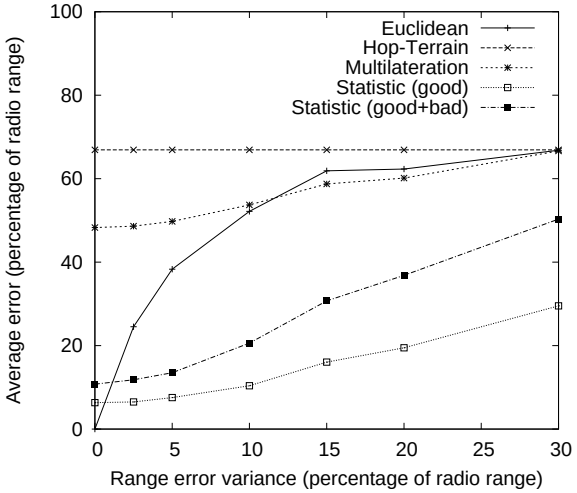


Figure 10: “Wobbly” square topology accuracy

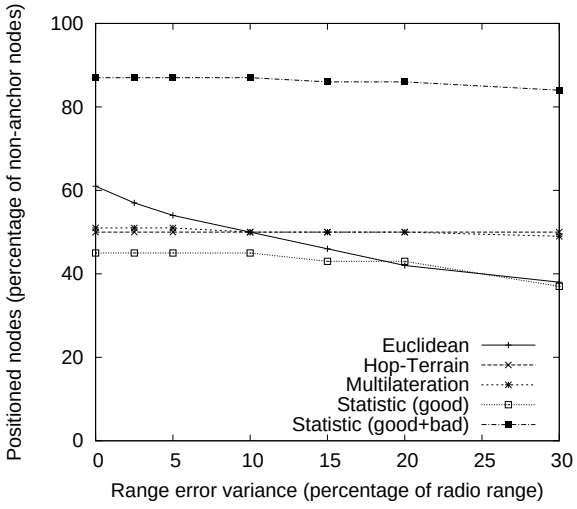


Figure 11: Square topology coverage

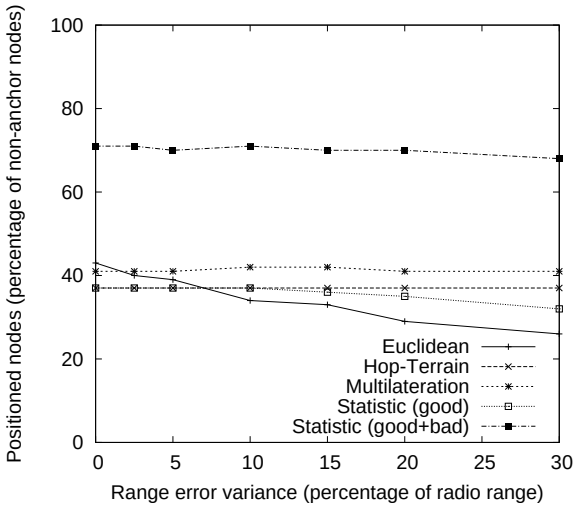


Figure 12: Cross topology coverage

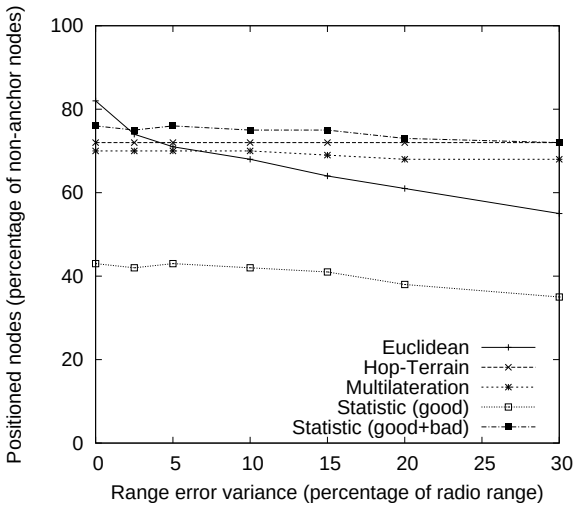


Figure 13: "Wobbly" square topology coverage

been set to a Gaussian distribution with variance as 20% of the radio range. The results for that particular variance of the actual errors are not much better than for other variances of that magnitude. Note therefore, that we can get good results even when the actual ranging information model is significantly different from the internal model of the algorithm. It is important to try and get the internal model as similar as possible to the actual model, but as these results show, good data can be acquired even when the internal model is inaccurate.

6 NODES IN MOTION

So far, we have considered networks consisting of static non-anchor nodes. 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 this part, we look at the problem of how to deal with moving non-anchor nodes. Firstly, we will examine what can be done with RSL, and how we can use the bounding box information 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 localisation. 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 where they were when the data was gathered. This is a likely scenario 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.

As we can not work out where the node is actually located (due to the lack of anchor information), we will concentrate instead on detecting motion of the non-anchor nodes, so that when anchor information is acquired, we can work out where the node was based on the motion information.

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. How-

ever, 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.

7 Motion with bounding boxes

Bounding boxes in RSL assume a sanity condition - that the current bounding box of a node and a new box that it has just received, and therefore wishes to intersect with, will always have a non-empty union.

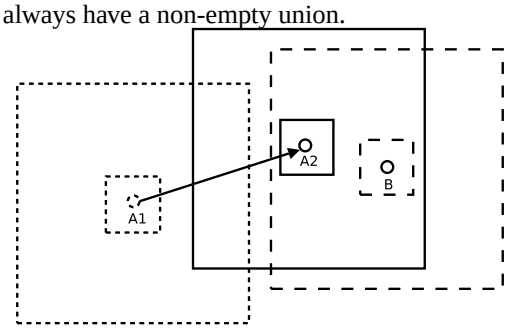


Figure 14: Motion example

Figure 14 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.

In a number of cases (especially with large bounding boxes) we will not be able to detect motion, but in these cases we do maintain bounding box consistency, so we can still perform statistical calculations, although with a slightly reduced accuracy. 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 another neighbour (not including either N or M) considers 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 2), 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.

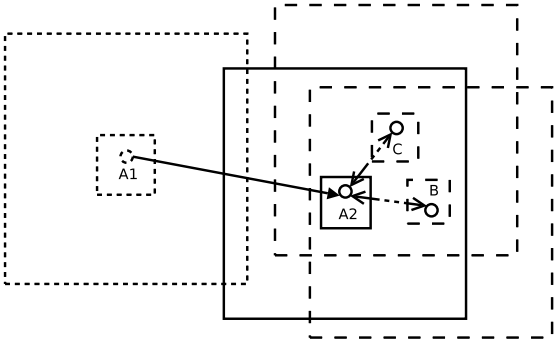


Figure 15: Inconsistency

Figure 15 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.

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 keeps 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. Note that this information is only from the node's 1-hop neighbours, and so the storage requirements will be kept to a minimum irregardless of the size of the network. In the event that a node recieves a bounding box from another node N, and then later another box from N, then the intersection of these two boxes is stored as the recorded bounding box

from N. This removes the problem of otherwise requiring large amounts of storage for the series of bounding boxes received from mobile nodes. 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.

8 Anchor-free motion

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 use RSL, because 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.1) for the node locations instead. Mass-spring models generally require more calculations than RSL, but in the absence of anchors, mass-spring models become a better option.

8.1 Motion detection

This is a simplified overview of our motion detection algorithm, for full details see Appendix A. 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 A.3).

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 A.4).
   In some cases, we will have chosen parent nodes that are unsuitable for placing C, and in these cases the algorithm will fail certain sanity tests. If this is the case, we then proceed to check other possible parent node pairs for suitability. If we cannot locate a suitable parent pair for a node, then so we resort to various alternative strategies (Appendix A.5) to place some of the remaining nodes before repeating the parent node testing.

3. The locations for the nodes are now further refined (Appendix A.6). Refinement is necessary because our initial configuration may well not be the most likely configuration of the nodes, as we do not take into account all of the links (Appendix A.2) 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 we moved.

For each pair of nodes which we will designate 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 A.7, Figure 16).

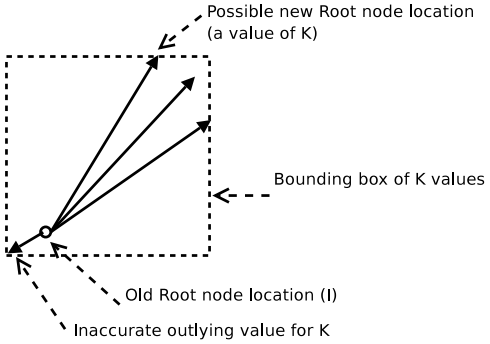


Figure 16: Calculating values for K

We now have a set of up to 4 possible locations for K which are checked against the known $R_{K,A}$ and $R_{K,B}$ values. The values that have correct ranges (at most 2 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 locations represents a "motion vector" (MV) for our root node. We can calculate MV from the

location of 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) as the vector between I and K. 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.

8.2 Results

We performed a series of experiments to test anchor-free mass-spring localisation, 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.

Our initial experiments tested our algorithm with two independent sets of "noisy" ranges, both generated from the "true" ranges. This generated range changes between the configurations up to twice the tested maximum error, as potentially the errors in each set could overshoot for one set and undershoot for the other set. Experimental tests [10] 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 adapted our experimental setup to mimic this.

We then took the topology and ranging information from the "true" locations, and added 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 A.3-A.6). 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 distances 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 A.7.

For all of the experiments, values 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 17, 18 and 19 show the results

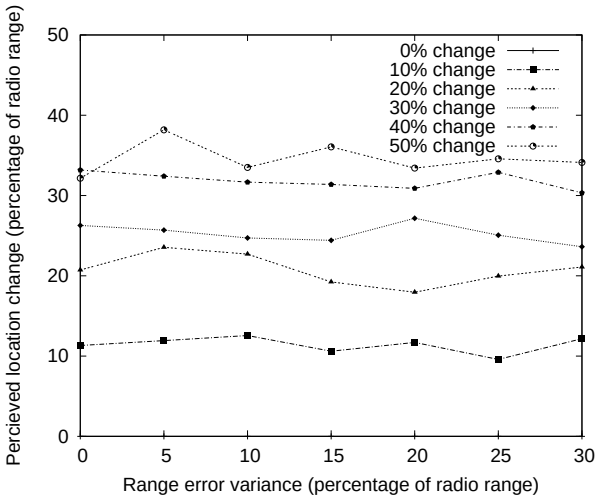


Figure 17: 0% second measurement inaccuracy

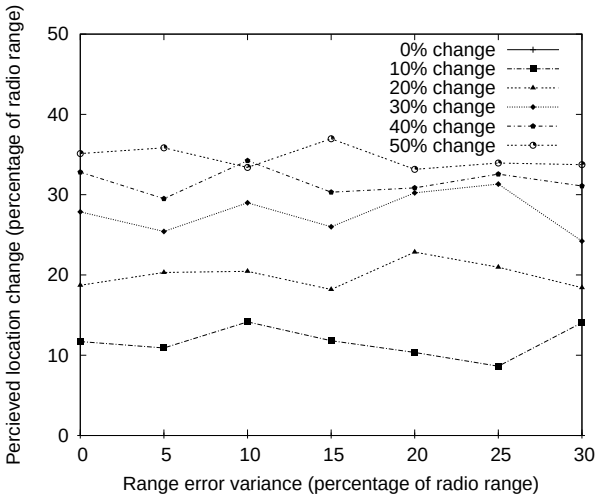


Figure 18: 5% second measurement inaccuracy

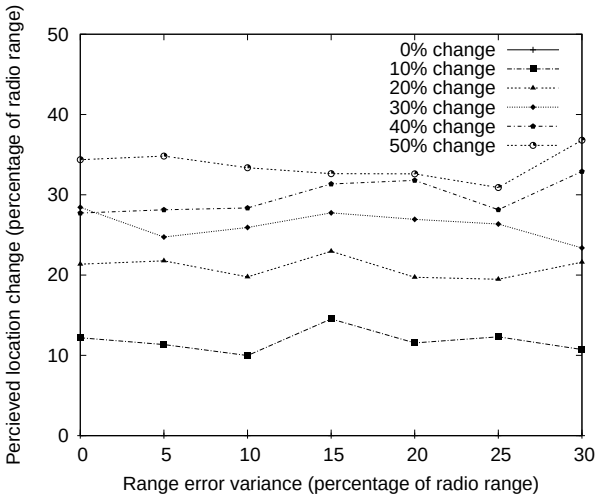


Figure 19: 10% second measurement inaccuracy

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 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 values that we have used here are similar to values shown in experimental testing ([10]).

9 Related Work

As previously mentioned, [14] was the initial work with statistic-based localisation and mobile beacons. They however required an order of magnitude more processing time, plus a far higher anchor coverage density to achieve similar results to RSL, and their system did not consider motion at all.

[4] did some earlier work using bounding boxes, with additional optimisations 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 [19, 20] indicate that even without such obstacles, bad links still occur in a significant percentage of cases.

[15] also looked at bounding boxes, but they relied on a very dense anchor distribution at all times, with only anchor nodes being allowed to transmit their bounding boxes.

[3] proposed a method for constraint-based localisation, including work with bounding regions. However, their techniques required several orders of magnitude more processing power than the methods proposed here and so they used centralised computation of their algorithm - rendering it unsuitable for large sensor networks due to the overheads of exchanging information with a central node. Their solution to the centralisation problem using a hierarchical distribution of the problem would still require much larger energy/computing resources than present on current node hardware, but now the additional capacity would have to be evenly spread across the network, reducing the feasibility of this technique even further. They also did not achieve significantly better results than the distributed algorithms demonstrated in this paper.

[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$\rightarrow$sink message routing) would be better off storing data locally and waiting for anchor information before transmitting.

[9] and [13] both 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.

10 Conclusions and Future Work

We presented here an approach that can provide good location information, even with non-uniform anchor distributions and considerable inaccuracies in the incoming ranging data. RSL provides a good solution to the problem of localisation even in small, resource-limited sensor

networks. We showed that we can calculate accurate position data for a high percentage of the sensor nodes in a network. We have improved both the quality and quantity of positioned nodes in sensor networks, both versus the earlier statistic-based method and deterministic localisation methods.

We have also shown that even in difficult localisation scenarios (such as the anchor-less scenarios for early stages of mobile anchor applications), even very limited information can be used, and that motion can be detected without knowing exactly where you are without additional hardware.

All of this has been tested using mobile anchor scenarios, which we have shown to be a realistic and usable method for the distribution of anchor data, as well as a cost-effective one - both in terms of energy costs for the sensor nodes of the network, and in terms of the necessary hardware required to create the sensor network. Getting rid of the errors in sensor measurements is hard to do well, but that is the price of gathering data from the real world. With statistical approaches, we have shown that it is possible to work around these errors, and derive good location information. Statistical approaches are somewhat more computationally expensive, but given the significant improvements in the location information, and that the computational expense results in a reduced level of required radio traffic during the localisation process (which increases the capability of other nearby nodes to do radio-dependent work efficiently during the localisation process), we believe that the trade-offs are worth it.

One of the long running problems in wireless sensor networks is how tightly integrated the different layers of the software should be - whether they should be heavily inter-dependent, or separate and modular. RSL, as shown here, is designed for a combination of these two approaches, allowing different layers of the application to communicate with each other to determine what is the best approach for the current application. To help achieve this, RSL provides a confidence value to its location information allowing applications to change their use of the location information depending on its accuracy.

In the future, we hope to expand on our work here to attempt to further improve the location 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.

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] L. Doherty, K. Pister, and L. El Ghaoui. Convex position estimation in wireless sensor networks. In *IEEE Infocom 2001*, pages 1655–1663, Anchorage, AK, April 2001.

[4] 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.

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

[6] 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.

[7] K. Langendoen and N. Reijers. Distributed localization in wireless sensor networks: A quantitative comparison. *Computer Networks, special issue on Wireless Sensor Networks*, (43):500–518, 2003.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] Yi Shang and Wheeler Ruml. Improved MDS-Based Localization, 2004.

[14] 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.

[15] S. Simic and S. Sastry. Distributed localization in wireless ad hoc networks. Technical Report UCB/ERL M02/26, UC Berkeley, 2002.

[16] Texas Instruments. *MSP430x1xx Family User's Guide*. SLAU049B.

[17] Eric W. Weisstein. Circle-Circle Intersection. http://mathworld.wolfram.com/Circle-CircleIntersection.html. From MathWorld - A Wolfram Web Resource.

[18] K. Whitehouse and D. Culler. Calibration as parameter estimation in sensor networks. In *First ACM Int. Workshop on Wireless Sensor Networks and Application (WSNA)*, pages 59–67, Atlanta, GA, September 2002.

[19] 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.

[20] 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.

A Motion detection algorithm

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" [19] effects. The techniques here have been influenced by [6].

A.1 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

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

A.2 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

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

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

A.3 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 can arbitrarily pick any of the non-root nodes as the reference node (2-hop neighbours could be used if we so wished), but ideally we need a node that is highly connected to the root node's immediate neighbours, which

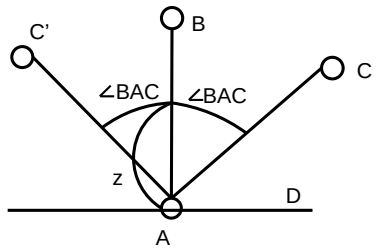


Figure 20: Placing C

will reduce the amount of calculations we need to do later on in many cases. 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 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.

A.4 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 $\overrightarrow{AB}$ to D

$n = \frac{A_x - B_x}{R_{A,B}}$, $z = sin^{-1}(n)$ where z is the angle of $\overrightarrow{AB}$ to D

Sanity assumption: $|n| \leq 1$

We can now calculate two possible values of θ (= angle of $\overrightarrow{AC}$ to D), using $\theta = z \pm \angle BAC$. We then have two possibilities for C's co-ordinates using the two values of θ and $C = (A_x + R_{A,C}cos(\theta), A_Y + R_{A,C}sin(\theta))$. These are shown on Figure 20 as C and C'. We choose the initial location of a node with the minimum amount of force (as defined in A.2) 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.

A.5 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 (A.3, but using only non-positioned nodes as possibilities), and place this newly selected neighbour at $(-\frac{\Sigma_p^{placed} p_x}{n}, -\frac{\Sigma_p^{placed} p_y}{n})$ 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 A.4. This is less accurate, but will still give us a reasonable first guess for the location.

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.

A.6 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, starting with the child nodes of the root node, then their child nodes and so on. 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.

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.

A.7 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.