Monte-Carlo Localization for Mobile Wireless Sensor Networks

Aline Baggio and Koen Langendoen

Delft University of Technology, The Netherlands
{A.G.Baggio, K.G.Langendoen}@tudelft.nl

Abstract. Localization is crucial to many applications in wireless sensor networks. This article presents a range-free anchor-based localization algorithm for mobile wireless sensor networks that builds upon the Monte Carlo Localization algorithm. We improve the localization accuracy and efficiency by making better use of the information a sensor node gathers and by drawing the necessary location samples faster. Namely, we constrain the area from which samples are drawn by building a box that covers the region where anchors' radio ranges overlap. Simulation results show that localization accuracy is improved by a minimum of 4% and by a maximum of 73%, on average 30%, for varying node speeds when considering nodes with knowledge of at least three anchors. The coverage is also strongly affected by speed and its improvement ranges from 3% to 55%, on average 22%. Finally, the processing time is reduced by 93% for a similar localization accuracy.

Keywords: Distributed localization algorithms, wireless sensor networks, mobility, Monte Carlo Localization, simulations.

1 Introduction

Many applications have a need for localization, be it for locating people or objects. Most of the time, data recorded from a wireless sensor only makes sense if correlated to a position, for example the temperature recorded in a given machine room or cold-store. Similarly, many end-user programs are location-aware, for example people would like to find the closest bus stop or mailbox, and emergency services need to localize persons to be rescued. In many cases, such as indoors, the Global Positioning System cannot be used. From now on, we will refer to a person, object or computer coupled with a wireless sensor to be localized as an *(unknown) node.*

This article presents a localization algorithm for wireless sensor networks specifically designed with mobility in mind. One important factor is to let the wireless sensors benefit from mobility and not only suffer from it. Literature [5,7,11,12,13,14] has shown that using mobile anchors in static wireless sensor networks helps improving the accuracy of the localization algorithm, as more nodes can benefit from the anchors' position broadcasts and as each node can hear more of these. Similarly, mobile sensors have a chance to get more information than in a fully static environment. The challenge, however, is that information in a mobile wireless sensor network gets invalidated more quickly if all the nodes are moving. Hu and Evans introduce a localization algorithm dealing with these different characteristics [9]. Their approach builds upon Monte Carlo Localization (MCL) methods used in robotics to locate a mobile robot. In this article, we present improvements to Hu and Evans' algorithm leading to better accuracy and lower computational cost when localizing nodes.

The remaining of this article is organized as follows. Section 2 presents some background information on localization mobile wireless sensor networks. Section 3 describes both our localization algorithm and Hu and Evans' algorithm it builds upon. Section 4 gives insight on the accuracy of the localization and efficiency of the algorithm. And finally, Section 5 concludes and gives some future work directions.

2 Dealing with Mobility

In the article [9], Hu and Evans present a range-free localization algorithm for mobile sensor networks based on the Sequential Monte Carlo method [4,8]. The Monte Carlo method has been extensively used in robotics [2,15] where a robot estimates its localization based on its motion, perception and possibly a prelearned map of its environment. Hu and Evans extend the Monte Carlo method as used in robotics to support the localization of sensors in a free, unmapped terrain. The authors assume a sensor has little control and knowledge over its movement, in contrast to a robot. A range-based version of the MCL algorithm has recently been proposed by Dil and al. [3].

Apart from the experiments with MCL, there are at the moment few localization protocols specifically designed with mobile wireless sensors in mind. Most of the papers presenting localization algorithms suggest that supporting mobility can be achieved by rerunning the localization algorithm after some time interval, either static or adaptable. While this is not optimal but feasible in some cases, the whole class of algorithms using information from distant nodes or iterative approaches will suffer from severe information decay. At the time the information reaches a distant node that wants to use it, it is very likely that the whole network configuration has changed. A node will therefore always calculate an inaccurate location, not due to the lack of information or to the intrinsic inaccuracy of the calculations it uses, but due to the way its localization algorithm gathers this information.

Mobility introduces a real-time component to the localization algorithms. Wireless sensor networks are usually considered delay-tolerant [6,11]. To the contrary, mobility makes a sensor network delay intolerant: information gathering and location calculation should happen in a timely manner, dependent on the speed of both the nodes and the anchors. This means that in a mobile wireless sensor network, methods relying on global knowledge such as calculating the number of hops or distances to all the anchors in the network are to be avoided.

Similarly, a mobile node cannot really benefit from iterative localization techniques where the location estimation is refined whenever a node receives more information from the network.

Besides possible information decay, a localization algorithm deployed in a mobile wireless sensor network should be able to cope with the temporary lack of anchors. In other words, the algorithms should be able to produce a location estimate in such conditions if the application layer has a need for it. In such cases, the location estimation could easily be tagged as uncertain, providing a mean for the application to assess how much the results of the localization algorithm should be trusted.

We believe mobility should be taken into account directly when designing new localization algorithms. A wireless sensor should benefit from mobility and exploit it to improve the efficiency of localization or get a better accuracy of its position estimates. The algorithms based on MCL are offering such guarantees. In the following, we build upon the range-free Monte Carlo Localization algorithm proposed by Hu and Evans [9] and show that by improving the way the anchor information is used, we can improve both the accuracy and the efficiency of the algorithm.

3 Localization in Mobile WSNs

This section first describes the basic MCL algorithm and then introduces our extensions.

3.1 Monte Carlo Localization

In [9], Hu and Evans define their localization algorithm as follows. The time is divided into discrete intervals. A sensor node relocalizes in each time interval. During the localization-algorithm initialization phase, a sensor picks a random set of N samples $L_0 = \{l_0^0, l_0^1, ..., l_0^{N-1}\}$, i.e. random localizations within the deployment area. From then on, the two steps, *prediction* and *filtering*, repeat. During the prediction step, a node picks random locations within the deployment area, possibly constrained by its maximum speed and the previous location samples. At time t, a sensor node thus generates a new set of samples L_t based on the previous set L_{t-1}. In practice, given a location l_{t-1}^i from L_{t-1}, a random location l_t^i is chosen from the disk of radius v_{max} around l_{t-1}^i, v_{max} being the maximum speed of a node. During the filtering phase, all impossible locations l_t^i are removed from the new set of samples L_t. The filtering occurs by using the position information obtained from both the one-hop and two-hop anchors. The one-hop-anchor group is composed of the anchors the sensor node heard directly. These anchors are assumed to be in the radio range r of the sensor node. The two-hop-anchor group is composed of anchors the sensor node did not hear itself but its one-hop neighbors did. These anchors are assumed to be in the range $2r$ of the sensor node but not within a radius r. In other words, MCL makes use of negative information. Note that this usually leads to an improved localization accuracy in an obstacle-free deployment area but is quite risky otherwise.

Note that after the filtering step, there may be less samples in the set than desired. The prediction and filtering process thus repeats until the desired number of samples is reached. The location estimate of a sensor at time t is the average of all the possible locations from the sample set L_t.

3.2 Monte Carlo Localization Boxed

Despite being quite accurate, especially in low-anchor configurations, MCL's efficiency can be improved. Drawing samples is a long and tedious process that could easily drain a lot of energy from a sensor node. Furthermore, the way MCL makes use of anchor information leaves room for improvement. Our version of the Sequential Monte Carlo Localization called Monte Carlo Localization Boxed (MCB) uses steps similar to those of MCL. The major differences lie in the way we use anchor information and the method we use for drawing new samples (see [1] for algorithmic details).

The original MCL algorithm uses information about one-hop and two-hop anchors at filtering time only, for rejecting impossible samples. In MCB, we use the information about the anchors heard to constrain the area from which the samples are drawn, method which we explain below. Reducing the area to sample from has two consequences. First, we draw good samples more easily and thus faster. Drawing good samples means that we have to reject samples less often in the filtering phase, reducing thereby the number of iterations the algorithm needs to fill the sample set entirely. The second consequence is implementation dependent. Unlike the pseudo-code shown in [9], the implementation of MCL sets a bound on the number of times a node can try to draw samples if its sample set does not contain the required number of samples yet. This boils down to avoiding that the algorithm loops endlessly if no valid sample can be drawn for a given configuration. In [9], Hu and Evans selected a sample-set size N of 50. A node tries at most twice 10,000 times to draw a sample. This happens once with a strict speed condition, drawing new samples from the disk of radius v_{max} around the old sample, and a second time with a relaxed speed condition, drawing new samples from the disk of radius $v_{max} + delta$ around the old sample. Drawing samples with a relaxed speed constraint only happens if the sample set is not full after the first series of 10,000 draws. After the 20,000 attempts, the sample set may still be not full, having less than 50 good samples. MCL does not try to fill the sample set any further. In MCB, we make sure that the sample set is as full as possible by drawing samples that do not have to be filtered and therefore do not require a redraw. In most cases, the sample set is full well before 10,000 tries. Experiences have shown that 100 attempts is ample enough to fill the sample set entirely. By ensuring that the sample set is full in 50 to 100 draws, a node can save precious battery power. Filling the sample set whenever possible also has a positive influence on localization accuracy over time.

The method used for constraining the area from which MCB draws samples is as follows. A node that has heard anchors – one-hop or two-hop anchors – builds a box that covers the region where the anchors' radio ranges overlap. In other words, this box is the region of the deployment area where the node is

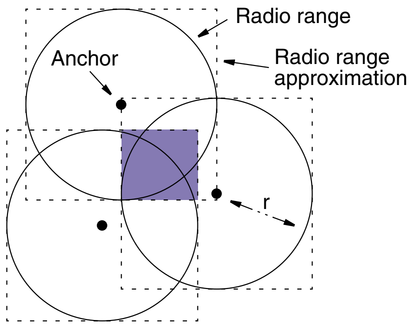

Fig. 1. Building the anchor box

localized. We call such a box the *anchor box.* Figure 1 shows an example of an anchor box (shaded area) in the case where three one-hop anchors were heard. For each one-hop anchor heard, a node builds a square of size $2r$ centered at the anchor position, r being the radio range. Building the anchor box simply consists in calculating coordinates (x_{min}, x_{max}) and (y_{min}, y_{max}) as follows:

$$x_{min} = \max_{j=1}^{n}(x_j - r) \qquad x_{max} = \min_{j=1}^{n}(x_j + r) \tag{1}$$

$$y_{min} = \max_{j=1}^{n}(y_j - r) \qquad y_{max} = \min_{j=1}^{n}(y_j + r) \tag{2}$$

with (x_j, y_j) being the coordinates of the considered anchor j and n being the total number of anchors heard. When considering two-hop anchors, we replace r by $2r$ in the above formulas.

In addition, in the simulation, the box-building algorithm cares for inconsistent or out-of-range boxes. In other words, for boxes where the minimum value x_{min} or y_{min} is larger than its respective maximum value x_{max} or y_{max}, the box is reset either to a box with one-hop anchors only, or to the whole deployment area. In the case where values are outside of the deployment area, for example x_{min} is negative, we reset the value to the coordinate of the border, in our example 0.

Once the anchor box is built, a node simply has to draw samples within the region it covers. Since the anchor box is a bare approximation of the radio range of the anchors, we keep a filtering step, as in the original MCL. And as in the original MCL, the prediction and filtering steps repeat until the sample set is full or until the maximum number of tries is reached.

Building an anchor box as described above is used in the case where the sample set is empty, for example at initialization time. In the case where we already have samples, the bounding box is built with an additional constraint, namely, for each old sample l^i_{t-1} from the sample set L_{t-1}, we build an additional square of size $2 * v_{max}$ centered at the old sample as follows:

$$x^i_{min} = max(x_{min}, x^i_{t-1} - v_{max}) \tag{3}$$

$$x^i_{max} = min(x_{max}, x^i_{t-1} + v_{max}) \tag{4}$$

$$y^i_{min} = max(y_{min}, y^i_{t-1} - v_{max}) \tag{5}$$

$$y^i_{max} = min(y_{max}, y^i_{t-1} + v_{max}) \tag{6}$$

where (x^i_{t-1}, y^i_{t-1}) are the coordinates of sample l^i_{t-1}. This updated box, which we call *sample box*, delimits per old sample the area a node can move in one time interval at maximum. Whenever a node has an initialized sample set but heard no anchor, we build the sample box solely based on the maximum node speed and the old samples. Box building remains a sequential process, where the anchor box is build first – and saved for subsequent uses – and updated independently for each old sample, creating thereby the sample box from which the new samples are effectively drawn.

Besides building anchor and sample boxes for drawing new samples, MCB tries to make the best possible use of all information a node received. This influences the localization algorithm in two ways. First, during the initialization phase or whenever the sample set becomes empty, MCB allows a node to use two-hop anchor information even if it has heard no one-hop anchor. Where the original MCL makes use of two-hop-anchor information only in combination with one-hop-anchor information during the filtering phase, MCB allows a node to use all information it got both at prediction and filtering time. This means that a node that heard only two-hop anchors can still draw samples using these and produce a location estimate.

Second, whenever a node has heard anchors and has an already initialized sample set but has failed to fill it (entirely) with new samples, MCB reverts to solely drawing new samples from the anchor box. In other words, the sample boxes are not used anymore. Not being able to fill the sample set typically happens when too many old samples are inconsistent with the current connectivity and speed constraints. To counter old sample inaccuracy and draw new valid samples, the algorithm would need to let the node travel a too long distance, i.e. more than what could be covered with speed v_{max} in one time interval, to finally meet the connectivity constraints. In such a case, MCL would try to draw new samples with a relaxed speed constraint $(v_{max} + delta)$. Drawing solely from the anchor box in MCB is equivalent to relaxing the speed. The advantage, however, is that no *delta* for the allowed speed increase has to be chosen in advance as it is the case with MCL.

4 Evaluation

In the following, we present the results of the simulations of three localization algorithms. We reused and extended the simulator used in [9]. First, we ran MCL as specified in [9] and presented above. MCB was implemented as described in Section 3. In addition, in order to compare with a well-known, simple and efficient localization algorithm, we chose to run the Centroid [10] algorithm in our simulations. Centroid calculates the position of an unknown node by computing the averages of all the x and all the y coordinates of the anchors heard.

The selected localization algorithms have been tested with simulated mobile wireless sensor networks. In the following, we assume a number of nodes and anchors deployed in an obstacle-free area of 500x500 units. We thus allow all algorithms to use negative information (see [1] for more on this topic). Both the nodes and anchors are mobile. The anchors know their location a priori, for example by using GPS. The radio range r is set to 100 units for both the anchors and the nodes.

A simulation run consists in feeding the simulator with a set of parameters such as the number of nodes in the network, the number of anchors, the maximum speed at which they move, the degree of irregularity used to model the radio communication. Time is discrete in the simulator. The speed of a node thus represents the distance in "units" a node can move per "time unit". For each selected maximum speed, the simulator generates a number of random network configurations, in our case 20. For each distinct network configuration, we simulate 200 time units. The first 100 units, the nodes move without localizing. For each subsequent time unit, the nodes first localize and then move. In other words, the time freezes and we localize the whole network using a snapshot. There is no movement while the nodes are localizing. This means that message transfer is instantaneous and that the received anchor locations are still accurate when a node receives them. As such, the simulation results represent a best-case scenario where no inaccuracy is introduced due to ongoing movement, communication delays, message loss or collisions, or other anchor-location inaccuracies (i.e. GPS error). As in [9], we use a modified random waypoint mobility model [16] where each node can vary its speed at each time step before it reaches its destination. The pause time is set to 0 and the minimum node speed is set to 0.1 to avoid speed decay [17]. The average node speed is close to $v_{max}/2$. In the following, speed is expressed as a multiple of the radio range r. Finally, as suggested in [9], we use a sample set of 50 location estimations.

To analyze the simulation results, we use the following metrics. First, we analyze the localization error. As done in [9], the localization error is calculated by measuring the distance between the real location of a node and its estimated location. Second, we consider the coverage of the different algorithms, that is the percentage of nodes that were able to calculate a location estimate. Third, we compare the processing times necessary for purely running each algorithm, thus excluding potential communications to gather anchor locations. Detailed network characteristics as well as other simulation results can be found in [1].

4.1 Localization Error

Figure 2 shows the localization error for all nodes, including the non-localized nodes, that is the nodes that are placed in the middle of the deployment area because they were not able to compute a location estimate (see [1] for more simulation results). Nodes can be non-localized for several reasons. First, they heard no anchor. This is typically the case with Centroid as it cannot produce a location estimate if no anchor is heard. In the case of MCL, this can happen at the beginning of the deployment when there is no previous sample set to build from. Second, in the case of MCL, a node that has heard anchors can sometimes still be non-localized. This happens when the algorithm is not able to fill the sample

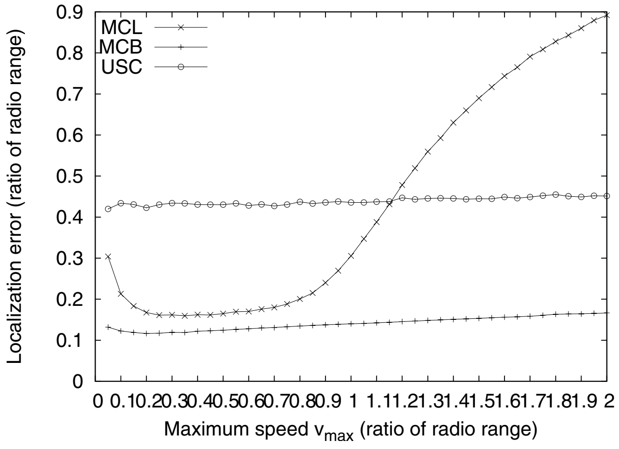

Fig. 2. Localization error (including the non-localized nodes)

set rapidly enough: the maximum number of random draws has been reached and the new sample set is still empty. This can be the case when the region to draw from is large and the area where the anchors' radio range overlap is small in comparison. Not being able to localize a node with anchors can also happen when the sample set becomes empty for some inconsistency reasons. Inconsistencies in a node's sample set generally occur after a period during which the node has heard no anchor. The new location estimations produced recursively from the old sample set gradually become less accurate as time passes and still no anchor is heard. Once an anchor is heard again, it can occur that all the new samples are rejected because they do not meet connectivity and speed constraints. Not being able to localize a node when anchors were heard is clearly unacceptable as it leads only to wasting energy and should be prevented as much as possible.

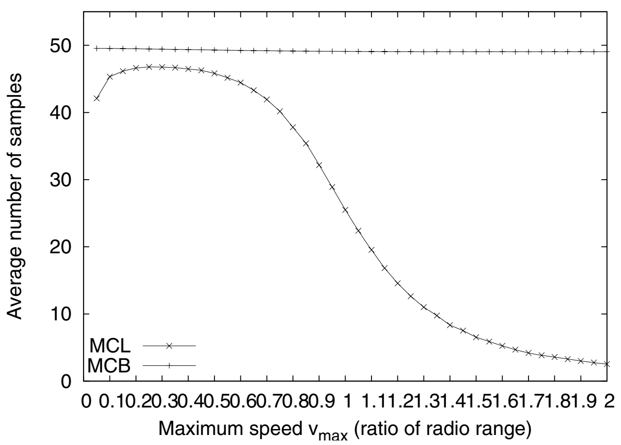

Fig. 3. Average number of samples

Figure 2 shows that MCL is rather sensitive to slow and high speeds while the curve for medium speeds, i.e. for a node moving at maximum between 20% and 70% of the radio range during one time interval, remains rather flat. The localization error for MCB as well as that of Centroid (USC) are rather independent of the node maximum speed and only show a slight deterioration of the accuracy as the maximum speed increases.

The behavior of the MCL localization error with respect to the maximum speed has several causes. Slow motion gives less chances to a node to hear anchors. More precisely, the average number of anchors heard remains quite stable as the speed varies, however, the time a node can remain anchor-less is on average longer. The reason for MCL's loss in accuracy for slow speeds is thus as follows: the slower an anchor-less node moves, the less chance it has to encounter a new anchor quickly since the whole network moves only in small steps at each time interval.

At slow speeds, nodes are thus more often producing location estimates without being able to use anchor locations. This increases the inaccuracy of the set of samples over time. In the worst case, nodes are localized in the middle of the deployment area if no valid sample can be drawn once an anchor is heard again. This effect is also noticeable in the coverage results (see [1]). The negative effect of slow motion was also observed by Hu and Evans in [9] for MCL.

For larger values of v_{max}, such as 0.8r and above, the motion of the nodes allows them to hear anchors more often and this limits the decay of the sample set. However, since the distance a node can travel in a time unit is larger, the area from which the random samples are drawn also increases. This affects the accuracy in a negative way. While the average number of anchors a node hears remains rather stable and the average number of anchor-less time intervals decreases as the speed increases, the average number of valid samples MCL is able to draw for high speeds considerably decreases. The coverage MCL achieves for high speeds is also decreasing. The impact of the maximum speed on MCL is thus purely due to the way the algorithm produces its samples. Hu and Evans also noticed this increase in inaccuracy in [9]. They did not provide a detailed study of the coverage and average number of samples though.

The behavior of MCB is not as dramatically affected by the maximum node speed as it is the case with MCL. The main reason is that the average number of samples the MCB variants can draw is rather stable with respect to speed as shown in Figure 3. This improves the coverage and the overall accuracy. This behavior is due to the more efficient way MCB draw samples.

4.2 Coverage

We studied the percentage of nodes that could be localized, i.e. for which a location estimate was produced, independently of how many anchors they heard. The coverage of Centroid is on average 96.62%. That of MCL is 92.13% on average, ranging from 96.87% (speed 0.45r) to 86.44% (speed 1r) down to 44.81%

(speed 2r). This comes from the fact that MCL is not able to draw enough good samples from a large draw area in which the overlap of the anchors' radio range is small. This occurs in general with high maximum node speeds. In the worst case, the new sample set remains empty leading to a non-localized node. The average coverage for MCB is extremely stable with respect to maximum node speed and stays around 99.98% (variation starts from the third decimal place).

4.3 Processing Time

Another factor positively influenced by the way MCB draws samples is the processing time, that is to say, the time needed by the algorithm to produce a location estimate. We consider here only the computation time and not the time needed for communicating (gathering anchor positions, listening to neighbors and forwarding anchor messages). Communication time is network-dependent and is identical for all the MCL variants. Only Centroid communicates less as it does not consider the two-hop anchors. We measured the processing time through simulation on a PC.

The processing time of the MCL variants depends on several factors. First, as the maximum number of samples N in the set grows, more samples have to be drawn and processing time also increases. There is of course a trade-off between the maximum number of samples in the set, the accuracy of the localization and the processing time. In [9], Hu and Evans provided an analysis of the impact of the maximum size of the sample set. We obtained similar results with our algorithm though the maximum number of samples can more easily be reduced with MCB than with MCL.

Second, the maximum number of random draws the algorithm is allowed to make also has an influence on the processing time. In the original MCL implementation, the maximum number of draws per sample is set to two times 10,000 draws for 50 samples, once with the maximum speed v_{max} and a second time with the relaxed maximum speed $v_{max} + delta$. More precisely, MCL allows two times 10,000 draws with an uninitialized sample set, and two times 200 draws per sample (maximum 50 samples) with an initialized sample set. With similar loop values, MCB is 40% to 50% faster while its accuracy and coverage are better than that of MCL and its average number of samples was higher (i.e. the sample set was full more often). These tests and those that follow were conducted using a 200 by 200 units deployment area with one unknown node and 32 anchors. The radio range was set to 50 units and the maximum speed to 50 units per time interval.

Thanks to its simplicity, Centroid performs much faster than any MCL variants. It runs in 0.0095% of the time needed by MCL and 0.0168% of the time needed by MCB when both MCL and MCB are using the original random-draw parameters (10,000 draws with an uninitialized sample set, 200 draws with an initialized sample set, 50 samples in the set at maximum).

Next, we compared MCL and MCB processing times for identical localization error when excluding the non-localized nodes. To get an identical accuracy, we varied the loop boundaries of both MCL and MCB. We kept the maximum number of samples unchanged (50 samples). In the case of MCL, we let the algorithm draw samples 1,000,000 times, once with the maximum speed v_{max}, and a second time with the relaxed maximum speed $v_{max} + delta$. For an initialized sample set, we allow 20,000 tries twice for each of the 50 samples. With MCB, we use a maximum of 100 draws for an uninitialized sample set. For an initialized sample set, MCB uses 50 draws from the sample box and, in the case of a partially full new sample set, it allows at maximum 100 extra draws from solely the anchor box. At speed 1r, MCL was able to produce 48.7227 samples on average and MCB 48.2153. The accuracy was 13.9% of the radio range for MCL and 13.8% for MCB. The coverage was 98.38% for MCL and 99.96% for MCB. The relative processing time was 100% for MCL and 6.238% for MCB. This demonstrates the fact that MCB is much faster than MCL for a similar localization accuracy. Even with a slightly lower average number of samples, the coverage of MCB is better than that of MCL.

5 Conclusion and Future Work

Localization in wireless sensor networks is a topic that has received much interest in the past years. Most proposed algorithms concentrate on static networks of sensors with either static or mobile anchors. The problem of localizing nodes in a mobile wireless sensor network has not yet received much attention although mobility needs to be taken into account at design time.

In this paper, we presented a localization algorithm that builds upon Hu and Evans' findings [9] and that makes Monte Carlo Localization more lightweight for use in wireless sensor networks. By making better use of the information a node gathers (one-hop and two-hop anchors) and by restricting the area a node has to draw samples from to a (small) box, we improve the whole process of localizing. The results of simulations of our algorithm, called Monte Carlo Localization Boxed, show that it allows a node to get an improved accuracy at a reduced cost. Most importantly, it ensures that a node having heard anchors will be localized and it will not pay a high price in term of processing time and energy expenditure because of the inefficiency of the localization algorithm (random draws). Our simulation results also show that the overall coverage of the localization algorithm is improved by ensuring that the sample sets are full as often as possible.

In the future, we are planning to deploy MCB on a test network of wireless sensors and study the behavior of the algorithm in a real-life setting. We will also make several extensions to the protocol so that it can benefit from extra information on the sensors' mobility patterns and mobility-pattern variability. This encompasses maintaining knowledge about sensors' speed and direction, possibly using additional equipment such as accelerometers and deploying it in heterogeneous networks using a mix of both mobile and static anchors.

References

1. A. Baggio. Monte-Carlo localization for mobile wireless sensor networks. Technical Report PDS-2006-004, Delft University of Technology, June 2006.
2. F. Dellaert, D. Fox, W. Burgard, and S. Thrun. Monte Carlo localization for mobile robots. In *IEEE International Conference on Robotics and Automation (ICRA99)*, Detroit, Michigan, USA, may 1999.
3. B. Dil, S. Dulman, and P. J. M. Havinga. Range-based localization in mobile sensor networks. In *Third European Workshop on Wireless Sensor Networks*, volume 3868 of *Lecture Notes in Computer Science*, pages 164–179, Zurich, Switzerland, Feb. 2006. Springer.
4. A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.
5. P. Dutta and S. Bergbreiter. Mobiloc: Mobility enhanced localization, Dec. 2003.
6. K. Fall. A delay-tolerant network architecture for challenged Internets. In *ACM SIGCOMM*, pages 27–34, Karlsruhe, Germany, Aug. 2003.
7. A. Galstyan, B. Krishnamachari, K. Lerman, and S. Pattem. Distributed online localization in sensor networks using a moving target. In *Third international symposium on Information processing in sensor networks (IPSN)*, pages 61–70, Berkeley, California, USA, Apr. 2004.
8. J. E. Handschin. Monte Carlo techniques for prediction and filtering of non-linear stochastic processes. *Automatica*, 4(6):555–563, July 1970.
9. L. Hu and D. Evans. Localization for mobile sensor networks. In *Tenth International Conference on Mobile Computing and Networking (MobiCom'04)*, pages 45–57, Philadelphia, Pennsylvania, USA, Sept. 2004.
10. D. E. N. Bulusu, J. Heidenmann. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5):28–34, October 2000.
11. P. N. Pathirana, N. Bulusu, A. V. Savkin, and S. K. Jha. Node localization using mobile robots in delay-tolerant sensor networks. *IEEE Transactions on Mobile Computing*, 4(3):285–296, May–June 2005.
12. R. Peng and M. L. Sichitiu. Localization of wireless sensor networks with a mobile beacon. In *First IEEE Conference on Mobile Ad-hoc and Sensor Systems (MASS 2004)*, Fort Lauderdale, FL, USA, Oct. 2004.
13. N. B. Priyantha, H. Balakrishnan, E. D. Demaine, and S. Teller. Mobile-assisted localization in wireless sensor networks. In *INFOCOM 2005*, Miami, FL, USA, Mar. 2005.
14. K.-F. Ssu, C.-H. Ou, and H. C. Jiau. Localization with mobile anchor points in wireless sensor networks. *IEEE Transactions on Vehicular Technology*, pages 1187–1197, May 2005.
15. S. Thrun, D. Fox, W. Burgard, and F. Dellaert. Robust Monte Carlo localization for mobile robots. *Artificial Intelligence*, 128(1–2):99–141, May 2001.
16. J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. In *IEEE INFOCOM*, San Franciso, CA, USA, March–April 2003.
17. J. Yoon, M. Liu, and B. Noble. Sound mobility models. In *ACM MobiCom*, pages 205–216, San Diego, CA, USA, Sept. 2003.