Monte Carlo Localization for Path-Based Mobility in Mobile Wireless Sensor Networks

Salke Hartung, Ansgar Kellner, Konrad Rieck and Dieter Hogrefe
Institute of Computer Science, University of Goettingen
Goldschmidtstr. 7, 37077 Goettingen, Germany
{hartung, kellner, krieck, hogrefe}@cs.uni-goettingen.de

***Abstract*—Localization is a mandatory requirement in Wireless Sensor Networks (WSNs). Many solutions focus on static networks and do not account for mobility. In this paper we present an extension of the Monte Carlo Localization method, which exploits the mobility behavior of certain applications in WSNs to reduce the localization error. Our approach called PO-MCL maps regular traveled paths of nodes to an internal grid, which is used to predict the node's behavior in the absence of seed nodes. We show by excessive simulations that our approach is able to reduce the localization error by up to 50 %.**[1]

I. Introduction

A. Localization in WSNs

Localization denotes the process of identifying the own position in space. Solutions for localization in WSNs can be classified into two groups of algorithms: range-based and range-free. Range-based algorithms require some sort of active sensing to determine distances to land-marks or anchor nodes while range-free algorithms are often based only on connectivity. A prominent example of a range-free solution is the Monte Carlo Localization algorithm (MCL) which originally has been proposed for localization of robots and was adapted for WSNs [1], [2]. The key idea of MCL is to represent the position of a node in the network using a sample set where each sample represents a possible location of the node. The average of all samples is the location estimation. The algorithm uses a filtering step to get rid of impossible locations. The filtering step is based on observations which are obtained from special nodes called seed nodes. These nodes are always aware of their position and broadcast the own location in regular intervals. MCL is famous for its simplicity and robustness compared to range-based solutions which often rely on either weak estimators such as the Received Signal Strength Indicator (RSSI) [3], [4], [5], [6] or require expensive hardware as in Angle-of-Arrival (AoA) determination [7], [8].

B. Motivation and Contribution

MCL is designed for applications in which all nodes are able to move freely in the deployment area. However, on closer examination in many applications one can assume that nodes are mainly moving on a set of paths. For instance, in wildlife monitoring animals often use a limited set of paths for moving in their territory. Prominent examples are big cats, the gnu migration, migratory birds [9], insect flight paths [10] or ant trails. Cars and trains only move on predefined routes, robots in warehouses can only move between shelves. More recently, biologists and neuroscientists show growing interest in the behavior of flying insects to study mating behavior, group dynamics, etc. [11]. Advances in the manufacturing size of modern transceivers [12] most likely are going to lead to sensor networks deployed in insect colonies formed by bees, wasps or hornets to gather completely new insights about these animals. In this paper we exploit this path-oriented mobility behavior and present an adapted solution based on the original MCL algorithm.

We use a prediction grid to divide the application area into cells to prognosticate the node's direction of movement. The grid is updated dynamically based on observations from seed nodes. We show that by using the grid we can achieve a better sample prediction and improve the weighting of the samples. As a side effect the prediction grid converges to the real paths taken by the nodes and can be used as a map representing the traveling paths of a node. This information is important to researchers using the sensor network for their specific applications. We implement our solution in Qualnet [13] and present detailed simulation results investigating several algorithm specific parameters in different path pattern scenarios. Our evaluation shows that our approach is outperforming MCL by up to $50\,\%$ and even reduces the localization error when applied in scenarios where all nodes move randomly (i.e. the prediction grid has no impact in these situations).

The rest of the paper is organized as follows. In Section II we define necessary terms, give information about our system model and revise the original MCL algorithm. Section III summarizes related work and shows existing approaches to improve MCL. Our algorithm is explained in detail in Section IV and we present our experimental results in Section V. Finally, we conclude our work in Section VI.

II. Definitions and Review of MCL

A. Terms

- Localization: The term localization refers to the process of determining the own position in space. Position and location are equally used in this paper.
- Seed nodes: A seed node is a node in the network capable of determining its position on its own, e.g. by using GPS, and is distributing this information at regular intervals as a broadcast message.

[1]Note to reviewers: We are aware that our paper exceeds the page limit, but want to stress that it is allowed to buy one additional page

1: **procedure** MCL
2: $L_t = \{\}$
3: **while** $|L_t| < N$ **do**
4: $R = \{l_t^i | l_t^i$ from $p(l_t|l_{t-1}^i)$, $l_{t-1}^i \in L_{t-1}\}$
$\forall\ i, 1 \leq i \leq N$
5: $R_{\text{filtered}} = \{l_t^i | l_t^i$ where $l_t^i \in R \wedge p(o_t|l_t^i) > 0\}$
6: $L_t = \text{choose}(L_t \cup R_{\text{filtered}}, N)$
7: **end while**
8: **end procedure**

Fig. 1. Original MCL algorithm [1].

- Ordinary nodes: All other nodes in the network which are trying to localize themselves are called ordinary nodes. They are the majority of the network and do not have the capability to locate themselves.
- Location announcement: A broadcasted message by a seed node including its own location and an identifier to be able to distinguish between multiple received location announcements.

B. System Model

The mobility model used in this paper is a combination of the Random Waypoint and the Manhattan Mobility Model [14], [15]. It is based on the Graph Mobility Model proposed in [16]. All nodes are allowed to travel on a set of predefined paths. Paths are represented by a connected graph $G = E \times V$ where E is the set of edges representing the paths and V is the set of vertices representing intersections or changes in direction of a path segment. Every segment can be directionally weighted to account for more frequently used paths or one-way paths. Nodes can choose a velocity in the interval $[v_{\text{min}}, v_{\text{max}}]$ at each $v \in V$ for the next path segment. The minimum order for all $v \in V$ is 1, i.e. the graph might have dead ends where nodes just can go back. While in the Manhattan Model nodes usually move on a continuous path, in our model nodes are always allowed to move back on the same segment they just came from.

All nodes are aware of their maximum communication range r. Communication is assumed to be of broadcast nature, i.e. a message sent by node n can be received by any node in a circular area with radius r around n. A node is also aware of its maximum velocity v_{max} and the dimensions of the deployment area $\dim_x \times \dim_y$. To measure orientation a magnetometer is used which can provide the angle α to magnetic North. Depending on α a node can determine its current direction of movement in terms of cardinal direction (i.e. N, NW, W, SW, S, SE, E, NE).

C. Revision of MCL

The MCL algorithm has been adapted from the area of robotics [2] and presented for the usage in WSNs by Hu and Evans [1]. In MCL a node's estimated location is represented by a set of weighted samples, L, where each sample l_t represents a possible location of the node at time t. The initial set, L_0, is selected by choosing random locations of the whole deployment area. A node will always maintain a fixed number of samples, N, to guarantee enough variability while still limiting the computational overhead. The MCL algorithm shown in Figure 1 computes the sample set L_t using the information of sample set L_{t-1} and observations of seed nodes, o_t, available at time t. To account for uncertainty about the node movement behavior the algorithm includes a *prediction step* (line 4) in which a new sample is drawn from a circular sampling area with radius $r_{\text{sarea}} = v_{\text{max}} \times t_{\text{check}}$ around its current position given by a transition equation $p(l_t|l_{t-1})$. In MCL v_{max} is the maximum velocity of a node and t_{check} is the time between two localization attempts of a node. The probability of the current location based on the previous location estimation is given by a uniform distribution [1], where $d(..)$ denotes the Euclidean distance between two samples as shown in Equation (1).

$$p(l_t|l_{t-1}) = \begin{cases} \frac{1}{\pi \times r_{\text{sarea}}{}^2}, & \text{if } d(l_t, l_{t-1}) \leq r_{\text{sarea}} \\ 0, & \text{if } d(l_t, l_{t-1}) > r_{\text{sarea}} \end{cases} \quad (1)$$

The generated set is put into the *filtering step* (line 5) which uses the observations o_t to filter impossible node locations from the sample set. Each node keeps track of its first-hop neighbor seeds S and of its second-hop neighbor seeds T. The filtering condition for a sample l is given in Equation (2).

$$\text{filter}(l) = \forall s \in S, d(l,s) \leq r \wedge \forall s \in T, r < d(l,s) \leq 2r \quad (2)$$

Samples passing the filter are assigned a weight of 1, all others a weight of 0. Samples with a weight of 0 are ignored in further re-sampling steps. If more than N samples have been generated, N random samples are chosen from the current set (line 6). The process is repeated until $|L_t| >= N$ (line 3). The final step is to compute the position estimation by calculating the weighted average of the sample set, i.e. the average of all samples with a weight of 1. For a more detailed description of MCL we would like to refer the reader to the original paper [1].

III. Related Work

Existing work on improvements of MCL mainly focuses on enhancing the filtering step either by introducing more sophisticated sample weighting or by defining more precise filtering conditions.

Rudafshani and Datta [17] take neighboring information into consideration for calculating sample weights. The authors present two versions of their algorithm: MSL* calculates weights for all of a neighbor's samples, while MSL tries to reduce the therefore emerging computational and communication overhead by calculating a single weight for every neighbor. A newly introduced closeness value is used to identify neighbors, which seem to have a good estimation of their own position and therefore provide more valuable information to the node, which tries to localize. The evaluation shows that using neighborhood information gives better results especially in situations where the seed degree is low.

Zhang et al. propose WMCL [18] (weighted MCL) which provides further improvements for the MCL algorithm family. A bounding box is constructed to reduce the area from which new candidate samples are drawn. In addition to that, WMCL

makes use of neighborhood position estimations to increase the localization efficiency. The authors state that their algorithm works better in static scenarios than previous solutions without tuning special parameters as done in MSL/MSL*. A real implementation on MICAz motes is provided in a small testbed to evaluate a static scenario.

Teng et al. [19] show how a single mobile seed node can be used to localize all nodes in a static sensor network in their approach called MA-MCL. The sensors are not expected to move after deployment. The seed node is the only node equipped with GPS and will move randomly in the whole deployment area. Since the unknown nodes do not move, they generate new samples based on their current position only in a static square area with side length β which is a tunable parameter of the algorithm and is standardly set to $\beta = 0.1r$ where r is the communication range of the ordinary node. The authors compare their work to the solution given in [17] and show that in these special scenarios their algorithm outperforms MSL and MSL*.

Additional sensor information is used by Hartung et al. [20] to reduce the localization error of MCL in situations where no seed information is available. With the help of an accelerometer and a magnetometer a rough approximation of the path a node travels when no seed information is available is recorded. The sample set is then moved relatively to the last known position estimation of the node. Simulations show that this strongly improves the accuracy in low seed degree situations.

Our approach presented in this paper differs from all mentioned works insofar as we are trying to avoid additional communication and computational overhead. Instead of relying on neighboring information we focus on the information directly available to a node.

IV. Path-Oriented MCL

Path-Oriented Monte Carlo Localization (PO-MCL) exploits the movement behavior of nodes in certain applications where nodes mostly move on a set of static paths, which are initially unknown to the node. Our aim is to reduce the localization error and to account for situations where no seed information is available.

A. Key Ideas

In PO-MCL all nodes are maintaining a prediction grid, which divides the whole deployment area into grid cells. A node is always located in exactly one of these cells. A grid cell has exactly eight neighboring cells except for the cells at the borders of the deployment area. Each of the neighboring cells is labeled with its corresponding cardinal direction (N, NE, E, SW, ..., NW). We assign a value to all grid cells, which represents the probability of moving to this cell next. Based on observations from seed nodes the grid is updated such that the probability of the cell the node has moved to is increased and the values of all other neighboring cells are decreased. As long as seed node information is available, the original MCL algorithm is executed except that samples are assigned the weight of their corresponding grid cell. Therefore, samples located in cells where the node currently is or has been before (i.e. cells corresponding to the path the node is moving on) will have a higher weight. In situations without seed information the node relies on the prediction grid information using an initial orientation determined by a magnetometer. The node will try to follow the path on the grid by looking for cells with high probabilities until seed information is available again.

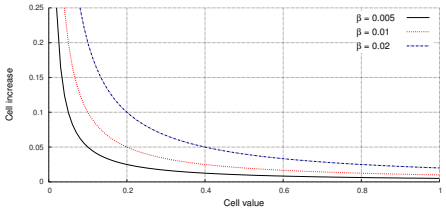


Fig. 3. Effect of different values for β.

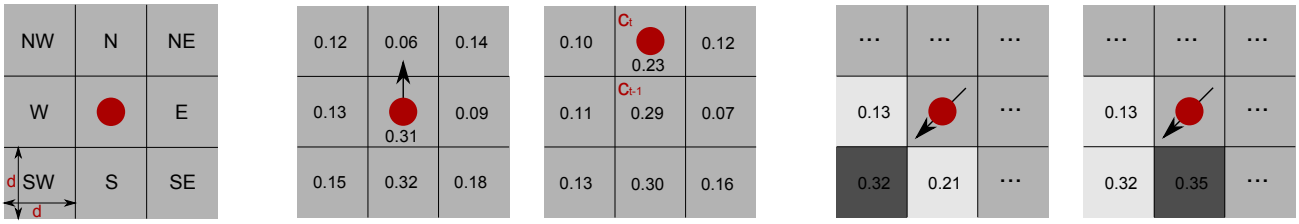
B. Prediction Grid Construction

The dimensions of the grid cells are an important parameter as they mainly decide over the memory overhead of PO-MCL. We are adapting the size of the grid cells based on $v_{\max}$ and t_{check}. Since the maximum distance a node can travel between two localization estimations is $d = v_{\max} \times t_{\text{check}}$ we also define the grid cell dimension as d as shown in in Figure 2a. It is obvious that with smaller values of d the resolution of the grid is growing and the traveled paths can be mapped to the grid with more precision. In the beginning all grid cells are initialized with the value $0.\overline{1}$, since no information about paths has been gathered yet and an arbitrary block of nine cells of the grid sums up to exactly 1. If a node can update its location estimation based on seed information, it checks if it has moved from its previous cell c_{t-1} to a different grid cell c_t. If yes, the probability of c_t is increased while the probability of all other neighboring cells of c_{t-1} is decreased. The amount of probability increase Δ_{inc} and decrease Δ_{dec} is determined based on the cell value of c_t using Equations (3) and (4) where β is a tunable parameter regulating the level of increase.

$$\Delta_{\text{inc}} = \frac{\beta}{\mathit{cellValue}} \quad (3) \qquad \Delta_{\text{dec}} = \frac{\Delta_{\text{inc}}}{8} \quad (4)$$

Cells with a low probability therefore will be increased faster than cells with high probabilities. A cell can have a maximum probability of 0.5 and we limit Δ_{inc} to be 0.2 at max. The effect of different values for β is shown in Figure 3. Smaller values of β will result in less probability increase and therefore slower grid convergence. An example of updating the grid is given in Figure 2b. Here, the node is moving North to c_t, therefore we increase the probability of c_t and decrease the probability of all other neighbors of c_{t-1} as described above.

Over time, the prediction grid will converge to a representation of the traveled paths of the nodes. An example of the convergence process is shown in Figure 4. Ideally, the values of all eight neighbors of a cell sum up to 1.0. However, since cells are affecting each other this is unlikely to happen. In this



(a) Grid directions. (b) Grid update process. (c) Grid movement prediction..

Fig. 2. Grid techniques in PO-MCL.

respect the term probability used in this paper is not entirely accurate, but used for the ease of presentation.

C. Grid Movement Prediction

In standard MCL without seed information the sample set L will degenerate more and more over time as uncertainty of the real position of the node is growing. In PO-MCL our constructed prediction grid can account for these situations as we have generated additional information about the node's behavior. Using a magnetometer we can determine the node's current heading. As the sensor readings might not be exactly correct and the prediction grid is only a very rough representation of the ground truth model, we choose not only the neighboring grid cell representing the direction determined by the magnetometer, but also its left and right neighbors. We then determine the highest probability of these three cells and move all samples by Δd in direction of the determined grid cell. Δd is calculated from the average of $v_{\min}$ and $v_{\max}$ as shown in Equation (5).

$$\Delta \mathrm{d} = \frac{v_{\min} + v_{\max}}{2} \times t_{\mathrm{check}} \tag{5}$$

Sample weights are reassigned based on the grid cells where the samples might have been moved to and the location estimation of the node is recalculated based on the updated sample set. The process is illustrated in Figure 2c: Assume the magnetometer determined a current direction of SW as indicated by the black arrow. The node selects its corresponding three neighboring cells and looks for the highest value. In the left example the result is consistent with the magnetometer direction (0.32). In the right example the cell in direction S has the highest probability (0.35). Therefore the predicted direction of the node indicated by the darker cell is S instead of SW.

D. Magnetometer Query Interval

Usage of a magnetometer is required for determining the heading of the node as soon as no more seed information is available. Since the magnetometer is consuming additional power it is desirable to use it as little as possible. Hardware tests in our experimental lab have shown that the time from powering on the magnetometer sensor to getting a first reading is negligibly low ($\approx$10 ms) [21] while GPS took almost 90 s to synchronize. Consequently, the magnetometer can be put into sleep mode and will be activated when required only. As the node might change its direction when reaching an intersection, it is necessary to query the magnetometer from time to time.

```
1:  procedure PO-MCL
2:      if |o_t| > 0 then
3:          MCL()
4:          magInterval = 0
5:          updatePredictionGrid()
6:      else
7:          if magInterval % magQuery == 0 then
8:              direction = getDirectionFromMagnetometer()
9:          else
10:             direction = getDirectionFromGrid(posOnGrid)
11:         end if
12:         moveSamplesByDirection(direction)
13:         magInterval++
14:     end if
15:     posOnGrid = determineGridCell()
16: end procedure
```

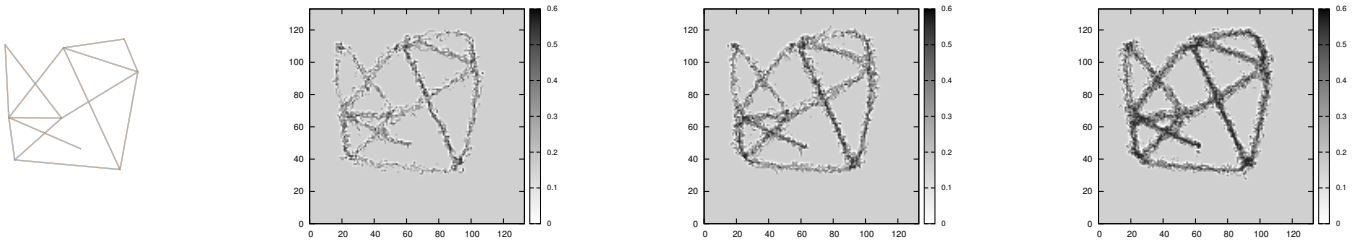
Fig. 5. PO-MCL algorithm.

The magnetometer query interval determines how often this is done. The most precise but also most power consuming solution would be to keep the magnetometer powered on. However, our simulation results show that it is possible to use the magnetometer only every 4th to 5th time while keeping a reasonable amount of precision. Detailed information on the trade-off between precision and magnetometer usage is given in Section V-D3.

E. Formal Description

In Figure 5 we show a pseudo code listing of PO-MCL. Depending on the circumstance if the node receives one or more location announcements o_t from seed nodes, either the MCL algorithm described in Section II-C will be executed or the grid will be used to update the sample set. In the former case the prediction grid is updated if the node has moved to a neighboring cell. As an additional improvement MCL has been slightly modified to weight samples according to the grid cell values. This is done by assigning the value of the grid cell where the sample is located to the sample weight.

The parameter *magQuery* determines how often the magnetometer is checked and *magInterval* is a simple counter to keep track of the number of executed localization attempts in periods where no location announcements are heard. It is always reset to zero as soon as MCL can be executed again. After the sample set has been updated in either way the new grid cell in which the node is located is determined.



(a) Ground truth. (b) After 360 min. (c) After 720 min. (d) After 1440 min.

Fig. 4. Grid convergence of a random path scenario.

V. Implementation Details and Evaluation Results

We implement and evaluate MCL and PO-MCL in Qualnet [13]. MCL and existing improvements have only been validated in a custom Java simulation software [1] provided by the authors of MCL. We strongly encourage researchers to use network simulation software well-established in research or industry to validate new algorithms in more reliable simulation environments. Custom network simulators often lack essential features like a full implementation of the protocol stack or path loss models for wireless transmissions and therefore might show distorted results. Nevertheless, we also modify the Java simulation environment used for evaluating MCL to support PO-MCL and receive similar results. The source code is publicly available [22]. Our code also provides a rudimentary GUI to visualize the ideas of MCL and PO-MCL which is suited for understanding the concepts of MCL and PO-MCL.

A. Mobility Model

Since Qualnet only provides a random waypoint mobility model we had to add our path-based mobility model described in Section II-B. ESRI shapefiles [23] are used as input files. A shapefile provides vector information for point, line and polygon data and is a common format used in geoinformatics. Consequently, shape files can be generated with a variety of tools including popular geographic information system software (GIS) like ArcGis [24]. The shapefile is parsed and represented by an internal graph data structure. When executing a scenario the mobility model generates a random path walk on the graph based on the configuration parameters, e.g. starting vertex, $v_{\max}$, pause time, etc.

B. Prediction Grid

The prediction grid is represented by a two-dimensional array of 32 bit floats. Depending on the grid cell size d and the size of the deployment area $\dim_x \times \dim_y$ the necessary memory to store the array is determined by $\frac{\dim_x \times \dim_y}{d^2} \times 32\,\text{bit}$. For instance, according to Section IV-B on a $1000\,\text{m} \times 1000\,\text{m}$ square with a maximum velocity of $v_{\max} = 1\,\text{m/s}$ and a localization interval of $2.5\,\text{s}$ a node would require $\frac{1000\,\text{m} \times 1000\,\text{m}}{(2.5\,\text{s} \times 1\,\text{m/s})^2} \times 32\,\text{bit}$ which is $\approx 625\,\text{kB}$. Although this might seem to be a large amount compared to what established research sensor mote platforms might provide, we argue that memory is no longer a strictly limiting factor in both size and financing.

C. Simulation Parameters

All experiments are conducted in a deployment area of $1000\,\text{m} \times 1000\,\text{m}$. The default parameters unless otherwise stated for all experiments are given in Table I.

In our simulations we investigate different path pattern scenarios, node velocities, numbers of seed nodes, magnetometer query intervals, radio ranges and quantities of maintained samples. All experiments use 200 ordinary nodes trying to localize themselves and mostly 25 or 50 seed nodes. We would like to point out that the number of seed nodes is set to a very low number compared to the size of the deployment area since we are interested in generating situations where no seed information has been acquired. Every experiment lasts 1 d (1440 min) to provide sufficient time for building the prediction grid. All experiments are repeated 10 times and averaged over all 200 ordinary nodes to get the final results.

$$\epsilon = \frac{\sum_{i=1}^{N} d(P_i, \hat{P}_i)}{N} \tag{6}$$

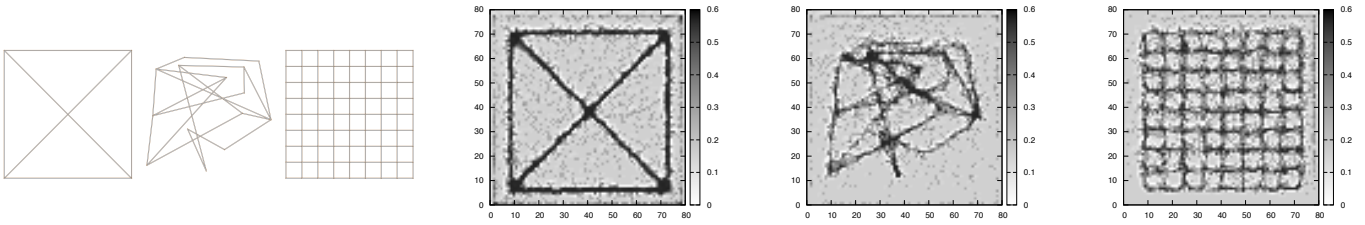
The localization error is given in multiples of r as the radio range is the main parameter for determining the absolute localization error (see Section V-D6). The error ϵ is given by averaging the error of all ordinary nodes as shown in Equation (6) where $d(..)$ is the Euclidean distance between the real position P_i and the estimated position $\hat{P}_i$ of a node and N denotes the number of ordinary nodes.

D. Simulation Results

1) Different Path Characteristics: We study the effect of four different path characteristics to see how well PO-MCL adapts to these scenarios. A square with diagonals is a simple test scenario and provides only four vertices and therefore only has limited expressive power. More realistic scenarios are the random path scenarios in which we generate a set of vertices and connect them using arbitrary edges. The last scenario is a grid with a cell size of $100\,\text{m}^2$ to test the behavior of PO-MCL in situations where a lot of changes in direction can be expected. The first random path scenario is already shown in Figure 4. The converged grids for the remaining scenarios are illustrated in Figure 6. The localization error for each scenario

TABLE I
Simulation parameters and default values

$v_{\max}$	2 m/s	t_{check}	2.5 s	N	25
$magQuery$	4	r	50 m		



(a) Ground truth. (b) Square scenario. (c) Random paths 2 scenario. (d) Grid scenario.

Fig. 6. Evaluation scenarios.

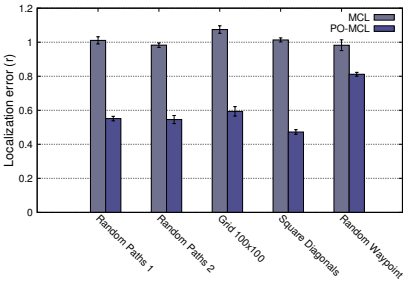


Fig. 7. LE for different shapefile scenarios.

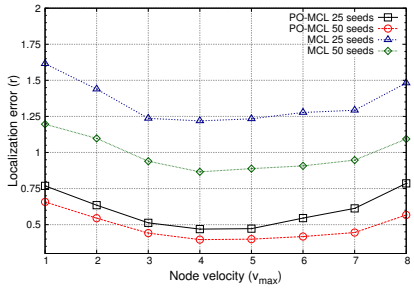


Fig. 8. LE for different node velocities.

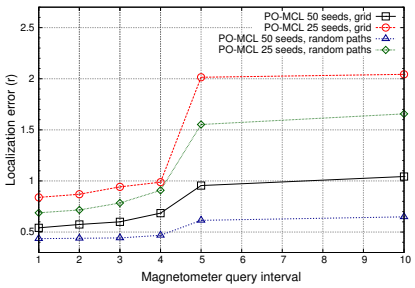


Fig. 9. LE for different magnetometer intervals.

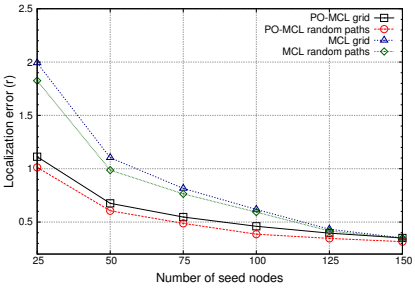


Fig. 10. LE for different number of seed nodes.

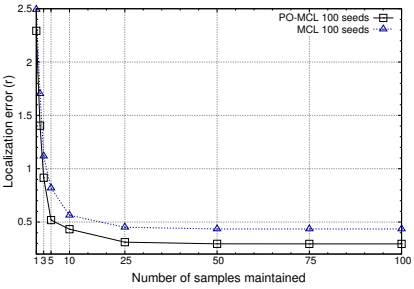


Fig. 11. LE for different sample set sizes.

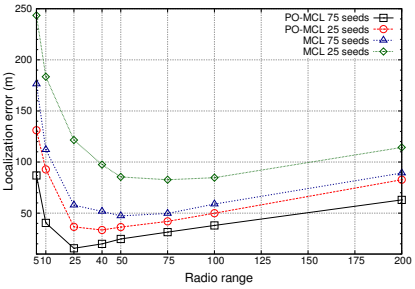


Fig. 12. LE for different radio ranges.

when executing MCL and PO-MCL is shown in Figure 7. It can be seen that PO-MCL outperforms MCL in all scenarios. Especially in the grid scenario, PO-MCL benefits from using the magnetometer and the prediction grid. On average, the localization error is reduced by about $40\,\%$ and even halved in the square scenario. Additionally we studied the behavior of PO-MCL if the random waypoint model is applied. Although PO-MCL cannot efficiently use the prediction grid in this case, it still benefits from its magnetometer.

2) Different Node Velocities: In Figure 8 we investigate different node velocities. Both algorithms benefit from an increasing node velocity in the beginning, since periods without seed information are getting shorter as nodes are moving faster. However, since the radio range is kept the same ($50\,\mathrm{m}$) for higher node velocities of $>4\,\mathrm{m/s}$ the localization error is growing as nodes lose contact to seed nodes more often. Depending on the radio range the local minimum of the curve might be found at a different node velocity, but the characteristics of the curve will be the same for other simulation parameters.

3) Magnetometer Query Interval: A very important parameter for the power consumption of PO-MCL is the magnetometer query interval, which describes how frequent PO-MCL will query the magnetometer to retrieve the node's orientation. The results are shown in Figure 9. The best results will be achieved, if the magnetometer is kept on all the time. In this case the grid is only used for sample weighting and not for predicting the movement of a node. When increasing the magnetometer query interval, only slight increase of localization error can be found up to values of 3 to 4. Mainly for the grid scenario, higher values result in increase of the error, because changes of direction happen more often than detected by the magnetometer. The characteristics of the curve heavily depend on the other parameters. If t_{check} is decreased while keeping the same $v_{\max}$ PO-MCL will be executed in shorter intervals and therefore the magnetometer will be queried more often while the same distance is traveled by a node. Ergo, for smaller values of t_{check} bigger magnetometer query intervals are possible.

4) Number of Seed Nodes: A crucial parameter for the overall precision of a localization algorithm is the number of seed nodes available to an ordinary node on average. In Figure 10 we show how both MCL and PO-MCL behave if the number of seed nodes in the scenario is constantly reduced. While the localization error for MCL tremendously increases, PO-MCL can compensate missing location announcements by using the magnetometer and its grid prediction techniques. If seed nodes are constantly available, the localization error

of MCL and PO-MCL will almost converge to a single curve, although PO-MCL still benefits little from its improved particle weighting.

5) Number of Samples Maintained: Computational time of both MCL and PO-MCL mainly depends on the number of maintained samples. There is a trade-off between the number of samples and localization error and it is desired to keep the localization error and the number of samples both as low as possible. In Figure 11 it can be seen that the localization error is rapidly decreasing when increasing the number of samples. This is due to the fact that larger sample sets can account for single imprecise samples. However, after a sample set size of 25 is reached there is no further improvement of the localization error. In contrast to Hu and Evens who recommend an optimal sample set size of 50 [1], we find this to be 25 which halves the computational overhead for the sample set.

6) Radio Range: Since MCL and PO-MCL both are connectivity-based algorithms the absolute localization error is mainly determined by the radio range r of the nodes. Smaller values of r will result in smaller absolute localization error given that a sufficient number of seed nodes is available. On the other hand with smaller r a bigger number of seed nodes is required to ensure the same level of seed node coverage. Figure 12 shows the absolute localization error when increasing the radio range.

VI. Conclusion

In this paper we presented PO-MCL, a localization solution for path-oriented mobility applications in WSNs. By using a prediction grid and a magnetometer to determine the initial heading we are able to improve the sample weighting of MCL and introduce an approach to predict the movement behavior of a node. Our results show that PO-MCL can adapt to arbitrary path characteristics and outperforms MCL while keeping a reasonable resource overhead. With our approach either the localization error or the number of seed nodes necessary to cover the network can be reduced. Our solution is suited for but not limited to applications where nodes are moving on predefined paths as it is often the case in wildlife monitoring, car traffic, robots or future sensor networks deployed in insect colonies which will strongly benefit from efficient localization solutions like PO-MCL.

In future work we will focus on combining our approach with existing solutions and on reducing the memory overhead of PO-MCL. Furthermore we intend to determine optimal values for the ratio of seed nodes and ordinary nodes and localization intervals.

Acknowledgment

This work has been partly supported by the Simulation Science Center project sponsored by the Volkswagen Foundation and the state Lower Saxony, Germany.

References

[1] L. Hu and D. Evans, "Localization for Wireless Sensor Networks," in 10^{th} *Annual International Conference on Mobile Computing and Networking (MobiCom 2004)*, Phildadelphia, USA, 2004, pp. 45–57.

[2] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *Proceedings of the IEEE International Conference on Robotics and Automation, 1999*, vol. 2, 1999, pp. 1322–1328 vol.2.

[3] S. Hartung, H. Brosenne, and D. Hogrefe, "Practical RSSI Long Distance Measurement Evaluation in Wireless Sensor Networks," in *2013 IEEE Conference on Wireless Sensors (ICWiSe)*, Kuching, Malaysia, 2013.

[4] R.-H. Wu, Y.-H. Lee, H.-W. Tseng, Y.-G. Jan, and M.-H. Chuang, "Study of characteristics of rssi signal," in *IEEE International Conference on Industrial Technology (ICIT), 2008.*, April 2008, pp. 1–3.

[5] O. G. Adewumi, K. Djouani, and A. M. Kurien, "RSSI Based Indoor and Outdoor Distance Estimation for Localization in WSN," in 14^{th} *IEEE International Conference on Industrial Technology (ICIT)*, 2013.

[6] Z. Zhong and T. He, "Rsd: A metric for achieving range-free localization beyond connectivity," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 11, pp. 1943–1951, Nov 2011.

[7] F. Bellili, S. Ben Amor, S. Affes, and A. Samet, "A new importance-sampling ml estimator of time delays and angles of arrival in multipath environments," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2014*, May 2014, pp. 4219–4223.

[8] S. Al-Jazzar, H. Strangeways, and D. McLernon, "2-d angle of arrival estimation using a one-dimensional antenna array," in *Proceedings of the 22nd European Signal Processing Conference (EUSIPCO), 2014*, Sept 2014, pp. 1905–1909.

[9] D. Anthony, W. Bennett, M. Vuran, M. Dwyer, S. Elbaum, A. Lacy, M. Engels, and W. Wehtje, "Sensing through the continent: Towards monitoring migratory birds using cellular sensor networks," in *ACM/IEEE* 11^{th} *International Conference on Information Processing in Sensor Networks (IPSN), 2012*, April 2012, pp. 329–340.

[10] J. R. Riley, U. Greggers, A. D. Smith, D. R. Reynolds, and R. Menzel, "The flight paths of honeybees recruited by the waggle dance," *Letters to Nature*, vol. 435, 2005.

[11] H. Sato, C. W. Berry, Y. Peeri, E. Baghoomian, G. L. Brendan E. Casey, J. M. VandenBrooks, J. F. Harrison, and M. M. Maharbiz, "Remote radio control of insect flight," *Frontiers in Integrative Neuroscience*, vol. 5, 2009.

[12] *Fractus Micro Reach Xtend Bluetooth, Zigbee, 802.11b/g WLAN Chip Antenna*, FRACTUS, S.A., October 2007.

[13] (2015, Feb.) Scalable Networks Incorporation. [Online]. Available: http://web.scalable-networks.com/content/qualnet

[14] T. Camp, J. Boleng, and V. Davies, "A Survey of Mobility Models for Ad Hoc Network Research," *Wireless Communications & Mobile Computing (WCMC): Special Issue On Mobile Ad Hoc Networking: Research, Trends And Applications*, vol. 2, pp. 483–502, 2002.

[15] E. Kulla, M. Ikeda, L. Barolli, F. Xhafa, and J. Iwashige, "A survey on manet testbeds and mobility models," *Computer Science and Convergence*, vol. 114, pp. 651–657, 2012.

[16] J. Tian, J. Hahner, C. Becker, I. Stepanov, and K. Rothermel, "Graph-based mobility model for mobile ad hoc network simulation," in *Proceedings of the* 35^{th} *Annual Simulation Symposium, 2002*, April 2002, pp. 337–344.

[17] M. Rudafshani and S. Datta, "Localization in wireless sensor networks," in 6^{th} *International Symposium on Information Processing in Sensor Networks (IPSN), 2007*, 2007, pp. 51–60.

[18] S. Zhang, J. Cao, C. Li-Jun, and D. Chen, "Accurate and energy-efficient range-free localization for mobile sensor networks," *IEEE Transactions on Mobile Computing*, vol. 9, no. 6, pp. 897–910, 2010.

[19] G. Teng, K. Zheng, and W. Dong, "MA-MCL: Mobile-Assisted Monte Carlo Localization for Wireless Sensor Networks." *IJDSN*, vol. 2011, 2011.

[20] S. Hartung, S. Taheri, and D. Hogrefe, "Sensor-assisted monte carlo localization for wireless sensor networks," in *IEEE* 4^{th} *Annual International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER), 2014*, June 2014, pp. 219–224.

[21] *AK8975/AK8975C 3-axis Electronic Compass technical manual*, Asahi Kasei Microdevices Corporation, 2015.

[22] (2015) PO-MCL source code. [Online]. Available: http://user.informatik.uni-goettingen.de/ shartun/pomclsim.zip

[23] *ESRI Shapefile Technical Description*, Environmental Systems Research Institute, July 1998.

[24] (2015, Feb.) ESRI ArcGIS website. [Online]. Available: http://www.esri.com/software/arcgis