Strategies to overcome Border Area Effects of Coarse Grained Localization

Ralf Behnke[1], Jakob Salzmann[1], Ralf Großmann[1], Dominik Lieckfeldt[1],
Dirk Timmermann[1], Kerstin Thurow[2]

[1]Institute of Applied Microelectronics and Computer Engineering
University of Rostock, Germany
[2]Center for Life Science Automation,
18119 Rostock, Germany

{ralf.behnke, jakob.salzmann, ralf.grossmann, dominik.lieckfeldt, dirk.timmermann}@uni-rostock.de
kerstin.thurow@celisca.de

***Abstract*— Localization of sensor nodes is one of the key issues in Wireless Sensor Networks. Next to the ability, to assign a phenomenon to a position, localization is a precondition for sensor network algorithms like geographic clustering and routing. A simple approach for coarse grained localization is Centroid Localization (CL), which was firstly presented by Bulusu et al. and assumes regularly arranged beacons. The localization accuracy was improved by various centroid-based algorithms, which use approximate distances to improve location estimation through weighting beacons in range, e.g. Weighted Centroid Localization (WCL). Nevertheless, all these approaches have in common an increased localization error near network borders. In this work, we investigate this error and present two strategies to reduce the localization error of border area nodes.**

***Index Terms*— Wireless Sensor Networks, Centroid Localization, Optimization**

I. Introduction

Recent technological advances led to the development of tiny wireless devices, which are able to sense the environment, compute simple tasks and exchange data among each other. Interconnected assemblies of such devices, called Wireless Sensor Networks (WSNs), are commonly used to observe large inaccessible areas [2]. For the majority of WSN scenarios the collected data need to be combined with geographic information to make it useful. Moreover, localization in WSNs forms the basis for geographic clustering [9] as well as geographic routing [1, 3]. Due to existing limitations in terms of size and energy consumption, localization within the network is preferred over utilizing a commercial positioning system like GPS [8]. Consequently, a typical approach is to assume that only a small number of selected nodes know their exact position a priori or gets it from common positioning systems. Then, all other nodes calculate their position with the help of these beacon nodes.

The accuracy of localization techniques ranges from high precision, when the location estimation is based on the solution of nonlinear distance equations, to low precision methods, which are sufficient for a lot of applications and additionally easy to calculate on computational constrained sensor nodes.

Bulusu et al. divided localization into coarse grained localization and fine grained localization and proposed a coarse grained localization algorithm, which needs only a minimum of computations, called *Centroid Localization* (CL) [7]. In CL, all non-localized nodes calculate their position as the centroid of the beacon's positions within their communication range, regardless of the distance or signal strength. A number of algorithms like *Weighted Centroid Localization* (WCL) [6, 10] and *Adaptive WCL* (AWCL) [4] have been published. They extended the idea of CL by using the distance towards a beacon in range to quantify the beacon position and emphasize nearer beacons, but without using it for exact calculations of the unknown position.

It is a common knowledge, that the accuracy of localization typically increases the more beacon nodes are used. This is achieved by choosing the communication range much higher than the distance between beacons. Beneficiaries of this approach are nodes in the central area of the network which therefore receive as much beacon nodes as possible. Nodes near the network border receive only few beacon nodes. Therefore these nodes have a higher localization error. In this work, we consider the relative localization error, calculated as the quotient of the real localization error, i.e. the distance between estimated position and real position, and the distance between two beacon nodes. This is given in equation (1), where $\bar{e}$ is the relative localization error, B is the beacon

distance as shown in Fig. 1 and $\vec{p}$ is the real position and the estimated position, respectively .

$$\bar{e} = \frac{\left|\vec{p}_{real} - \vec{p}_{estimated}\right|}{B} \quad (1)$$

In this paper, we first investigate the localization error in the border area and afterwards present two strategies which can be used to reduce the error in that area. We analyze the impact of our approaches regarding three centroid based localization algorithms, i.e. CL, WCL and an adaptation called *Modified WCL* (MWCL).

The remainder of the paper is organized as follows. In Section II the localization algorithms used, i.e. CL, WCL and MWCL, are reviewed. A detailed discussion concerning the causes for the increased localization error near the network borders follows in Section III. In Section IV, we briefly introduce a border area detection algorithm which constitutes a prerequisite for our improvement strategies, presented in Section V. Section VI analyses the gain we achieve with the presented approaches. Finally, Section VII closes with a conclusion and future work.

II. Related Work

This section briefly reviews the original CL, which was published by Bulusu [7], as well as the existing adaptations WCL [6, 10] and MWCL [5].

The basis of all presented approaches is a couple of sensor nodes, which differ from normal nodes as they are aware of their own position. These nodes will be called beacon nodes. They can be used by other sensor nodes to estimate their position. For our analyses the beacon nodes are arranged in a regular grid. The basic Algorithms are explained for a grid consisting of 2 x 2 beacon nodes. Further analyses with emerging border areas are conducted for 5 x 5 beacons.

A. Centroid Localization

The basis of CL, as well as of other CL based algorithms like WCL [5], is the square deployment of a couple of beacons. The pure CL is the simplest of all CL based algorithms. It does not utilize the Received Signal Strength Indicator (RSSI) or any other parameter, indicating the distance between a beacon node and a normal sensor node, which is referred to as unknown. The only kind of distance information used in CL is the binary information whether the unknown is in the communication range of a beacon or not. CL assumes that each beacon has a circular area within it can communicate with other nodes, known as unit disc graph model. In Fig. 1 it is shown how the communication ranges of four beacons, arranged as described above, build up to thirteen intersection areas, within which an unknown node can be localized.

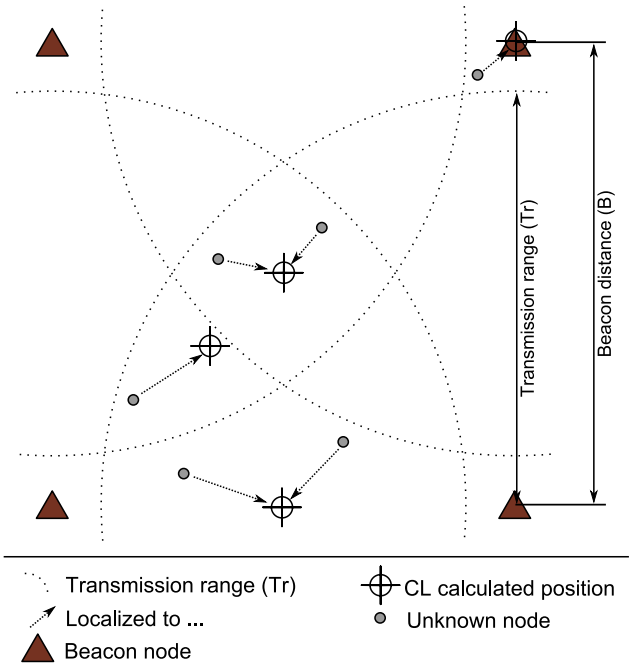

Fig. 1. Basic principle of the CL-algorithm. Nodes assign themselves to a cell by listening to messages from the beacons. Then, a node localizes itself at the estimated centroid of the according cell.

The algorithm uses the location information of all beacons in range to calculate the position as the centroid of the received beacon positions, as shown in (2). In this formula, $P_i(x,y)$ indicates the position of unknown i given by its two dimensional coordinates. The known position of beacon j is given by $B_j(x,y)$. The number of beacons which are within the communication range of the unknown node is indicated by m.

$$P_i(x,y) = \frac{1}{m}\sum_{j=1}^{m} B_j(x,y) \quad (2)$$

As indicated by Fig. 1, a node which is situated within one of the intersection areas will calculate its position at one single point, regardless of its exact position within the intersection area. For each intersection area an own localization point exists which is the centroid of the beacon positions in range. For example, a node which is able to communicate to all of the four beacons would determine its position as the centre of the arrangement. In contrast, a node which has only two beacons in range would calculate its position on the half of the edge that connects the two beacons. This behavior leads to a relatively high localization error, given as the Euclidian distance between the exact position of a sensor node and its calculated position.

B. Weighted Centroid Localization

The low accuracy in location estimation of CL has motivated the development of WCL. WCL introduced the quantification of the beacons depending on their distance towards the unknown node. The aim is to increase the impact

of those beacons which are nearer to the unknown. As the RSSI as well as the Link Quality Indicator (LQI) also increases with a decreasing distance it can be selected as an appropriate quantifier.

The basic idea of WCL, published in [5], is to quantify each beacon's position with a quantification function that uses the distance from an unknown node towards each beacon in range. The quantifier is described as shown in (3), where w_{ij} describes the quantification for beacon j used by node i. The distance between beacon j and node i is given by d_{ij} and g denotes the *degree*, that regulates the influence of far away beacons.

$$w_{ij} = \frac{1}{\left(d_{ij}\right)^{g}} \tag{3}$$

Using this weighting the initial equation (2) has been expanded for WCL as shown in (4).

$$P_i(x,y) = \frac{\sum_{j=1}^{m}\left(w_{ij} \cdot B_j(x,y)\right)}{\sum_{j=1}^{m} w_{ij}} \tag{4}$$

An appropriate value for g was found in g=2. Therefore the term WCL in the paper in hand means WCL with degree two. By the way, WCL with a degree of zero is equal to CL.

C. Modified WCL

The basic idea of MWCL is close to WCL. Both algorithms differ in the function used for quantification. In contrast to WCL, MWCL uses the nodes transmission range Tr in addition to the approximated distance d_{ij}. To obtain a quantifier that is low for far away beacon nodes and high for near beacon nodes, equation (5) is used to define the influence of beacon j on node i.

$$w_{ij} = Tr - d_{ij} \tag{5}$$

This weighting is used as usual in equation (4). Due to the mentioned quantification used in MWCL, localization is less complex in comparison to WCL.

III. Localization Error of Border Areas

As it was shortly described in the introduction, nodes which are situated near the network border have a higher localization error than nodes which are far enough from the border. Fig. 2 illustrates the distribution of the localization error within a sensor network, consisting of 5 x 5 beacon nodes. The adjusted transmission range is chosen as twice the distance between two beacons. For simplicity, the unknown nodes have been localized using CL.

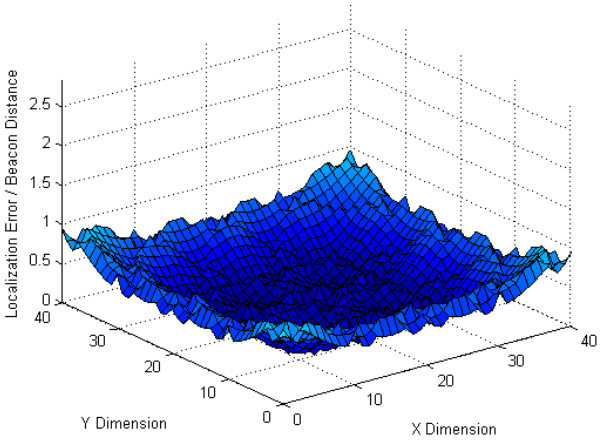

Fig. 2. Localization error distribution for CL localization within the sensor network field shows an increase near the border. Transmission range was set to twice the beacon distance of 10 units.

As depicted by Fig. 2, the application of the existing approaches leads to the behavior that a border area emerges, which has an increased localization error, compared to the area within the network's core. Two reasons for this decrease in localization accuracy exist and are examined in the following.

The first handicap for nodes near the border is simply the reduced number of beacon nodes within transmission range. Fig. 3 illustrates two representative nodes within the described sensor network. The nodes transmission range is illustrated as circles around the nodes. It can be seen that the node close to the border have to calculate its position with only eight beacon nodes, while the other one uses twelve beacons for its

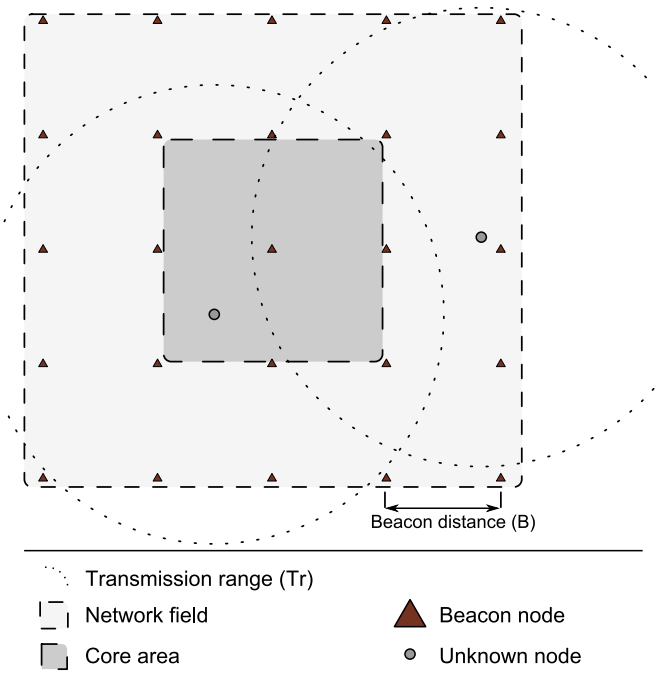

Fig. 3. Representatives of nodes within a sensor network. One within the core area, having twelve beacons in range, and one near the border, having eight beacons in range. Transmission range was set to twice the beacon distance.

calculations. Considering only the decreased number of beacons, the localization error should become a value between those in the core area and that which would be achieved with a low adjusted transmission range, i.e. a low number of beacons in range due to low transmission range. However the localization error at the border is much higher. This is caused by the second, more important handicap.

All CL based localization algorithms have in common, that a beacon in range virtually pulls the unknown node, i.e. the node to locate, towards its position. CL does this with the same strength for each beacon in range, regardless of its distance to the unknown. WCL and its derivates are acting the same way but with a distance dependent strength. Looking ones more at Fig. 3, one can identify that the border node not only uses less beacons, but also that the beacons used have an unbalanced distribution. There are more beacons which pull the unknown node towards the network center, than those pulling it to the boundary. Therefore the improvement in accuracy which is achieved in the network core by using higher transmission ranges comes along with impairment near the border. The size of this border area is growing with the used transmission range.

IV. Border Detection Algorithm

An important precondition for handling the increased localization error near the border is to find out whether a node resides near the border. A useful algorithm for this challenge has been published by Blumenthal [5]. This is utilized in our approaches which are trying to minimize the described border effects. The detection algorithm works as follows.

The algorithm utilizes neighborhood knowledge of the beacon nodes. Each beacon is able to communicate with other beacons in its range. Also the distance between two adjacent beacon nodes is given a priory. Therefore it is assumed that each beacon can check whether in each direction, called north, east, south and west, its direct neighbor exists or not.

Firstly, to find out whether the node is near the border, it counts the beacon nodes it is able to receive. Through comparison with a specific threshold for nodes in the network core, the node itself can decide whether it resides in the core or not. To find such a threshold, a beacon density is defined as the number of beacons within the network over the whole network area. This density can be multiplied by the communication area of a node to find the mean number of beacons in range of a node situated in the center. The communication area is assumed as a circular area around the node with its transmission range as radius. In the same way the mean number of beacons in range of a border node can be approximated. Both approximations are given in equation (6), where *tb* is the total number of beacons within the network and *A* is the network area. The threshold to detect a border node was chosen as br_{border}, knowing that the number of beacons in range of a node depend on its position.

$$br_{center} = \left\lfloor \frac{tb}{A} \cdot \pi \cdot Tr^2 \right\rfloor$$
$$br_{border} = \lfloor 0.5 \cdot br_{center} + 1 \rfloor \quad (6)$$

The next and more important step is to find out where within the border area the node is situated. This is necessary because, as described further, nodes will be virtually pulled towards the network center. The knowledge about the approximated position within the border area provides knowledge of the direction in which the node is pulled and therefore indicates the direction in which a potential correction should be made. For this reason, the whole network is divided into nine parts, i.e. center, upper left, top, upper right, left, … and lower right. A node outside the center would try to find out to which of the other parts it belongs to. Therefore it gets additional knowledge from beacon nodes.

Via communication each beacon finds out whether it has a direct neighbor in each direction. This information is saved in a vector of four binary values, one for each direction. The value is set to logical 1 if a direct neighbor exists and to logical 0 otherwise. This information is send to the nodes in addition to the beacons position. A node trying to locate itself sums the vectors of the beacons he received and gets a new four dimensional vector. If all components of this vector are equal, no beacon is situated at the border and therefore no correction is needed. Otherwise a small value of one of the components denotes that the node is near a border and indicates the direction in which a correction should be made.

A sample application of the border detection algorithm is presented in Fig. 4. The picture shows an example of three Beacon nodes, one in a central position, one at the edge and one in a corner. Also the different vector assignments for these beacons are illustrated in this figure. Taking both

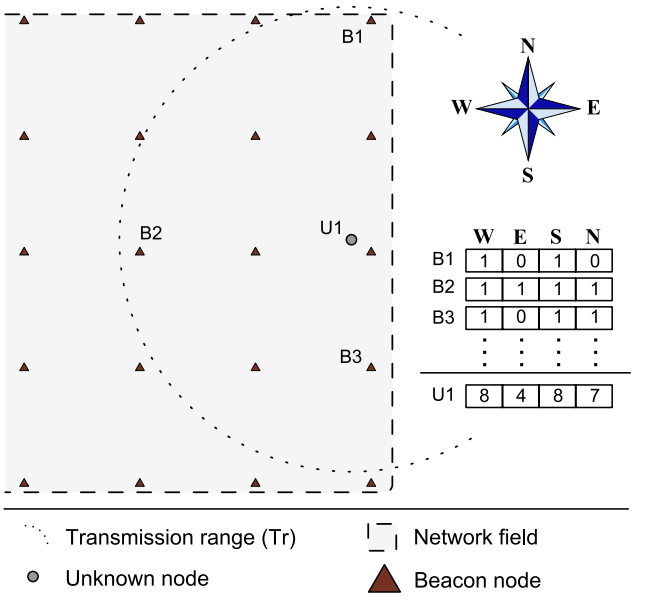

Fig. 4. Neighborhood vectors transmitted by beacon nodes. Illustrated beacon nodes show characteristic cases.

concepts together, i.e. the number of beacons in range and the summed neighborhood vector, the border detection works as follows: A node is classified as a border node, if at least one of the components of the summed vector is less than the threshold for border nodes. In addition, this component has to be smaller than the component for the opposite direction. In the case that this applies to two components instead of one, the node resides in a corner.

V. Strategies to Overcome Border Effects

As described in Section III there are two facts causing an increased localization error on nodes within the border area: Reduced number of beacon nodes and virtual pull towards the network core because of the strength of the residual beacon nodes. Both effects are caused by the unbalanced distribution of beacon nodes around such border nodes. The following approaches try to mitigate this error sources by creating virtual beacons outside the network area and by ignoring existing beacons, respectively.

A. Virtual Beacon Creation

Our first approach is to create virtual beacons. That means the node itself generates additional beacon information outside the network area. This can be done because of the regular, i.e. square, distribution of beacon nodes. If a node detects that it resides at one edge, it firstly generates as many virtual beacons as would be needed in the worst case, i.e. when the node resides exactly on the border. The number of at most needed virtual beacons depends on the transmission range and can be computed as described in (7), where *Tr* means the transmission range and *B* means the beacon distance as illustrated in Fig. 1. This equation distinguishes between nodes along the edge ($\Delta_{\parallel}$) and those down to the edge ($\Delta_{\perp}$).

$$\Delta_{\parallel} = \left\lceil 2 \cdot \frac{Tr}{B} + 1 \right\rceil$$
$$\Delta_{\perp} = \left\lfloor \frac{Tr}{B} \right\rfloor \quad (7)$$

The described number of virtual beacons has to be generated from a particular starting point. Nodes that have been classified as residing at one of the corners can use the corner to arrange the virtual nodes around it. Nodes classified as edge node are not affected by an increased localization error along the detected edge and can therefore use this unaffected coordinate as central point around which virtual nodes have to be arranged. A sample arrangement is illustrated in Fig. 5.

In the second step the node has to decide which of the generated virtual beacons are in its transmission range. The criterion for this decision is the distance between the virtual beacon and the node, which generated the virtual beacon. If

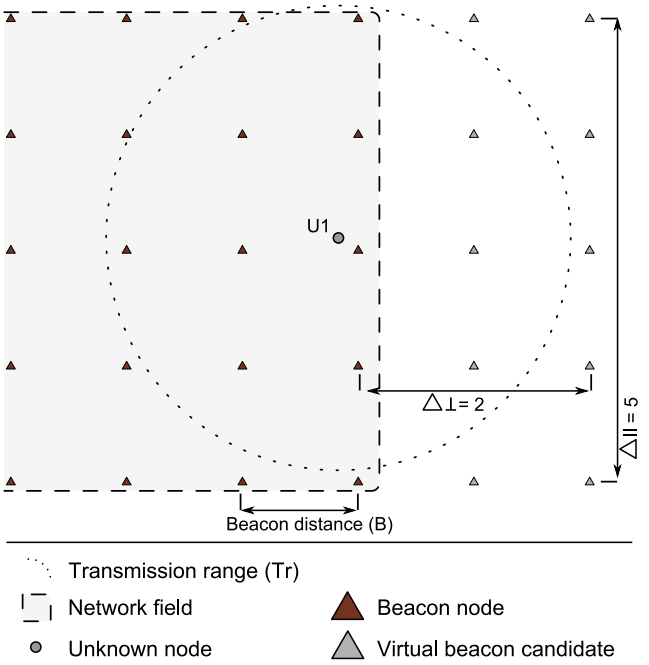

Fig. 5. Virtual Beacons, generated at the border in a first step.

this distance is less than the transmission range, the virtual beacon will be used. Otherwise it will be dropped. In fact the distance towards a virtual beacon cannot be figured out exactly, because the nodes position is unknown. At this point an approximation has to be applied, which approximates the position as centroid of the nearest three beacon nodes. There is a closer look on how to find the nearest beacon nodes in the next subsection. Using this assumption, the generated nodes can be classified as required or can be dropped, respectively. After this successful virtual beacon generation process the node calculates its position as normal for the chosen algorithm.

B. Drop Beacon Strategy

On the one hand, our second approach can be seen as the opposite of the first one. On the other hand it has a lot in common with it. The first algorithm uses a subset of beacons to decide which of the generated beacons should be used and which should be dropped. The idea of this algorithm is a simplification of the first one. It tries to rebuild the balance of beacon nodes around the current node by dropping existing beacons.

The first step to localize a node is to detect if it resides at the border. This is done with the help of the described border detection method (Section IV). As a border node, it now has to find out the four nearest beacons. Only these will be used to calculate its position. One approach is to use RSSI or similar approximations like LQI or Time of Arrival measurements (ToA) to find the nearest four. Either RSSI or LQI is available on most platforms. Furthermore all weighted CL based algorithms build on such a distance approximation. If a feasible distance approximation is not available, only CL can be used and the nearest four have to be found otherwise.

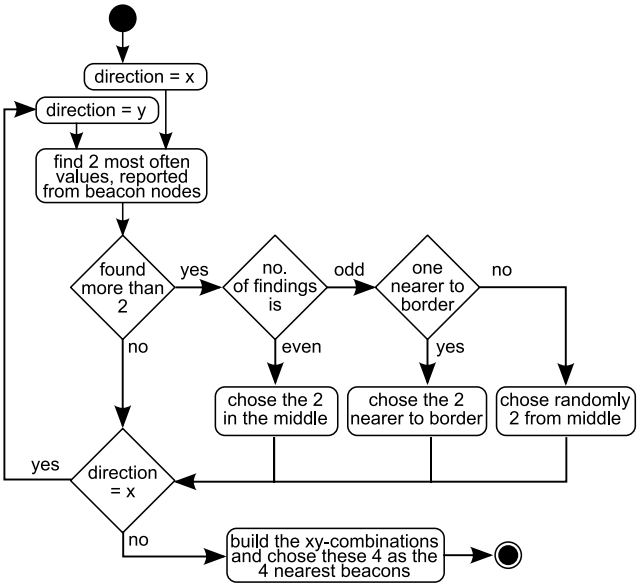

Fig. 6 Proposal to find the nearest four beacon nodes without distance information.

Therefore we propose the procedure, described in Fig. 6 to choose the nearest four beacon nodes. In our simulations we assumed to get the nearest 4 beacons via distance information, e.g. RSSI or LQI information.

VI. Analyses and Simulations

To analyze the presented strategies we used a Matlab based simulation environment. Within this environment, an area of 40 x 40 units of length, e.g. meter, is defined. Beacon nodes are placed at every tenth unit in both directions, which means 5 x 5 beacon nodes are distributed within this field. Normal sensor nodes, i.e. nodes to be located, have been placed on a regular grid with one unit spacing between grid points, i.e. 1600 sensor nodes or one sensor node at 1600 positions, respectively. Localization within this environment has been performed using CL, WCL and MWCL, without any kind of correction and with the described improvements. In addition, the transmission range which is assumed to be the same on beacon nodes and normal nodes, has been varied from 8 units of length to 56 units of length.

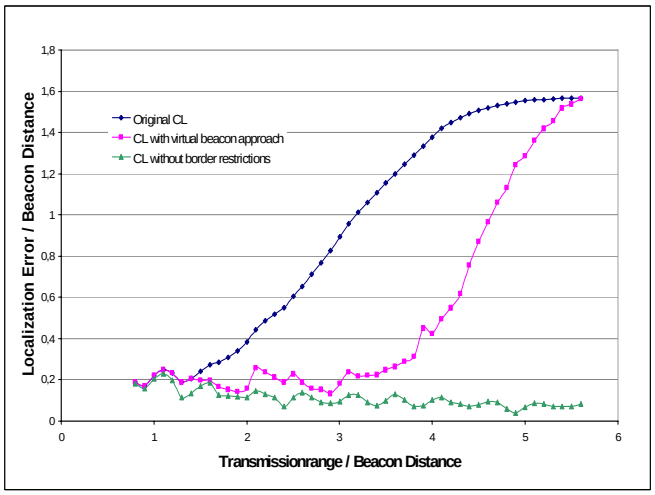

Fig. 7. Mean localization error versus transmission range. Lowest error is achieved, if no border affects the localization, i.e. unlimited beacons in each direction. Otherwise virtual beacon approach can minimize the mean error compared to CL.

A. Beacon virtualization

We first analyzed the strategy of beacon virtualization. As a representative for the studied localization algorithms, Fig. 7 shows various types of CL. The mean localization error over all node positions is plotted versus the transmission range. The worst graph belongs to the original CL algorithm, which is affected by border effects. The best graph, illustrated at the bottom, is also original CL but without border effect, i.e. enough beacons have been provided with global knowledge. The third graph shows CL with additional virtual beacons as described above. The mean error is all the time less or at least equal to the original CL. This indicates a reduction of the localization error caused by border effects.

To ensure that the achieved improvement in localization, which is shown by the mean error, really belongs to an improvement within the border area, we also plotted the distribution of the localization error. In Fig. 8 two error distribution graphs are illustrated concerning CL. On the left side the nodes in the area were localized via the original algorithm. On the right side, the same localization algorithm

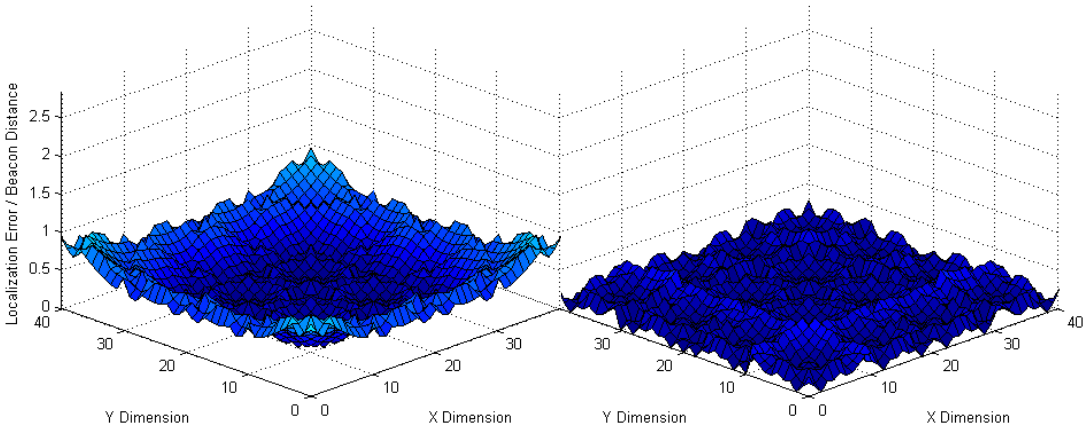

Fig. 8. Localization error distribution without virtual beacons (left) and with virtual beacons (right). Used localization algorithm: CL

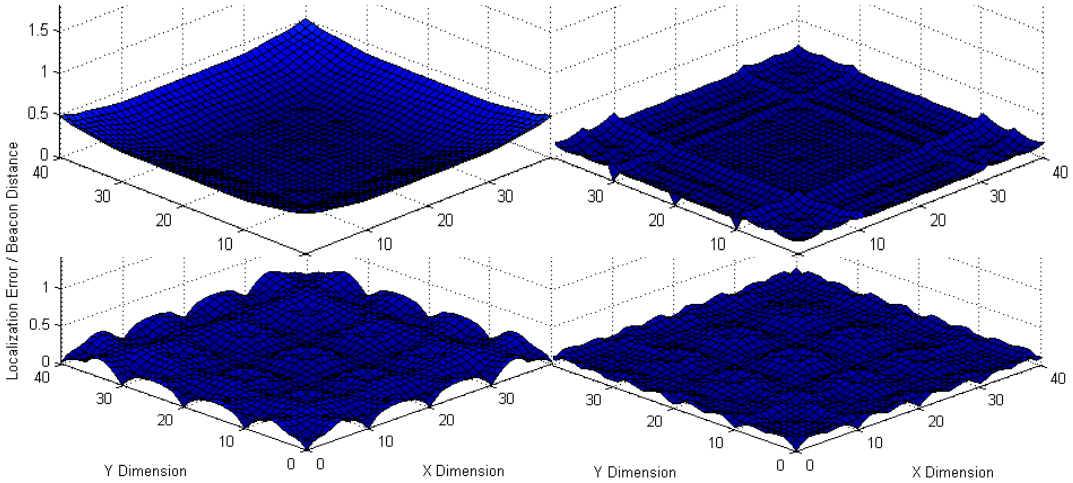

Fig. 9. Localization error distribution without virtual beacons (left) and with virtual beacons (right). Localization algorithms: MWCL (top) and WCL (bottom).

with applied virtual beacon approach was used to localize nodes in the area. The error distributions clearly show that the average reduction of localization error depicted by Fig. 7 is the result of improved localization accuracy near the borders.

It is shown in Fig. 9 that the virtual beacon approach is also able to improve the localization accuracy of WCL and MWCL.

B. Beacon Dropping

We also investigated our beacon dropping approach in the same way as our beacon virtualization. As expected, the localization accuracy can be increased by applying this approach. In Fig. 10, CL with beacon dropping is compared to normal CL. Similar to beacon virtualization, this approach also yields improved localization accuracy.

C. Comparison

To compare both strategies, we determined the mean localization error for the localization approaches CL, WCL and MWCL. All three algorithms have been used 1) without border correction, 2) with our beacon virtualization approach and 3) with our beacon dropping approach. The resulting localization error of CL is plotted against the transmission range in Fig. 11. The graph shows that our presented approaches only slightly differ from each other. In this case beacon virtualization performs slightly better than beacon dropping.

We obtained similar results for WCL and MWCL. To keep the illustration as simple as possible, only the results for WCL and MWCL are illustrated in Fig. 12. For both algorithms the mean error is illustrated versus the transmission range using beacon virtualization as well as beacon dropping.

The illustrated diagrams show that both of our proposed approaches have a positive impact to localization in border areas. Both approaches provide similar results. In comparison to our virtual beacon approach, beacon dropping produces equivalent localization accuracy with significant smaller demands. In particular, beacon dropping, firstly, does not need to generate additional beacons and, secondly, it works with simpler border detection and nodes do not need to determine which edge they belong to. From this point of view the cost-performance ratio of beacon dropping is better than this of virtual beacon.

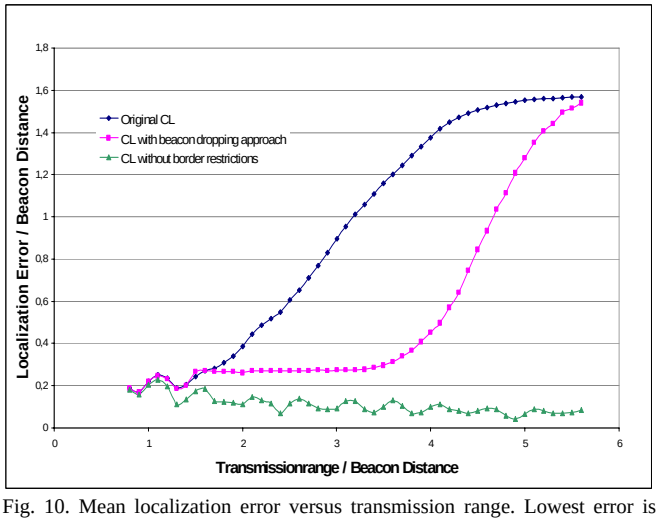

Fig. 10. Mean localization error versus transmission range. Lowest error is achieved, if no border affects the localization, i.e. unlimited beacons in each direction. Otherwise beacon dropping approach minimizes the mean error compared to CL.

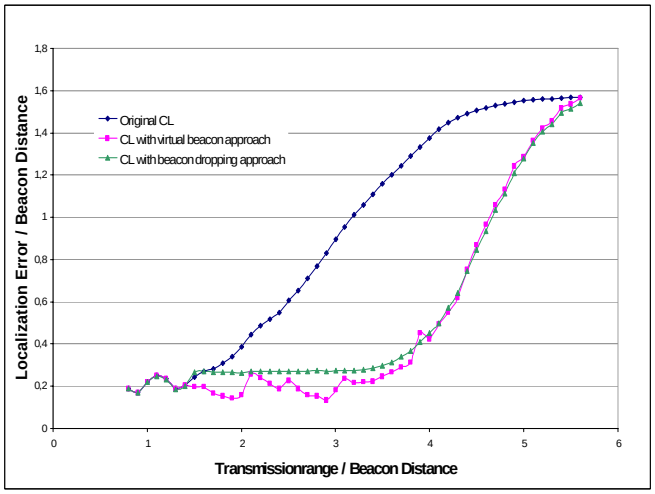

Fig. 11. Mean localization error versus transmission range. Virtual beacon approach and beacon dropping approach with similar performance.

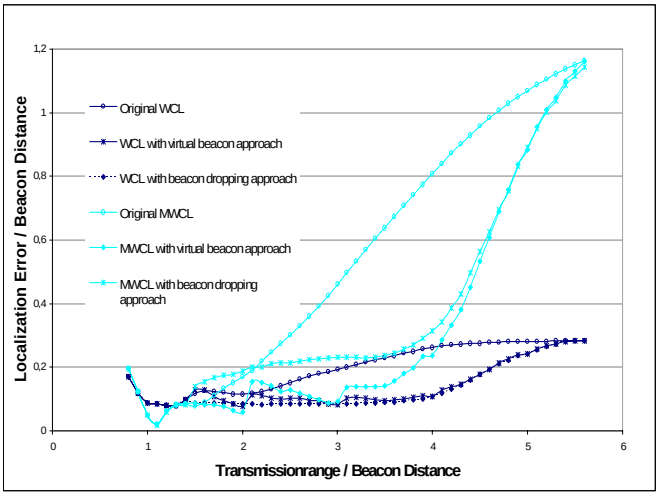

Fig. 12. Mean localization error versus transmission range. Virtual beacon approach and beacon dropping approach performed with WCL and MWCL as examples of weighted localization approaches.

D. *Evaluation*

Our presented approaches can improve localization for all CL based localization algorithms. The actual gain depends on the used localization algorithm. As illustrated in Fig. 11, CL is significantly improved by beacon virtualization. In contrast, WCL becomes only slightly improved. This is due to the fact that WCL is more accurate than CL. Therefore the potential for improvements is smaller. Additionally, WCL itself inherently carries out some form of beacon dropping strategy, as the beacon's influence is the smaller the higher its distance towards the node is. The impact of this behavior can be controlled with the degree *g*.

Nevertheless for high transmission ranges both methods approach the same mean localization error as the original algorithm. This is due to the utilization of the border detection algorithm. If the transmission range is too high, this approach cannot determine a single edge or corner to which the node belongs to. Consequently, the node is classified as situated in the network's center and neither the beacon virtualization approach nor the beacon dropping approach is able to correct the localization. A border detection scheme which only relies on the beacon density would be able to detect the border even in this case. Therefore, the beacon dropping approach would be able to improve localization also in this case.

VII. Conclusion and Future Work

We presented two strategies to handle the increased localization error near the border of a sensor network, using CL-based algorithms for localization. We firstly discussed the reasons for this decrease in accuracy. Thereafter, we outlined a border detection algorithm used by our error reduction strategies. The first strategy uses virtual beacons to rebuild a balanced beacon distribution for border nodes. These virtual beacons compensate for the unequal distribution of beacon nodes around a node near the network border. The second strategy strives for balanced beacon distribution by ignoring those beacons in range which are pulling the nodes position towards the network core.

We showed that both strategies effectively reduce the increased localization error near the border and therefore improve the mean localization accuracy of the whole network. Both strategies perform nearly equal. This is caused by approximations used for the virtual beacon strategy. We have also illustrated, that not all weighted localization algorithms benefit equally from the proposed strategies. This is due to the fact that these algorithms, namely WCL, apply a smaller weight to distant beacons. This can be regarded as another kind of border error correction. Therefore, the achievable improvement depends on the quantification of the localization algorithm.

The presented algorithms build on information about adjacent beacons as well as on an assumed beacon density. Further work should use either one of this basic information or any combination of both to directly influence the localization process. We found out that the border detection process impacts our error correction strategies. Therefore improved border detection or even a border detection free algorithm would be of great interest. Additionally, we intend to evaluate our approach using real world deployments.

References

[1] Kemal Akkaya and Mohamed F. Younis. A survey on routing protocols for wireless sensor networks. *Ad Hoc Networks*, 3(3):325–349, 2005.

[2] I. F. Akyildiz, Su Weilian, Y. Sankarasubramaniam, and E. E. Cayirci. A survey on sensor networks. *IEEE Communications Magazine*, 40(8):102–114, 2002.

[3] J. N. Al-Karaki and A. E. Kamal. Routing techniques in wireless sensor networks: a survey. *IEEE Wireless Communications*, 11(6):6–28, 2004.

[4] R. Behnke and D. Timmermann. AWCL: Adaptive Weighted Centroid Localization as an Efficient Improvement of Coarse Grained Localization. *Positioning, Navigation and Communication, 2008. WPNC 2008. 5th Workshop on*, pages 243–250, March 2008.

[5] Jan Blumenthal. *Ressourcenarme und dezentrale Lokalisierung autonomer Sensorknoten in Sensornetzwerken*. PhD thesis, University of Rostock, Germany, 2008.

[6] Jan Blumenthal, Ralf Grossmann, Frank Golatowski, and Dirk Timmermann. Weighted Centroid Localization in Zigbee-based Sensor Networks. In *IEEE International Symposium on Intelligent Signal Processing, WISP 2007*, Madrid, October 2007.

[7] Nirupama Bulusu, John Heidemann, and Deborah Estrin. GPS-less low cost outdoor localization for very small devices. *IEEE Personal Communications Magazine*, 7(5):28–34, October 2000.

[8] Jerry D. Gibson. *The Mobile Communications Handbook*. CRC Press, Boca Raton FL, USA, 1996.

[9] Jakob Salzmann, Ralf Behnke, Dominik Lieckfeldt, and Dirk Timmermann. 2-Mascle - A Coverage Aware Clustering Algorithm with Self Healing Abilities. In *3th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, ISSNIP2007*, December 2007.

[10] Stephan Schuhmann, Klaus Herrmann, Kurt Rothermel, Jan Blumenthal, and Dirk Timmermann. Improved weighted centroid localization in smart ubiquitous environments. *Ubiquitous Intelligence and Computing*, pages 20–34, 2008.