Using QR-Updating with Reduced Complexity for Precise Localization in Mobile Sensor Networks

Frank Reichenbach, Dominik Lieckfeldt, Dirk Timmermann
Institute of Applied Microelectronics and Computer Engineering
University of Rostock, Warnemuende, Germany
{frank.reichenbach,dominik.lieckfeldt,dirk.timmermann}@uni-rostock.de

Abstract

Localizing tiny sensor nodes in large wireless sensor networks is extremely complex, due to the node's strict resource limitations regarding memory size, processor performance and overall battery capacity. Moreover, in case of mobile nodes the developing process for localization methods is even more challenging.

This paper presents an efficient approach for localization in mobile sensor networks, which is based on an extension of the already known "Distributed Least Squares (DLS)"-algorithm. The extension consist of a mobile phase, where sensor nodes update their initial position by QR-factorization. We show in extensive simulations that the mobile DLS-algorithm leaves only minimal computations at the resource constrained sensor nodes.

1 Introduction

Most localization algorithms in the literature focus on static sensor networks. However, to model behavior of realistic networks, mobility of sensor nodes has to be taken into account. Consequently, this paper addresses the extension of recent works in this field in order to also consider mobile sensor networks. Mobility contributes significantly to the overall behavior, since (i) the phenomenon to be observed can change its location, (ii) nodes themselves can move either due to environmental dynamics (e.g. wind) or (iii) because they are attached to objects and persons of interest, respectively. As a consequence, mobility in wireless sensor networks poses a considerably greater challenge on the algorithms used:

- Positions must be updated frequently.
- Beacon positions, distances or angles must be determined simultaneously.
- Synchronization of medium access becomes more difficult, since beacons share the same communication medium.
- Additionally, finding the optimal path for moving beacons in terms of sensor coverage is an important aspect, which can reduce the the number of unlocalized nodes[1] [10].

Despite all challenges, mobility also demands for new approaches to improve the accuracy of localization:

- Additional parameters, like velocity, acceleration, direction of movement and older positions can be used to restrict the solution space.
- Mobile beacons can be ordered to move towards unlocalized nodes, which allows these nodes to perform localization as well.
- Few mobile beacons, following smart movement patterns replace many static beacons.
- The density of nodes can be increased in the vicinity of phenomenons and decreased in uninteresting regions.

On one hand, mobility leads to more complex algorithms as expected. On the other hand, suitable algorithms, which exploit mobility have been reported to improve network coverage, enable localization of isolated nodes and, therefore, increase overall accuracy of localization. A trivial approach to solve the mobile localization problem would be to completely repeat the static localization process. However, this would lead to suboptimal results and an unnecessary high energy consumption.

This paper is structured as follows. Section 2 provides information about related work. A brief introduction to the "Distributed Least Squares"-algorithm is given in Section 3. Next, Section 4 describes the new mobile DLS-algorithm

[1] These are nodes with no beacons in receiving range and thus unable to localize themselves.

and presents simulation results showing the performance of the proposed algorithm regarding localization error and energy consumption in Section 5. Finally, Section 6 concludes the work.

2 Related Work

Bayesian filtering is a common approach to deal with the challenges of a dynamic system. The Bayesian approach provides means to estimate and predict the most likely current or next state of a dynamic system. This prediction is based on the knowledge of its previous states. Applied to the localization problem, Bayesian filters can answer the following question: What is the likelihood that a specific sensor is located at $P_{est}(x, y)$ if previous observations of the mobile sensor network are known?

There exist some extensions of the general concept of Bayesian filtering, which are especially suited for sequential estimation/prediction. One of these extensions is the Kalman filter, which is used for linear systems and Gaussian noise. In order to overcome the limitations of the original Kalman filter, several improvements have been published. One of the most interesting ones are the Particle filter and the sequential Monte Carlo Method [11, 8], which can be used with non-linear systems and non-Gaussian noise. Particle filters can be applied to a wide range of applications. Their basic working principle is to consider a cloud of particles, where each particle represents a point in the state space. In addition, each particle constitutes an assumed probability density distribution of the phenomenon of interest and, therefore, a specific hypothesis of the real world. In the next step, each particle is assigned a weight according to measurements. Since each particle represents an independent hypothesis and the accuracy of this approach depends on the number of particles used, Particle filters are relatively resource intensive.

More interesting is the case if mobility patterns of nodes, characterized by velocity, acceleration, and direction, are known as well. One corresponding example is the Joint Audio and Visual Localization of moving objects [21]. Cars moving in a GSM (Global System for Mobile Communications) cell consitute another example. The "Constant Velocity Model", "Nearly Acceleration Model" or the "Singer Model" can be used to approximate the movement of cars. For more details, we suggest [6].

In [9] the authors report a Monte Carlo-based localization scheme, which determines the location in regular time intervals. After choosing random locations, which are within the network's physical bounds, a "Prediction" and a "Filtering" step follow. In the prediction-step random locations are chosen, which are now based on the previous locations and the velocity of the target. The data set of locations is recalculated regularly. In the filtering-step, which is supported by distance measurements, all locations being unable to reach are eliminated. An optimization of this algorithm is discussed in [2]. Here, a more accurate data set is created in the prediction-step, which contains less impossible locations.

3 Background: Localization with the "Distributed Least Squares (DLS)"-Algorithm

This section describes the DLS-algorithm. Here, we assume m as the number of beacons and s the number of sensor nodes. Moreover, the localization error f_i is defined as distance between exact position $P_{exact}(x, y)$ and estimated position $P_{est}(x, y)$ of sensor node i:

$$f_i = \sqrt{(x_{exact} - x_{est})^2 + (y_{exact} - y_{est})^2} \tag{1}$$

3.1 Building a Model

Estimating $P_{est}(x, y)$ requires at least three known points (2D). Given m known coordinates $B(x_i, y_i)$ and the corresponding distances r_i between P and each $B(x_i, y_i)$ we obtain:

$$(x - x_i)^2 + (y - y_i)^2 = r_i^2 \quad (i = 1, 2, \ldots, m) \tag{2}$$

This system of equations must be linearized with a term extension [14]. For that, we use the first equation of (2). By adding and subtracting x_1 and y_1 to all other equations this leads to:

$$\begin{array}{l}(x - x_1 + x_1 - x_i)^2 + (y - y_1 + y_1 - y_i)^2 = r_i^2 \\ (i = 2, \ldots, m)\end{array} \tag{3}$$

Given the distance r_1 (r_i), which is the distance between the unknown point and the first (i'th) beacon, and the distance d_{1j}, which is the distance between B_1 and B_j, this leads, after resolving and simplifying, to:

$$\begin{array}{l}(x - x_1)(x_i - x_1) + (y - y_1)(y_i - y_1) \\ = \frac{1}{2}\left[r_1^2 - r_i^2 + d_{1j}^2\right] = b_{1j}\end{array} \tag{4}$$

This consitutes a linear system of $m - 1$ equations and $n = 2$ unknowns.

$$\begin{array}{rcl}(x - x_1)(x_2 - x_1) + (y - y_1)(y_2 - y_1) & = & b_{12} \\ (x - x_1)(x_3 - x_1) + (y - y_1)(y_3 - y_1) & = & b_{13} \\ & \vdots & \\ (x - x_1)(x_m - x_1) + (y - y_1)(y_m - y_1) & = & b_{1m}\end{array} \tag{5}$$

This system of equations can be written in matrix form $A\mathbf{x} = \mathbf{b}$:

$$A = \begin{pmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \\ \vdots & \vdots \\ x_m - x_1 & y_m - y_1 \end{pmatrix},$$

$$\mathbf{x} = \begin{pmatrix} x - x_1 \\ y - y_1 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} \frac{1}{2}\left[r_1^2 - r_2^2 + d_{12}^2\right] \\ \frac{1}{2}\left[r_1^2 - r_3^2 + d_{13}^2\right] \\ \vdots \\ \frac{1}{2}\left[r_1^2 - r_m^2 + d_{1m}^2\right] \end{pmatrix} \quad (6)$$

This basic form has to be solved using the linear least squares method. Due to the fact that overdetermined systems of equations with $m >> n$ may have no exact solution for $A\mathbf{x} = \mathbf{b}$, we apply the Euclidean norm and minimize the sum of the squares of the residuals:

$$\underset{x \in \Re^n}{Minimize} \quad ||A\mathbf{x} - \mathbf{b}||_2. \quad (7)$$

To summarize, linear systems of equations can be solved iteratively using *splitting techniques*, directly with *normal equations* or *orthogonal factorization* [12].

Solving normal equations is a good choice if the linear system has many more equations than unknowns, i.e. $m >> n$, because after multiplying $A^T A$ the result is only a quadratic $[n \times n]$-matrix. However, numerical difficulties may lead to completely wrong positions. For that reason, orthogonal techniques should be used instead. Moreover, the update mechanisms, developed for orthogonal techniques, can be efficiently used for localization in mobile sensor networks.

3.2 Solving the Linear Least Squares Problem

Many orthogonal techniques exist in the literature e.g. Cholesky-decomposition, Schur-decomposition, QR-factorization (QRF) or SV-decomposition.

Due to its extensive documentation and library support, we focus on QRF for solving the linear least squares problem [20]. Furthermore, there are several implementations of QRF available that are numerously stable and which can be easily used for custom developments.

QRF is based on any $m \times n$ matrix A with $m > n$. For such a matrix, there exists an orthogonal $m \times m$ matrix Q and an upper triangular matrix R_1.

$$A = Q \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \quad (8)$$

With QRF the problem $A\mathbf{x} \approx \mathbf{b}$ can be transformed into the triangular formed system:

$$\|A\mathbf{x} - \mathbf{b}\|_2 = ||Q \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \mathbf{x} - \mathbf{b}||_2 = || \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \mathbf{x} - Q^T\mathbf{b}||_2. \quad (9)$$

Reconverting leads to:

$$\begin{pmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{pmatrix} = Q^T\mathbf{b} = \begin{pmatrix} Q_1^T\mathbf{b} \\ Q_2^T\mathbf{b} \end{pmatrix}. \quad (10)$$

This system can be solved with the following steps:

1. calculate the reduced QRF of A with $A = Q_1 R_1$
2. determine $\mathbf{z} = Q_1^T\mathbf{b}$
3. get the result by backward substitution of $R_1\mathbf{x} = \mathbf{z}$

Hereby, determining Q_1 and R_1 dominates the computational overhead. In our implementation, we realized QRF with Householder matrices and Givens rotations.

3.3 Distributing the Complex Task

DLS is based on the mathematical formulations introduced in the background section. The key for distributing the workload for localization is to exploit the properties of the matrices in equ. (6) which is explained in the following. First, all elements in the coefficient matrix A are generated by beacon positions $B_1(x, y) \dots B_m(x, y)$ only. We assume that all sensor nodes can establish communication links between all beacons in the first instance. In this case, A is identical on every sensor node. Second, the vector $\mathbf{b}$ contains distances between sensor nodes and beacons $r_1 \dots r_m$, which have to be estimated on every sensor node independently. Consequently, the QRF can be split into two parts - a more complex part, the *precalculation*: calculation of Q_1 and R_1 and a simple part: calculation of $R_1\mathbf{x} = Q_1^T\mathbf{b}$, $x = \mathbf{z} + (x_1 \ y_1)^T$, in the following referred to as *postcalculation*. Here, the *precalculation* is executed on one high performance node. Assuming the existence of such a node also avoids high redundancy, because otherwise this *precalculation* has to be executed on all sensor nodes separately. It is very important to emphasize that the *precalculation* is identical on each sensor node. Thus, *precalculation* is conducted only once, which results in significant savings regarding battery capacity. The simple *postcalculation* is then executed on each sensor node with its individual distance measurements to all beacons. This approach complies with two important design strategies for algorithms in large sensor networks - a **resource-aware** and **distributed** localization procedure. Finally, the algorithm requires less communication overhead compared with other exact algorithms.

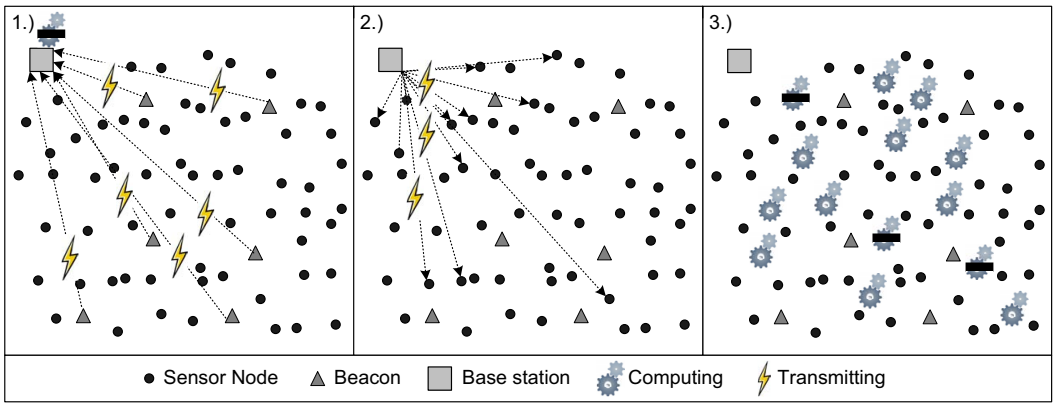

Figure 1. Procedure of the DLS-algorithm, which is divided into three phases.

3.4 The Algorithm in Brief

At this point we briefly describe the algorithm. DLS is divided into three phases, which are shown in Fig. 1. In phase 1, all beacons send their position $B(x, y)$ hop by hop over their adjacent beacons to the base station[2]. In phase 2, the base station starts generating the initial matrices and computes Q_1 and R_1. The result is fowarded by the beacons to all sensor nodes. Sensor nodes measure the distances to all beacons, execute the *postcalculation* and finally estimate their location.

In [18, 16] DLS was analyzed in detail. In the following, we will briefly review the main benefits of DLS:

Communication: Every sensor node must receive the precalculated matrix $Q_1^T, R_1, \mathbf{d}_p$ and x_1, y_1 with $[2 \cdot (b-1)] + 2 + (b-1) + 2$ elements. This results in receiving 303 elements or 1212 bytes if 100 beacons are used and floating point representation of every element is assumed.[3]

Computation: The *postcalculation* on sensor nodes requires $12b - 9$ flops, which is marginal compared to the complete calculation with $4b^3 - 5b^2 + 13b - \frac{5}{2}$ flops. If 100 beacons are included in the calculation, then the complete calculation would take ca. 3.9 Mflops and the *postcalculation* 1.2 kflops, which is less than 97%.

Memory: At best, space in memory has to be reserved for only some temporary variables and two accumulators, which reduces memory consumption to a minimum. Following, we will describe the modifications for a mobile DLS.

[2] A base station can be a laptop or a high powered node.

[3] On common microcontrollers that are presently integrated on sensor node platforms, every element is stored in floating point representation as a 4 byte number.

4 Adapting the DLS-Algorithm for Mobile Networks

Regardless which nodes are mobile, the DLS-algorithm must be processed once in a so called initialization phase. After this phase, every sensor node has determined its initial position. After initialization all sensor nodes must relocate themselves within a specific time interval, which depends on the application. It first comes in mind to continuously repeat DLS again and again. But that makes only sense if beacons are highly mobile and the sensor nodes are non-mobile, which is described in Section 5.2. For all other types of mobility a complete repetition would be a waste of resources. This observation motivated us to find efficient update mechanisms that work only upon the available informations. These updates extend DLS to be applicable in mobile scenarios. We refer to the mobile version of DLS as mDLS. Next, we will introduce the different mechansims to update the matrices Q and R.

4.1 Adding a Row: "QR-Updating"

QRF provides an updating of Q and R instead of calculating them again when coefficients in A changed. If A is expanded by a new row $\mathbf{w}^T$, which is equal to get new beacon positions, this leads to:

$$\breve{A} = \begin{pmatrix} \mathbf{w}^T \\ A \end{pmatrix}. \tag{11}$$

The updated matrices $\breve{Q}$ and $\breve{R}$ can be determined based on the updated coefficient matrix $\breve{A} = \breve{Q}\breve{R}$. With Givens rotations the solution can be found.

4.2 Eliminating a Row: "QR-Downdating"

Rows $\mathbf{w}^T$ in A can be eliminated to get the updated $\breve{A}$ with the following approach in [15]. Find the QRF of $\breve{A}$ with Givens rotations:

$$G_1^T \cdots G_{m-1}^T q = \gamma e_1 \tag{12}$$

, where q^T is the first row of A and $\gamma = \pm 1$. By using an upper Hessenberg matrix, an orthogonal matrix Q can be determined. Details on how to calculate $\breve{A} = QR_1$ are presented in [15]. Eliminating any desired row in A is analogue. If Q is not available, then a possible solution can be found in [5].

4.3 Changing a Row: "QR-Upgrading"

This can be accomplished with a complexity of $\mathcal{O}(n^2)$, instead of repeating a complete QRF with the complexity of $\mathcal{O}(n^3)$ [7]. More details can be found in [4, 3, 13].

4.4 mDLS: Algorithm Description

4.4.1 Basic approach

The classical DLS is not appropriate for networks with completely mobile nodes, because both distances and beacon positions change frequently. Therefore, we improved DLS to a mobile version called mDLS based on the updating methods for QRF, which is explained in detail in the following.

Initially, DLS is processed, so that a first position and all matrices are stored on each sensor node. Then in a next mobile phase, sensor nodes wait for new beacon packets including beacon positions. If necessary, the matrices Q and R_1 can be updated on every sensor node. Communication overhead is minimal, because beacons send only few packets. All resource limited sensor nodes can be left in receive mode rather than transmit mode to save energy.

Two possibilities can initiate the relocalization on sensor nodes in the mobile phase. Either beacons send new packets after they located a new position with their positioning system (*GPS check period*) or sensor nodes localize themselves autonomously in every *Relocate period*. This leads to the following situations:

1. A sensor node receives a beacon packet:
   (a) The sensor node already stored an older position of this beacon $\rightarrow Q$ and R_1 must be updated by a QR-Upgrade (see Section 4.3).
   (b) The beacon is unknown. $\rightarrow Q$ and R_1 must be added by a QR-Update (see Section 4.1).
2. Every sensor node measures distances to all known beacons within the relocate period.
   (a) A new distance measurement is missing due to the absence of a beacon $\rightarrow Q$ and R_1 must be adapted by a QR-Downdating (see Section 4.2).
   (b) A new beacon packet was not received, but distances changed $\rightarrow Q$ and R_1 can be kept as they are.

Every of the before mentioned cases is followed by a *postcalculation* with the new position as result.

4.4.2 Analysis

Eliminating a beacon position (QR-Downdating) and thus deleting a row in Q must be handled very carefully, because QR-Downdating only works with Q and not with its reduced form Q_1. In detail, 50 beacons already require 361 elements or 1.4 kB, respectively, for calculations because the dimension of Q is $(b-1) \times (b-1)$. Nevertheless, mDLS can be very efficient for these cases:

- Sensor nodes are equipped with enough memory.
- Sensor nodes are highly mobile and the relocate period is very small.
- A high number of distances can temporarily not be determined.

By using mDLS sensor nodes change higher memory requirements against high computation as well as communication cost. This theoretically leads to a lower energy consumption.

4.4.3 Extensions

For some applications it is more feasible to accept increased communication effort and, instead, reduce the memory needed while keeping the advantages of mDLS. We assume a clustered network structure with cluster beacons that conduct the *precalculation* of the base station. Cluster beacons permanently store the latest matrices Q and R by communicating with each other and provide them to their corresponding cluster. This can be achieved by broadcasting only the reduced matrices Q_1 and R_1 in a specific period of time to all nodes in the individual cluster. If a node desires to update its position, it (i) receives the latest matrices, (ii) measures the distances to all available Cluster beacons , and (iii) conducts the *postcalculation* to finally determine its location. Although nodes additionally consume energy for receiving the matrices, the overall computational overhead and memory usage for localization is reduced. The simulation results presented in Section 5.1 prove these statements.

5 Finding the Best Solution for Every Type of Mobility

At this point, we will suggest an adequate localization process for every type of node mobility.

5.1 Mobile Nodes and Static Beacons

As we already said, we assume that every sensor node has an initial estimate of its location. The static nature of the beacons leads to a highly resource aware localization procedure by simply repeating the *postcalculation* (phase 3) of DLS. This is possible because the orthogonal matrices Q and R_1 will always be the same. Additionally, every sensor node must measure the distances to all beacons, which are the elements of **b**. To summarize, the *postcalculation* completely avoids communication overhead and needs the before mentioned complexity which is only $8m - 11$ flops.

The simulations presented in the following were conducted with the network simulator "J-Sim" from Ohio State University [1] with some modified and also some new components. Results are based on a custom energy model, which considers energy dependent transmission ranges and realistic energy consumption for different computation complexities. As reference platform for specific energy parameters, we used MICA2. Implementation details can be found in [19, 17, 16]. The simulations provide information about the behavior of localization error and energy consumption. All nodes are randomly deployed according to a uniform distribution. The base station is placed at position $P(1\mathrm{m}; 1\mathrm{m})$. To simulate mobility, nodes move randomly within a *moving period* to a new position in the sensor field. Moreover, the localization process is repeated within a *relocate period*. In mobile fields there is a high probability for unconnected nodes. Thus, we define two fields, where beacons are deployed over $150\ \mathrm{m} \times 150\ \mathrm{m}$ and sensor nodes over a smaller field with $100\ \mathrm{m} \times 100\ \mathrm{m}$. Since it is possible for a beacon to be deployed outside the smaller field, unconnected nodes are simulated.

Tab. 1 lists the simulation parameter. To guarantee that every new position can be detected reliably, the relocate period is always one half of the moving period.

Fig. 2 and 3 show the results of the simulation. Details of the first 3 phases of DLS can be found in [19, 17, 16]. In the following, we will focus on the fourth phase, where the algorithm regards mobility.

At $109\ \mathrm{s}$, sensor nodes start with the periodical relocation process. In Fig. 2, the dominant part of energy consumption is related to the measurement of the received signal strength indicator, which requires an active transceiver in receiving mode. We estimated with this configuration an overall runtime of the sensor network for 572 days until the batteries would be exhausted. This estimated network lifetime is achieved at a mean relative localization[4] error of 3.1% (compare fig. 3) at maximal 3.5% and minimal 2.7%.

Parameter	Value
Field dimensions [m]	100×100
Number of sensor nodes	300
Number of beacons	25
Transmission range of all beacons [m]	45
Variance of the noisy distances [m]	10
Moving period [s]	10
Relocate period [s]	5
Simulation time [s]	300

Table 1. Parameter of a simulation with mobile sensor nodes and static beacons.

5.2 Mobile Nodes and Mobile Beacons

For that case, we developed the mDLS-algorithm. Parameters are similar to those of the simulation described before. Additionally, the *GPS check period* defines the relocation time of the beacons. Tab. 2 lists all simulation parameters.

Fig. 4 and 5 illustrate the results. The first three phases of DLS are completed at ca. $109\ \mathrm{s}$. After that the sensor nodes start to move. From ca. $109\ \mathrm{s}$ the beacons move as well. Until $160\ \mathrm{s}$ all beacons move within the regular sensor field. From $160\ \mathrm{s}$ on, the localization error increases, because the first beacons move out of the regular sensor field and thus lose connection to sensor nodes. The minimal number of beacons temporarily reaches only three. For the rest of the simulation, 24 beacons are never reached again. Two outliers in fig. 4 are remarkable, which are due to both defective locations and the situation that only few beacons are reachable. In the relocalization phase mDLS achieved a mean error of $F = 2.4\%$ (at $f_{max} = 3.5\%$ and $f_{min} = 2.1\%$).

Fig. 5 shows the relatively high energy consumption that occurred between $124\ \mathrm{s}$ and $160\ \mathrm{s}$, because beacons de-

[4] The mean localization error is an average over all sensor nodes errors.

Parameter	Wert
Field dimensions [m]	100×100
Number of sensor nodes	300
Number of beacons	25
Transmission range of all sensor nodes [m]	15
Transmission range of all beacons [m]	75
Variance of the noisy distances [m]	10
Moving period [s]	10
GPS check period [s]	10
Relocate period [s]	10
Simulation time [s]	300

Table 2. Parameter of a simulation with both mobile sensor nodes and mobile beacons.

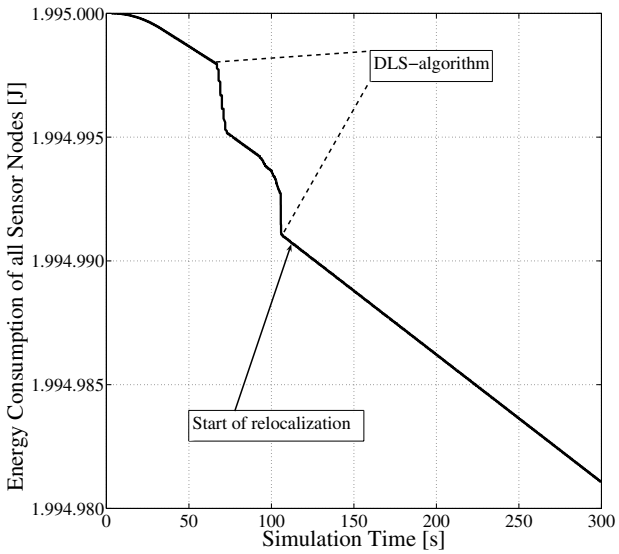

Figure 2. Accumulated energy capacities of all sensor nodes over simulation time.

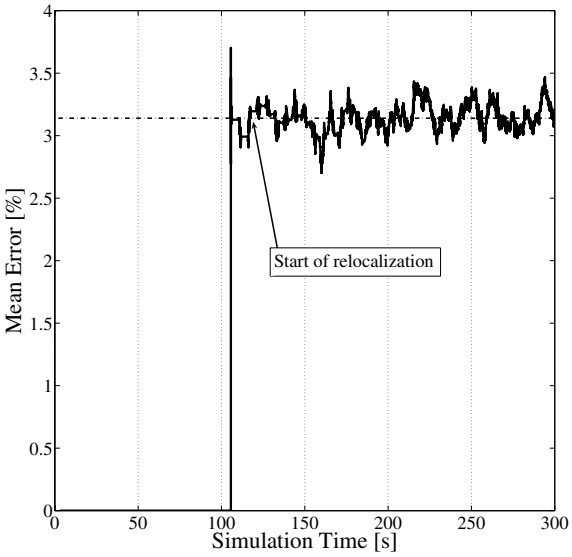

Figure 3. Relative mean localization error over simulation time.

tected new positions very often and thus sent many position packets. This leaded to a higher overhead on the sensor nodes, because of more received data and more *postcalculations*. Starting at 160 s beacons move out of the field that decreases the number of sent position packets and the energy consumption. Until simulation end over 66000 position packets had been received.

Finally it must be mentioned that Reichenbach proved in [16] the localization error's stability over time, which may not increase.

5.3 Static Nodes and Mobile Beacons

This type of mobility is similar to the before discussed solution, because both the distances and the beacons positions change permanently. If beacon mobility is very high, then DLS must be completely repeated in every relocation process. Otherwise, if beacon mobility is low, mDLS can be used again. This leads to the following cases when a new position packet is received on a sensor node:

1. The sensor node already know an older position of the beacon $\rightarrow$ Matrices Q and R are refreshed by an QR-Upgrade (see Section 4.3).

2. The beacon is unknown. $\rightarrow$ The new position is added by a QR-Update (see Section 4.1) to matrices Q and R.

By determining new distances and executing a *postcalculation*, the latest positions can be determined.

6 Conclusions

We presented a new approach to achieve resource aware localization of sensor nodes in mobile networks. For that, the classical "Distributed Least Squares"-algorithm (DLS), which demands only a minimum of memory, communication, and computation on every node was extended to a mobile version. This mobile DLS (mDLS) uses the QR-factorization to solve the least squares problem of calculating an initial position of every sensor node. If the sensor node itself or one of the beacons move, its latest position is calculated with minimal communication and computation overhead resting upon update operations. This avoids to completely repeat the first calculation. Extensive simulations in a network simulator showed the efficiency of mDLS for networks with both mobile sensor nodes and mobile beacons.

Acknowledgment

This work was supported by the German Research Foundation under grant number TI254/15-1 (keyword: Geosens).

References

[1] J-sim: A component-based, compositional simulation environment, http://www.j-sim.org/, 2007.

[2] A. Baggio and K. Langendoen. Monte-carlo localization for mobile wireless sensor networks. pages 317–328, 2006.

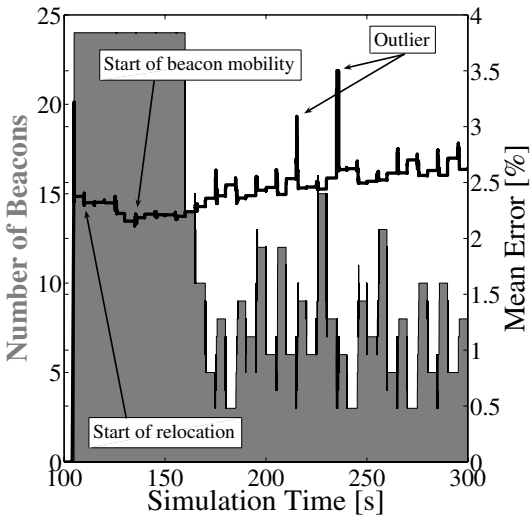

Figure 4. Behavior of the error function and the number of included beacon positions over simulation time.

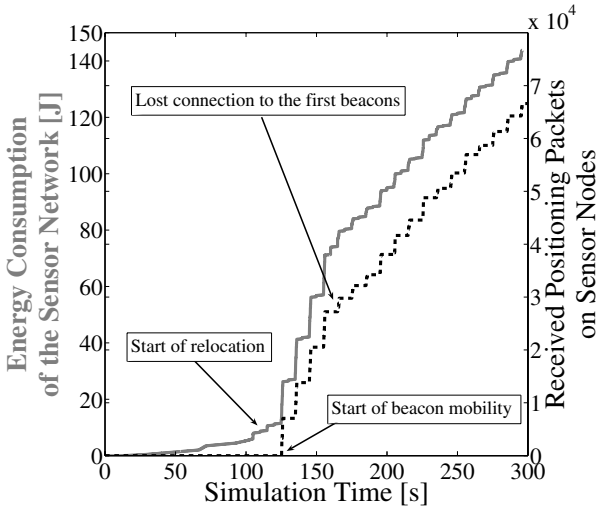

Figure 5. Energy consumption of the sensor network and the number of received packets of all sensor nodes over simulation time.

[3] C. Bendtsen, P. C. Hansen, K. Madsen, H. B. Nielsen, and M. Pinar. Implementation of QR up- and downdating on a massively parallel computer. *Parallel Computing*, 21:49–61, 1995.

[4] J. Daniel, W. Gragg, L. Kaufman, and G. Stewart. Reorthogonalization and stable algorithms for updating the gram-schmidt qr factorization. pages 772–795, 1976.

[5] J. Dongarra, C. Moler, J. Bunch, and G. Stewart. *LINPACK User's Guide*. SIAM, 1979.

[6] D. Fox, J. Hightower, L. Liao, D. Schulz, and G. Borriello. Bayesian filtering for location estimation. *IEEE Pervasive Computing*, 2:24–37, 2003.

[7] G. Golub and C. Reinsch. *Singular Value Decomposition and Least Square Solutions, Linear Algebra, Volume II of Handbook for Automatic Computations*. Springer Verlag, 1971.

[8] J. Hightower and G. Borriello. Particle filters for location estimation in ubiquitous computing: A case study. pages 88–106, 2004. Nottingham, England.

[9] L. Hu and D. Evans. Localization for mobile sensor networks. pages 45–57, 2004.

[10] D. Koutsonikolas, S. Das, and Y. C. Hu. Path planning of mobile landmarks for localization in wireless sensor networks. page 86, 2006.

[11] C. Kwok, D. Fox, and M. Meila. Adaptive real-time particle filters for robot localization. pages 2836–2841, 2003.

[12] C. L. Lawson and R. Hanson. *Solving Least Squares Problems*. Englewood Cliffs, NJ: Prentice-Hall, 1974.

[13] C. Lucas. *Updating the QR Factorization and the Least Squares Problem*. 2006.

[14] W. Murphy and W. Hereman. Determination of a position in three dimensions using trilateration and approximate distances. 1995. Colorado, USA.

[15] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, USA, 1992.

[16] F. Reichenbach. *Resource-aware Algorithms for exact Localization in Wireless Sensor Networks*. PhD thesis, University of Rostock, 2007.

[17] F. Reichenbach, A. Born, J. Salzmann, D. Timmermann, and R. Bill. Dls: A resource-aware localization algorithm with high precision in large wireless sensor networks. pages 247–254, 2007. Hanover, Germany.

[18] F. Reichenbach, A. Born, D. Timmermann, and R. Bill. A distributed linear least squares method for precise localization with low complexity in wireless sensor networks. pages 514–528, 2006. San Francisco, USA.

[19] F. Reichenbach and D. Timmermann. On improving the precision of localization with minimum resource allocation. pages 1093–1098, August 2007. Honolulu, Hawaii, USA.

[20] G. Stewart. *The Decompositional Approach To Matrix Computation*. Computing in Science and Enginieering, 2000.

[21] N. Strobel, S. Spors, and R. Rabenstein. Joint audio-video object localization and tracking. *IEEE Signal Processing Magazine*, 18(1):22–32, January 2001.