A Distributed Linear Least Squares Method for Precise Localization with Low Complexity in Wireless Sensor Networks

Frank Reichenbach[1], Alexander Born[2], Dirk Timmermann[1], and Ralf Bill[2]

[1] University of Rostock, Germany
Institute of Applied Microelectronics and Computer Engineering
{frank.reichenbach,dirk.timmermann}@uni-rostock.de
[2] University of Rostock, Germany
Institute for Geodesy and Geoinformatics
{alexander.born,ralf.bill}@uni-rostock.de

Abstract. Localizing sensor nodes is essential due to their random distribution after deployment. To reach a long network lifetime, which strongly depends on the limited energy resources of every node, applied algorithms must be developed with an awareness of computation and communication cost. In this paper we present a new localization method, which places a minimum computational requirement on the nodes but achieves very low localization errors of less than 1%. To achieve this, we split the complex least squares method into a less central precalculation and a simple, distributed subcalculation. This allows precalculating the complex part on high-performance nodes, e.g. base stations. Next, sensor nodes estimate their own positions by simple subcalculation, which does not exhaust the limited resources. We analyzed our method with three commonly used numerical techniques - normal equations, qr-factorization, and singular-value decomposition. Simulation results showed that we reduced the complexity on every node by more than 47% for normal equations. In addition, the proposed algorithm is robust with respect to high input errors and has low communication and memory requirements.

1 Introduction

The increasing miniaturization in the semiconductor field is leading to the evolution of very small and low-cost sensors [1]. Due to their small size they are strongly limited with respect to processor capacity, memory size and energy resources. Several thousands of such sensor nodes get into wireless contact with each other and form large ad hoc sensor networks. A wireless sensor network (WSN) will be placed around a field of interest or within an object. The sensor nodes are able to monitor different environmental parameters and to transmit them to a beacon or an infrastructure node. WSN enable new applications such as timely detection of wood fire or monitoring of artificial dikes.

The resulting data are only meaningful when combined with the geographical position of the sensor. Possible positioning technologies are the Global Positioning System (GPS) or the Global System for Mobile Communication (GSM) [2],[3]. These systems are however, due to the size of the equipment, the high prices and the high energy requirements, unsuitable for miniaturized sensor nodes and could only be used for a small number of nodes [4].

In this paper we present a new approach to energy-saving determination of unknown coordinates with a high precision. Using this method, the calculations are split between the resource-limited sensor nodes and the high-performance base station.

This paper is structured as follows: In Section 2 we give a basic overview of the methods for positioning in wireless sensor networks. In Section 3 we describe the position estimation based on relationships to known points. Then, in Section 4, we examine the complexity of three classical solution techniques in order to compare them with our new method. Next, we present in Section 5 our new approach to split the least squares method with the aim to minimize the load on the sensor nodes. Furthermore, the new method is analyzed with respect to complexity, memory requirement and communication effort. After discussing the simulation results in Section 5, we finally conclude the paper with Section 6.

2 Related Work

For the above-mentioned reasons, existing positioning techniques (e.g. GPS) cannot be integrated on all sensor nodes. The number of nodes with known position has to be limited. These nodes are referred to here as beacons, with the remaining nodes classed as sensor nodes. For the positioning of the sensor nodes we distinguish between approximate and exact methods.

Approximative Localization Many approximate approaches for the determination of sensor nodes exist in literature. These algorithms are resource-efficient but also result in higher positioning errors. Examples of such approaches are the hybrid methods [5], the Coarse Grained Localization [6], by using local coordinate systems [7], the Approximate Point in Triangulation-algorithm (APIT) [8], and the Weighted Centroid Localization (WCL) [9].

Exact Localization In contrast to approximation methods, exact methods use the known beacon-positions and the distances to the sensor nodes in order to calculate their coordinates through the solution of non-linear equations. Using a minimum of three beacons (in two dimensions), the coordinates of the sensor nodes may be determined using intersection. The use of more than three beacons gives more information in the system and allows the refinement of the position and the detection and removal of outlying observations. The least squares method (LSM) is used for the solution of the simultaneous equations. The LSM produces

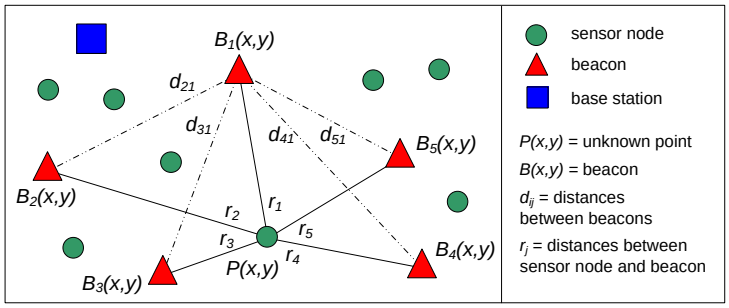

Fig. 1. Sensor network with one unknown point and beacons as reference points. Here beacon one was chosen as linearization tool.

accurate results, however it is complex and resource-intensive and therefore not feasible on resource-limited sensor nodes. Savvides et al. described methods to overcome these problems in [10]. Kwon et al. presented a distributed solution using least squares whereby errors in acoustic measurements can be reduced [11]. Ahmed et al. published a new approach to combine the advantages of absolute and relative localization methods [12]. Moreover, Karalar et al. developed a low-energy system for positioning using least squares which may be integrated on individual sensor nodes [13]. A general overview of distributed positioning systems is given by Langendoen and Reijersin [14].

We demand exact localization methods that work on tiny sensor nodes with high limited energy resources. To achieve that, we transfer the complex calculations such as matrix multiplication, matrix inversion, and eigenvalue determination to the base station that can be e.g. a powerful desktop computer or a more efficient node in the network. Consequently, only simple calculations have to be executed on the sensor nodes. Additionally, we reduce the communication and memory requirements by optimizations of the proposed algorithm.

3 Background: Linearization and Least Squares Method

Estimating the position of an unknown point $P(x, y)$ requires in two-dimensions at least three known points (see Figure 1). With m known coordinates $B(x_i, y_i)$ and its distances r_i to them we obtain:

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

This system of equations must be linearized with either Taylor series [15] or a linearization tool [16]. Although the linearization tool is not as exact as the Taylor series, it requires no mathematical differentiation and it is suitable for a distributed implementation (discussed later in Section 5). Thus, we use the j'th

equation of (1) as the linearization tool. By adding and subtracting x_j and y_j to all other equations this leads to:

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

With the distance r_j (r_i) that is the distance between the unknown point and the j'th (i'th) beacon and the distance d_{ij} that is the distance between beacon B_i and B_j this leads, after resolving and simplifying, to:

$$(x - x_j)(x_i - x_j) + (y - y_j)(y_i - y_j) = \frac{1}{2}\left[r_j^2 - r_i^2 + d_{ij}^2\right] = b_{ij}. \tag{3}$$

Because it is not important which equation we use as a linearization tool, $j = 1$ is sufficient. This is equal to choosing the first beacon and if $i = 2, 3, \ldots, m$ this leads to a linear system of equations with $m - 1$ equations and $n = 2$ unknowns.

$$\begin{array}{rcl} (x - x_1)(x_2 - x_1) + (y - y_1)(y_2 - y_1) & = & \frac{1}{2}\left[r_1^2 - r_2^2 + d_{21}^2\right] = b_{21} \\ (x - x_1)(x_3 - x_1) + (y - y_1)(y_3 - y_1) & = & \frac{1}{2}\left[r_1^2 - r_3^2 + d_{31}^2\right] = b_{31} \\ \vdots & & \\ (x - x_1)(x_m - x_1) + (y - y_1)(y_m - y_1) & = & \frac{1}{2}\left[r_1^2 - r_m^2 + d_{m1}^2\right] = b_{m1} \end{array} \tag{4}$$

This system of equations can be written in the matrix form:

$$A\mathbf{x} = \mathbf{b} \tag{5}$$

with:

$$A = \begin{pmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \\ \vdots & \vdots \\ x_m - x_1 & y_m - y_1 \end{pmatrix}, \mathbf{x} = \begin{pmatrix} x - x_1 \\ y - y_1 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} b_{21} \\ b_{31} \\ \vdots \\ b_{m1} \end{pmatrix}. \tag{6}$$

This is the basic form that now has to be solved using the linear least squares method.

3.1 Solving the Linear Least Squares Problem

Due to the fact that overdetermined systems of equations with $m >> n$ have not exact one solution for $A\mathbf{x} = \mathbf{b}$, we have to apply the L2-norm [17]. This is also called the Euclidean Norm, which minimizes the sum of the squares:

$$\underset{x \in \Re^n}{Minimize} \ ||A\mathbf{x} - \mathbf{b}||^2. \tag{7}$$

To summarize (see Figure 2), linear systems of equations can be solved iteratively using Splitting techniques or directly with the normal equations or

orthogonal factorization. Existing techniques are numerous but often the differences between them are small. For this reason, we focus our studies on three popular methods - normal equations, qr-factorization, and singular-value decomposition. For all others we recommend [18].

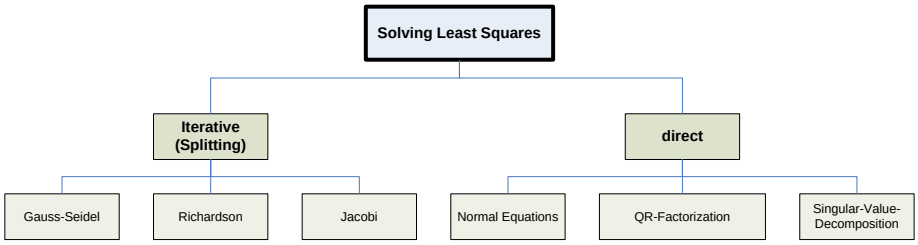

Fig. 2. Classification of common methods to solve a linear system of equations.

Normal Equations A trivial solution of the least squares problem is to reconvert after $\mathbf{x}$. In this case, the unique solution of $A\mathbf{x} \approx \mathbf{b}$ is given by:

$$||A\mathbf{x} - \mathbf{b}||^2 \rightarrow A^T A\mathbf{x} = A^T\mathbf{b}. \tag{8}$$

Solving normal equations is a good choice if the linear system has many more equations than unknowns, i.e. $m >> n$, because after the multiplication $A^T A$ the result is only a quadratic $[n \times n]$-matrix. That decreases the following computation and makes it easy to implement in software. However, the numerical difficulties that can occur sometimes determine a completely wrong position, which leads to orthogonal techniques.

QR-Factorization Orthogonal matrices transform vectors in different ways while they keep the length of the vector. Moreover, orthogonal matrices are invariant against the L2-norm, i.e. errors are not increased.

The qr-factorization transforms overdetermined linear systems of equations of the form $A\mathbf{x} \approx \mathbf{b}$ in a triangular system with the same solution, because it is:

$$\|A\mathbf{x} - \mathbf{b}\|^2 \rightarrow ||Q \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \mathbf{x} - \mathbf{b}||^2 = || \begin{pmatrix} R_1 \\ 0 \end{pmatrix} \mathbf{x} - Q^T\mathbf{b}||^2, \tag{9}$$

where Q is an orthogonal matrix (meaning that $Q^T Q = I$) and R_1 is an upper triangular matrix. This factorization is a standard method in numerics, is robust and stable to execute. In addition, the processing of rank defect matrices is possible.

Singular-Value Decomposition A second method that we want to explain is the singular-value decomposition. If A has full rank and is a $[m \times n]$-matrix with $m > n$, then we can transform $A\mathbf{x} \approx \mathbf{b}$ in a diagonal system with the solution:

$$||A\mathbf{x}-\mathbf{b}||^2 \rightarrow ||U \begin{pmatrix} S_1 \\ 0 \end{pmatrix} V^T\mathbf{x}-\mathbf{b}||^2 = || \begin{pmatrix} S_1 \\ 0 \end{pmatrix} V^T\mathbf{x}-U^T\mathbf{b}||^2 = || \begin{pmatrix} S_1 \\ 0 \end{pmatrix} \mathbf{y}-U^T\mathbf{b}||^2, \tag{10}$$

where U is an orthogonal $[m \times m]$-matrix, V is an orthogonal $[n \times n]$-matrix and S is a diagonal matrix. The original algorithm has been implemented by Golub and Reinsch in [19]. This algorithm is also robust and stable to compute, but requires high computation effort due to root and eigenvalue operations.

4 Analysis: Complexity of the Methods

In the following, we will analyze the complexity of all three introduced methods. Although the literature offers numerous specifications, later we will reduce specific parts of the calculation. In order to mathematically define the complexity, we count the number of floating point operations (flops), which is commonly used in literature. The required number of computation cycles strongly depends on the hardware. Therefore, we count for every operation one flop whether it is an addition, subtraction, multiplication or division[3]. At this stage, we do not consider copying-operations in the memory, because this operation depends on the individual implementation of the algorithm.

As before, we will confine the explanation to two dimensions. Due to the linearization with a linearization tool the matrix A and the vector $\mathbf{b}$ have $(m-1)$-rows. For a clearer understanding we calculate with k-rows and substitute at the end: $m = k + 1$.

4.1 Complexity of the Normal Equations

The linear system of equations:

$$\mathbf{x} = \left(A^T A\right)^{-1} A^T \frac{1}{2}[r_1^2 - \mathbf{r}^2 + \mathbf{d}^2] \tag{11}$$

has to be solved. We divide the calculation into the following complexities.

1. Multiplying the $[n \times k]$-matrix A^T with the $[k \times n]$-matrix A leads to $\frac{n(n+1)}{2}$ flops[4].
2. The $[n \times n]$-matrix, resulting from 1., must be inverted[5] with a complexity of n^3.

[3] It should be noted that in the arithmetic unit of a processor a division is a more complex operation than an addition. We will focus on theoretical analysis.
[4] Some operations can be saved by multiplying a transposed matrix with itself.
[5] The inversion of a matrix is very complex with n^3 flops (see [20]).

3. The $[n \times n]$-matrix, resulting from 2., must be multiplied with the $[n \times k]$-matrix A^T, which costs $2n^2k - nk$ flops. This leads to the precalculated Matrix A_p.
4. The matrix A_p must be multiplied with the k-vector $\mathbf{b}$. This step has a complexity of $2kn - n$ flops.
5. The calculation of $\mathbf{b}$ needs $5k + 1$ flops.

With $k = m - 1$ this leads to a total complexity of $15m - 5$ for the least squares method with m beacons and $n = 2$ unknowns.

4.2 Complexity of the QR-Factorization

Now, the complexity for the qr-factorization has to be studied. First the partial matrices of Q have to be determined, which in our case for $n = 2$ are limited to only two matrices; Q_1 and Q_2.

1. The calculation of Q_1 needs $\frac{5}{2}k^2 + \frac{9}{2}k - n + 5$ flops.
2. The calculation of Q_2 needs $\frac{5}{2}(k-1)^2 + \frac{9}{2}(k-1) - n + 5$ flops, because we do not need to consider the last line in the calculation.
3. The multiplication of Q_1Q_2A, where Q_1 and Q_2 have the size $k \times k$, needs $2k^3 + 3k^2 - 2k$ flops.
4. The calculation of $\mathbf{b}$ needs $5k + 1$ flops (see 4.1).
5. For the calculation of $Q_1^T\mathbf{b}$ it has to be considered, that only the upper two rows are needed for the multiplication. This leads to $8k - 4$ flops.
6. The calculation of $Rx \approx Q^T\mathbf{b}$, by back substitution, requires exactly 4 flops.

Summarized, the complexity of the qr-factorization is $4m^3 - 5m^2 + 13m - \frac{9}{2}$ flops.

4.3 Complexity of the SV-Decomposition

The sv-decomposition is more complicated than the previously discussed procedures, because a determination of eigenvalues is necessary, which has to be determined using several methods. In principal, $9n^3 + 8n^2k + 4k^2n$ flops are needed for the computation of U, V and S referring to [20]. With $n = 2$ this leads to $8k^2 + 24k + 48$ flops. Additionally, $\mathbf{x}$ must be determined with $S_1y = U_1^Tb$ and $\mathbf{x} = Vy$. Finally, this leads to a complexity of $8m^2 + 25m + 54$ flops for the svd.

Now, after discussing the standard methods, we will explain the new approach for distributing them in wireless sensor networks.

5 New Approach: Distributing Least Squares Problem

Linearizing non linear equations with a linearization tool has a significant advantage. All elements in matrix A are beacon positions $B_1(x, y)..B_m(x, y)$ only. Moreover, vector $\mathbf{b}$ consist of distances between the unknown sensor node and all

beacons $r_1..r_m$ and distances $d_1..d_m$ between the first beacon and all others. Consequently, we split the complex computation into two parts - a less complex and a very simple part. First, we precalculate matrix A into a different form, which strongly depends on the solution method and will be discussed later. Then, with this precalculated form, a simple subcalculation starts. This splitting method works in a similar fashion with all three solution methods.

So far, in sensor networks it is desired to execute the entire localization algorithm on every node; completely distributed. With this assumption, the precalculation of a least squares method would be exactly the same on every sensor node, because matrix A is the same for every sensor node in a static network. This wastes limited resources and produces high redundancy.
Now, with our distributed approach, the high-performance base station executes the complex precalculation and the resource-aware sensor nodes estimate their own position with the simple subcalculation. In this subcalculation all sensor nodes use the same precalculated information combined with their individual measured distances to every beacon. This assumes that all beacons are able to communicate with every sensor node directly. This is difficult to achieve in real environments, due to obstacles and limited transmission ranges, but can be solved by multi-hopping techniques [21]. Thereby, beacons send packets over neighboring nodes hop by hop to the destination node. The number of hops is an indicator for a distance to the beacon. However, the focus in this paper is not the distance determination, but the localization process. Before we describe the methods in detail, we will postulate the entire algorithm.

Distributed Least Squares Algorithm

Step 0: Initialization Phase:
All beacons send their position $B(x, y)$ to the base-station.
Step 1: Complex Precalculation Phase (central):
Base station builds matrix A and vector d_p.
Starting the complex precalculation (result strongly depends on the solution method).
Step 2: Communication Phase (distributed):
Base station sends precalculated data and vector d_p to all sensor nodes.
Step 3: Simple Subcalculation Phase (distributed):
Sensor nodes determine the distance to every beacon $r_1..r_m$.
Sensor nodes receive the precalculated data and vector d_p, built vector b and estimate their own position P_{est} autonomously.

In the next section, we will adapt the algorithm to all three solution methods and analyze the results in detail.

5.1 Reduced Complexity

In Section 4.1 we already analyzed the complexity of all three solution methods. At this point it is important to know what we can save on the sensor nodes without complex precalculations regarding only the remaining subcalculation.

Normal Equations We assume the constellation of the network in Figure 3 with sensor nodes, beacons, and a base station. On bases of (11) the base station precalculates $A_p = (A^T A)^{-1} A^T$ and $\mathbf{d}_p = \mathbf{d}^2$. The matrix A and vector $\mathbf{d}_p$ are sent to all sensor nodes. Together with the distances $\mathbf{r}$ to all beacons, which every sensor node must determine itself, the subcalculation starts:

$$\mathbf{x} = A_p \frac{1}{2}(r_1^2 - \mathbf{r}^2 + \mathbf{d}_p) \tag{12}$$

This computation requires $8m - 11$ flops.

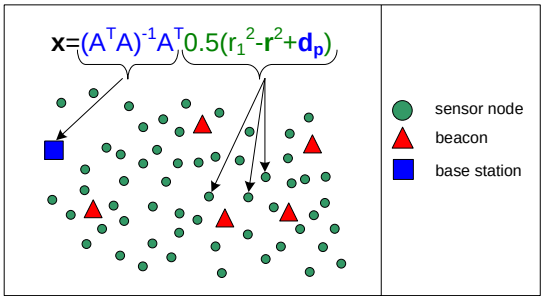

Fig. 3. Splitting the normal equations.

QR-Factorization Here, the base station transmits the precalculated matrices Q, R, and also vector $\mathbf{d}_p$. With this, every sensor node reduces computation to the following:

1. Creating $\mathbf{b}$ needs $4k + 1$ flops (We use the substitution k=(m-1) again.).
2. Multiplying $\mathbf{y} = Q_1^T \mathbf{b}$, where Q_1^T is a $[2 \times k]$-matrix and $\mathbf{b}$ is a k-vector needs $8k - 4$ flops.
3. Solving $R\mathbf{x} \approx \mathbf{y}$ by back-substitution finally requires 4 flops.

Summing up, the subcalculation requires a reduced complexity of $12m - 11$.

SV-Decomposition As the third method, the sensor nodes receive from the base station the matrices U, S, V and $\mathbf{d}_p$ and have to compute:

1. Creating $\mathbf{b}$ requires $4k + 1$ flops.

2. Solving $U_1^T\mathbf{b}$, where U_1^T is a $[2 \times k]$-matrix and $\mathbf{b}$ is a k-vector, results in the vector $\mathbf{z}$ and requires $4k - 2$ flops.
3. Then, $\mathbf{y}$ is calculated by back-substitution of $S_1\mathbf{y} = \mathbf{z}$. Due to the two zero elements in S_1 this requires only 2 flops.
4. The last part of the calculation requires 6 flops, where $\mathbf{x}$ is determined by $\mathbf{x} = V\mathbf{y}$.

Adding all together leads to a reduced complexity of $8m + 1$ flops.

To compare all complexities Figure 4 was created. The complete solutions of the least squares method require much more operations than the reduced methods. Especially qr-factorization and sv-decomposition would exhaust the sensor nodes resources, because with only 50 beacons more than 10^4 floating point operations are required. In contrast, the normal equations are less complex. Obviously, all reduced calculations decrease the complexity and make the localization on resource aware sensor nodes feasible. However, a low communication is also demanded, which we will analyze next.

5.2 Communication Effort

As already described, communicating between sensor nodes is critical and must be minimized. Particularly, sending data over long distances stresses the energy capacity of sensor nodes. Communication between base station and beacons is less critical and must be preferred if possible. Therefore, we classify communication in two phases. In an uncritical phase, all beacons send their positions to the base station. This causes no energy loss on the sensor nodes. Additionally, in a critical phase the base station sends precalculated information to the sensor nodes that have, in theory, to receive only. Practically, transmitting/sending is never lossless, due to errors in the transmission channel and protocols that require acknowledge packets etc. Furthermore, the base station cannot reach every

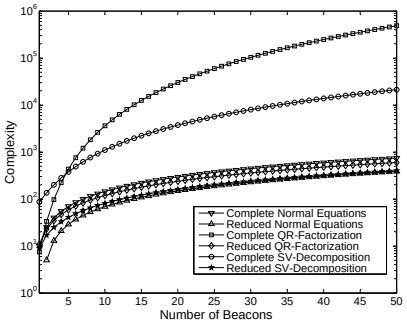

Fig. 4. Complexity of all complete and all reduced calculations in flops.

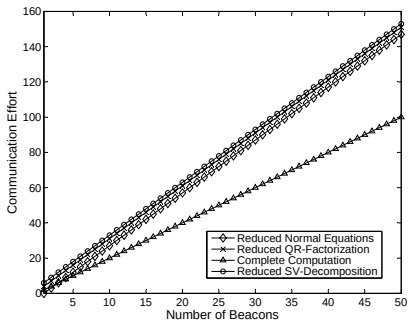

Fig. 5. Communication effort of all reduced and the traditional method.

sensor node in one hop, which demands multi-hopping over some nodes.

Normal Equations Here, we focus on a theoretical comparison of the algorithms that is, for the moment, independent of protocol definitions and media access operations. Hence, every sensor node must receive the precalculated matrices and vector d_p. The communication effort directly depends on the used solution method.
At the normal equations the sensor node receives matrix A_p and vector $\mathbf{d}_p$, with $[n \cdot (m-1) + (m-1)]$ elements. This results in receiving $(3m-3)$ elements.

QR-Factorization This requires transmitting Q_1^T with $[2 \cdot (m-1)]$ elements, matrix R with only two elements and also $\mathbf{d}_p$. Summarized, $(3m+1)$ elements must be send in the critical phase.

SV-Decomposition By applying the sv-decomposition, $(3m+3)$ elements must be transmitted, because S_1 consists of two elements, U_1^T has $[(m-1) \cdot 2]$ elements, the quadratic matrix V consist of two elements and vector d_p of (m-1) elements.

We compared all communication efforts in Figure 5. The communication effort of all reduced methods is relatively low, comparing to the traditional method with much computation overhead[6]. The direct solution of the normal equations minimizes communication. As an example, with 50 beacons not more than 100 elements must be received.

5.3 Memory Considerations

Normal Case The reduced calculations must be feasible on sensor nodes with a very small memory, mostly not more than a few kilobyte RAM. In our case, the memory consuming operation is always the multiplication of $\mathbf{b}$ with the precalculated data. Without optimizations this would be for the three methods:

1. $A_p \cdot \frac{1}{2} \cdot (r_1^2 - \mathbf{r}^2 + \mathbf{d}_p)$
2. $Q_1^T \cdot \frac{1}{2} \cdot (r_1^2 - \mathbf{r}^2 + \mathbf{d}_p)$
3. $U_1^T \cdot \frac{1}{2} \cdot (r_1^2 - \mathbf{r}^2 + \mathbf{d}_p)$

In the worst case A_p, Q_1^T, U_1^T, and $\mathbf{r}$ plus $\mathbf{d}_p$ must be stored temporarily in memory before the execution on the sensor node can start. In more detail, $[2 \cdot (m-1)] + (m-1) + (m-1) = (4m-4)$ elements must be stored. On common microcontrollers, that are presently integrated on sensor node platforms, every element is stored in floating point representation as a 4 byte number. Accordingly, with $m = 100$ beacons, already 0.796 kb must be allocated, for localization

[6] "Complete Computation" stands for the classical method, where all beacons send their positions directly to all sensor nodes. Thus, every sensor node must receive at least two positions to determine its own position that results in $2m$ elements.

only. Normally, the localization task is part of the middleware that has to execute many more tasks. Besides, temporary variables are needed that increases the memory consumption. Given these facts, we studied the critical operations in more detail and will describe optimizations in the next section.

Optimizations In reality, input data for sensor nodes arrive in packets and will be disassembled into a serial data stream. Due to problems in the transmission channel (e.g. different paths or transmission errors) a sorted order of the incoming packets cannot be guaranteed. The data can arrive in an unsorted form and the calculation begins after receiving all data.

However, the reduced calculation has a further useful quality. Individual calculations of $A_p \cdot \mathbf{b}$ can be executed after the arrival of only some elements without collecting all data. Only one accumulator for the position $P_{est}(x)$ and one for the position $P_{est}(y)$ of the sensor node is needed. If we define $W = A_p, Q_1^T, U_1^T$ independently from the specific method, the following multiplication will always result:

$$\begin{pmatrix} w_{11} \cdots w_{1(m-1)} \\ w_{21} \cdots w_{2(m-1)} \end{pmatrix} \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_{(m-1)} \end{pmatrix} = \begin{pmatrix} w_{11} \cdot b_1 + \ldots + w_{1(m-1)} \cdot b_{(m-1)} \\ w_{21} \cdot b_1 + \ldots + w_{1(m-1)} \cdot b_{(m-1)} \end{pmatrix}.$$

With w_{ij} $(i = 1..2, j = 1..m-1)$ and b_s $(s = 1..m-1)$ the following assumptions can be made. If elements with $j = s$ are available, an immediate multiplication of $w_{ij} \cdot b_s$ and a subsequent accumulation in $P_{est}(x, y)$ is possible. The index i distinguishes into which accumulator it must be written; $P_{est}(x)$ at $i = 1$ or $P_{est}(y)$ at $i = 2$. Finally, the optimized reduced calculation requires a worst-case calculation time of $(m-1)/2$ if the elements arrive in reverse order.

To avoid the case of unsorted data it is also possible to send the elements w_{ij}, d_s in appropriate tuples, e.g. $w_{11}, d_1; w_{12}, d_2; \ldots; w_{ij}, d_s$. In best case, space in memory has then to be reserved for only a few temporary variables and two accumulators which reduces memory consumption to a minimum.

5.4 Example

The algorithm is intended for implementation and execution on a sensor platform. Due to this, we have represented the results in Table 1. We assume $m = 100$ beacons and $n = 2$ for the second dimension[7]. Furthermore, we assume that a floating point representation requires 4 byte of memory.

The direct calculation by normal equations requires minimal computation whereas the calculation by the qr-factorization requires lowest data traffic. However, the normal equations are numerically instable and the sensitivity of the

[7] It must be considered here to add x_1 and y_1 to the final coordinates due to the linearization in (6).

linear equalization problem can deteriorate. Remarkable are the saving of the calculation for the qr-factorization and the sv-decomposition, because the precalculation requires the largest expenditure. Summarized, this overall comparison shows the potential advantages of the new distributed approach.

5.5 Noisy Observations

Estimating the position basically requires beacon positions and distances. Due to various error influences, e.g. imprecise measuring of the signal of flight or defective GPS-coordinates, the applied algorithm must be robust to input errors. For this reason, we studied the behavior of the described methods in different simulations concerning noisy distances $\mathbf{r}$ and noisy beacon positions $B(x, y)$. To realize this simulation, we substituted the exact value by a chosen random value out of a Gaussian distribution $r_{appr}, B_{appr} \sim \mathcal{N}(\mu_{exact}, \sigma^2)$. The exact value was the arithmetic mean and the variance σ^2 was a parameter. We always simulated with 500 nodes, where 5% were beacons and the rest sensor nodes. We executed numerous series and averaged the results to avoid a strong influence of outliers. We created a test field with the size 100×100 where all nodes were placed by a uniform distribution. We determined in all simulations the averaged relative localization error.

In a first simulation we compared the achievable precision of the three methods that solves the least squares method. Figure 6 shows that with exact input the error for all three reduced methods ranges only in an interval of $0.5 \cdot 10^{-14}$. Following simulations with noisy input showed the same differences, leading us to continue simulating using only the normal equations.

Next, a simulation with noisy distances was executed (see Figure 7). An increasing variance of the Gaussian distribution resulted in an increasing relative localization error up to 6% at $\sigma^2 = 100$. The standard deviation is relatively small.

Table 1. Performance comparison with floating point numbers and 100 beacons. The communication effort and the memory capacity in the table refer to the reduced methods.

Algorithm	**Full Complexity [flops]**	**Reduced Complexity [flops]**	**Savings [%]**	**Communication Effort [bytes]**	**Memory Capacity [bytes]**
Normal Equation	1497	791	47.16	1188	≈ 1588
QR-Factorization	3951302	1201	99.97	1204	≈ 1588
SV-Decomposition	82556	803	99.03	1212	≈ 1588

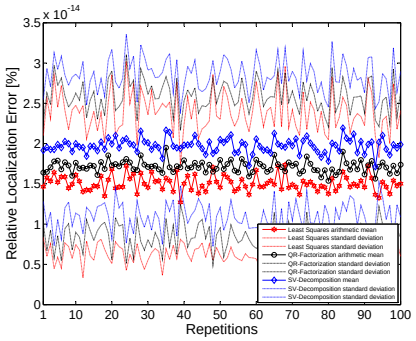

Fig. 6. Localization error with exact input.

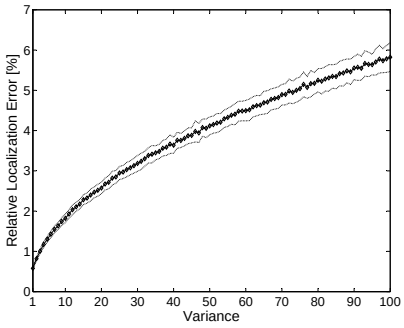

Fig. 7. Error with noisy distances.

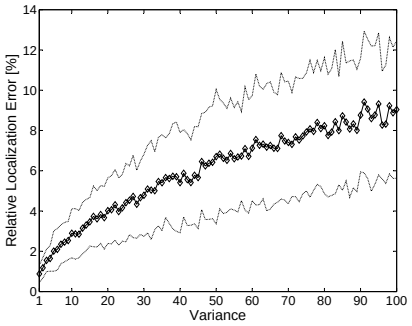

Fig. 8. Error with noisy beacon positions.

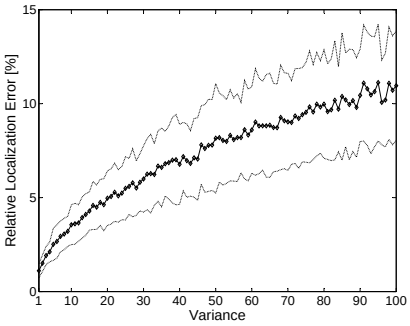

Fig. 9. Error with complete noisy input.

In more simulation series we studied the error for noisy beacon positions. This simulation describes the behavior of the algorithm in a slightly dynamic network, where the beacon positions can change after the precalculations are already executed and transmitted to the sensor nodes. The sensor nodes would determine new distances to the beacons but combine them with wrong beacon coordinates. Figure 8 shows the increasing error that rises over 8% at a variance of 100. The standard deviation is also higher compared to the previous simulation. Defective beacon positions influences the results strongly, but normally they are not the main problem. This means that the algorithm is able to manage slight changes in a dynamic network with an acceptable error.

In a last simulation we increased the variance for both, distances and beacon positions. Figure 9 shows the result. Here, the highest error occurs, as it was expected. However, these results show the robustness of the algorithm.

6 Conclusion

We have presented a new method for exact localization in resource-limited sensor networks by distributing the least squares method. Usually the calculation of this method is very complex with an increasing number of beacons. However, the use of the linearization tool enables us to split the complex calculation into a complex part, precalculated on the high-performance base station, and a very simple subcalculation on every sensor node. With low communication traffic the base station sends precalculated data to all sensor nodes. Sensor nodes must only receive data and compute their own position autonomously.

Simulations show that a complexity reduction of 99% (for qr-factorization and singular-value decomposition) and 47% (for normal equations), using 100 beacons, is achievable without increasing the communication requirements. Moreover, we described optimizations where the algorithm starts executing as soon as the first data arrive at the sensor node. This allowed high savings in the required memory capacity with only a few kilobytes of memory considered. Currently, we are studying the algorithm in extensive network simulations. In future, the implementation on a real sensor platform is planned.

Acknowledgment

This work was supported by the German Research Foundation under grant number TI254/15-1 and BI467/17-1 (keyword: Geosens). We appreciate comments given by Edward Nash, which helped us to improve this paper.

References

1. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: A survey. Computer Networks **38** (2002) 393–422
2. Bill, R., Cap, C., Kohfahl, M., Mund, T.: Indoor and outdoor positioning in mobile environments a review and some investigations on wlan positioning. Geographic Information Sciences **10** (2004) 91–98
3. Gibson, J.: The mobile communications handbook. CRC Press (1996)
4. Min, R., Bhardwaj, M., Cho, S., Sinha, A., Shih, E., Wang, A., Chandrakasan, A.: Low-power wireless sensor networks. In: International Conference on VLSI Design. (2001) 205–210
5. Savarese, C., Rabaey, J., Langendoen, K.: Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In: USENIX Technical Annual Conference. (2002) 317–327
6. Bulusu, N.: Gps-less low cost outdoor localization for very small devices. IEEE Personal Communications Magazine **7** (2000) 28–34
7. Capkun, S., Hamdi, M., Hubaux, J.P.: Gps-free positioning in mobile ad hoc networks. Cluster Computing **5** (2002) 157–167
8. Tian, H., Chengdu, H., Brian, B.M., John, S.A., Tarek, A.: Range-free localization schemes for large scale sensor networks. In: 9th annual international conference on Mobile computing and networking. (2003) 81–95

9. Blumenthal, J., Reichenbach, F., Timmermann, D.: Precise positioning with a low complexity algorithm in ad hoc wireless sensor networks. PIK - Praxis der Informationsverarbeitung und Kommunikation **28** (2005) 80–85
10. Savvides, A., Han, C.C., Srivastava, M.B.: Dynamic fine grained localization in ad-hoc networks of sensors. In: Seventh Annual ACM/IEEE International Conference on Mobile Computing and Networking. (2001) 166–179
11. Kwon, Y., Mechitov, K., Sundresh, S., Kim, W., Agha, G.: Resilient localization for sensor networks in outdoor environments. In: 25th IEEE International Conference on Distributed Computing Systems. (2005) 643–652
12. Ahmed, A.A., Shi, H., Shang, Y.: Sharp: A new approach to relative localization in wireless sensor networks. In: Second International Workshop on Wireless Ad Hoc Networking. (2005) 892–898
13. Karalar, T.C., Yamashita, S., Sheets, M., Rabaey, J.: An integrated, low power localization system for sensor networks. In: First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services. (2004) 24–30
14. Langendoen, K., Reijers, N.: Distributed localization in wireless sensor networks: A quantitative comparison. Computer Networks (Elsevier), special issue on Wireless Sensor Networks **43** (2003) 499–518
15. Niemeier, W.: Ausgleichsrechnung. de Gruyter (2002)
16. Murphy, W.S., Hereman, W.: Determination of a position in three dimensions using trilateration and approximate distances. (1999)
17. Gramlich, G.: Numerische Mathematik mit Matlab - Eine Einführung für Naturwissenschaftler und Ingenieure. dpunkt.verlag (2000)
18. Lawson, C.L., Hanson, R.: Solving Least Squares Problems. Englewood Cliffs, NJ: Prentice-Hall (1974)
19. Golub, G.H., Reinsch, C.: Singular Value Decomposition and Least Square Solutions, Linear Algebra, Volume II of Handbook for Automatic Computations. Springer Verlag (1971)
20. Golub, G.H., Loan, C.F.V.: Matrix Computations. The Johns Hopkins University Press (1996)
21. Niculescu, D., Nath, B.: Ad hoc positioning system (aps) using aoa. In: Proceedings of the IEEE Annu. Joint Conf. IEEE Computer and Communications Societies. (2003) 1734–1743