Splitting the Linear Least Squares Problem for Precise Localization in Geosensor 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. Large amounts of cheap and easily deployable wireless sensors enable area-wide monitoring of both urban environments and inhospitable terrain. Due to the random deployment of these sensor nodes, one of the key issues is their position determination. Noisy distance measurements and the highly limited resources of every sensor node, due to tiny hardware and small battery capacity, demand the development of robust, energy aware, and precise localization algorithms.
We believe this can be achieved by appropriately distributing the complex localization task between all participating nodes. Therefore, we use a linearization tool to linearize the arising non-linear system of equations into a linear form that can be solved by a distributed least squares method. It is shown in this paper that we can save with this new approach more than 47% of computation cost whilst maintaining a low network traffic. Additionally, we describe memory optimizations to process the complex matrix operations with only a few kilobyte of memory on the sensor node.

1 Introduction

1.1 Sensor Networks

Wireless sensor networks (WSN) are composed of hundreds of tiny electronic devices, able to sense the environment, compute simple tasks and communicate with each other. Gathered information (e.g. temperature, humidity etc.) are transmitted in a multi hop fashion over direct neighbors to a data sink, where the data is interpreted [1]. With methods such as self configuration and self organization the network reacts to node failures.

Due to the desired node size of only a few cubic millimeters, the dimensions of the communication module and the battery are critical. Consequently, the scarcest resource within a network is the available energy [2]. Therefore, achieving a long lifetime of the sensor network requires low power hardware as well as optimized algorithms.

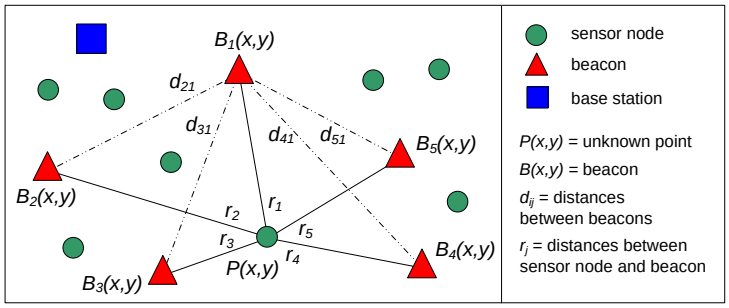

Fig. 1. Localization problem with one unknown point and five beacons (known points).

1.2 Problem Statement

After deploying the sensor network over an area of interest, initially the sensor nodes carry no position information. Sensor information are only useful if combined with their geographical position. Possible positioning technologies are the Global Positioning System (GPS), the Global System for Mobile Communication (GSM) or soon the European System Galileo [3],[4],[5]. However, these systems are unsuitable for miniaturized sensor nodes and could only be used for a small number of nodes, due to the size of the hardware, the high prices and the high energy requirements. Thus, it is a common technique to integrate an existing localization system on some more powerful nodes, further called beacons. Then, all remaining nodes estimate their own position with measurements such as distances to these beacons autonomously. Figure 1 shows an example constellation of a network with one unknown sensor node and five beacons.
For several reasons, a node's position is very important:

- Sensed data without a location where they were gathered are generally useless.
- Full covered sensor networks enable energy aware geographic routing.
- Self configuration and self organization are key mechanisms for robustness and can be easily realized with position information.
- In many applications the position itself is the information of interest.

A very comprehensive overview of geosensor networks can be found in [6]. In this paper we present a new approach to energy-saving determination of unknown coordinates with a higher precision compared to approximate positioning methods [7]. 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 localization in wireless sensor networks. In Section 3 we describe the position estimation based on relationships to known points. In particular two methods to linearize the non-linear system of equations are discussed. Then, in

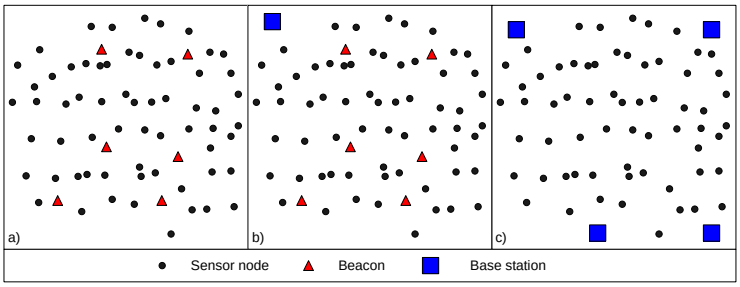

Fig. 2. Ad hoc connected sensor nodes: a) with beacons, b) with beacons and one fixed base station (**loosely coupled network**), c) without beacons and four base stations (**infrastructure case**).

Section 4, we study the robustness of our model to noisy distances and moreover, we examine the complexity of the least squares 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. We finally conclude the paper with Section 6.

1.3 Motivation

As a scalable and decentralized system the wireless sensor network permits the automatized and area-wide observation of distributed, hardly accessible properties or phenomena in motion. In relevant fields of geoinformatics the main issue is to adapt geometric-topological localization algorithms to sensor networks. Thus, it is conceivable to develop hybrid methods with high precision and low computation effort. This is achievable by e.g. meshed triangles [8] or Radial-Sweep methods [9] that can be applied in 2d as well as in 3d. Wireless geosensor networks enable new possibilities for timely detection of wood fire, monitoring of slope slipping, and "Precision Farming". In these applications sensor data is gathered and analyzed using extensive interpolation algorithms. As an example, precision farming has to be mentioned. Sensors are arbitrary distributed over an agricultural crop field - e.g. dropped off by airplane. These sensors may measure parameters like humidity, pH-value, etc. over a specific period of time. The data can be collected at the base station at the farmers house. Afterwards or online in a spatial information system, surface models are generated from the point related data. For that an accurate position of the sensor nodes is needed.

2 Related Work: Localization Algorithms in Sensor Networks

The localization process in a WSN is based on different network topologies. We described in Section 1.2 that most of the localization algorithms assume beacons,

initially knowing their own position. So, the first topology (shown in Figure 2a) consists of ad hoc connected sensor nodes and these beacons. In this case either the localization algorithm is processed fully distributed on all sensor nodes and/or additionally on beacons. This brings the advantage that every node is responsible for the computation of its own position with lowest communication traffic in the network.

Complementary, as shown in in Figure 2c, all sensor nodes transmit their data to the base stations (e.g. server, desktop-pc) in the network. Base stations relieve all sensor nodes from the complex computation tasks. However, this solution leads to the highest communication overhead and fails if the routes to the base stations are defective or the base stations are down. Beneficially, the base stations are part of an infrastructure that allows the determination of a geodetic date (origin, rotation and scale of a coordinate reference system). Thus, every sensor node can be defined in a world wide valid frame of reference system.

However, both topologies implies either overhead in communication or computation that qualifies a mixed topology in Figure 2b, where the incomplex tasks of the localization are processed on all sensor nodes and energy consuming tasks are delivered to the base station with respect to the communication overhead. This hybrid topology must be flexible enough to compensate all kinds of node failures to allow a fully distributed as well as a centralized computation. This yields in longest network lifetime.

Given these facts, all localization methods in WSNs can be classified into exact and approximate methods (see Figure 3), regarding the achievable precision and the complexity.

Approximated 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. Some approximate algorithms completely avoid the defective distance measurements to estimate a position.

For example Bulusu postulated the "Coarse Grained Localization" in [7]. In this algorithm in the first phase, all beacons send their position $B(x, y)$ to all sensor nodes within their transmission range. In the second phase, all sensor nodes calculate their own position by a centroid determination from all n positions of the beacons in transmission range. This algorithm requires only receiving few data on the sensor nodes and minimal computation overhead, but lead to higher localization errors. However, these errors can be acceptable in specific applications.

Tian et al. completely avoid distances in their approach [10]. First, triangles are combined with all beacon coordinates in the field. Then every unknown checks by a "Point in Triangulation-test" on which triangle surface it is situated. In this test only neighbor relations are used. A following intersection test with

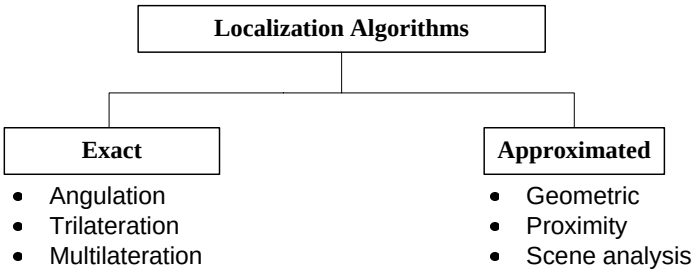

Fig. 3. Classification of common localization techniques in sensor networks.

all filtered triangles shrinks the region where the unknown is probably located.

More examples of such geometric or proximity approaches are the hybrid methods [11], by using local coordinate systems [12], and the "Weighted Centroid Localization" [13].
Scene analysis with imagery is also a well researched method, but less important in sensor networks due to the size and the high energy consumption of the cameras.

Exact Localization In contrast to approximate algorithms, 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 (trilateration). 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 (multilateration). The least squares method (LSM) is used for the solution of the simultaneous equations. The LSM produces best linear unbiased estimators (BLUE) in a stochastical sense, however it is complex and resource-intensive and therefore rarely feasible on resource-limited sensor nodes.

Savvides et al. described methods to overcome these problems in [14]. For example, an iterative multilateration algorithm starts by estimating the position of the sensor node with the maximum number of beacons using atomic multilateration. When a sensor node estimates its location, it becomes a beacon. This process repeats until the positions of all the nodes that eventually can have three or more beacons are estimated. In a further collaborative multilateration algorithm it is possible to compute the position of sensor nodes with less than three beacons, which normally are not able to estimate its position.

Next, Kwon et al. presented a distributed solution using least squares whereby errors in acoustic measurements can be reduced [15]. Ahmed et al. published a

new approach to combine the advantages of absolute and relative localization methods [16]. Moreover, Karalar et al. developed a low-energy system for positioning using least squares which can be integrated on individual sensor nodes [17]. A general overview of distributed positioning systems is given by Langendoen and Reijersin [18].

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

3 Background: Determination of a Position With Noisy Observations

3.1 Errors in Distance Measurement

All exact localization methods need distances (lateration) or angles (angulation) to known beacons, which are measured using different techniques. Due to the fact that measuring angles needs additional hardware on the sensor nodes (e.g. an array of antennas), the distance determination is the most common technique on sensor nodes. Dominating methods in geodesy such as laser or image correlation cannot be adapted to sensor networks, due to the limited energy resources. For this reason, only phase shifting of a signal, time of signal flight, and received signal strength measurement are feasible.
Figure 4 shows a characteristical received signal strength measurement over the distance for an 868 MHz Chipcon transceiver. The demand for high precision in measuring will not be achieved because:

- Reflections of radio waves at obstacles cause multi-path propagation.
- Other electrical fields in the environment interfere with the transmitted radio waves.
- The resolution of the received signal strength indicator can be limited by the transceiver hardware.
- The required line of sight between nodes cannot be guaranteed.
- Node mobility influences the measurement.

Therefore, noisy distances must be considered when developing a robust localization algorithm.

3.2 Building a Model

Estimating the position of an unknown point $P(x, y)$ in two-dimensions requires at least three known points (see Figure 1). In this paper we will confine the

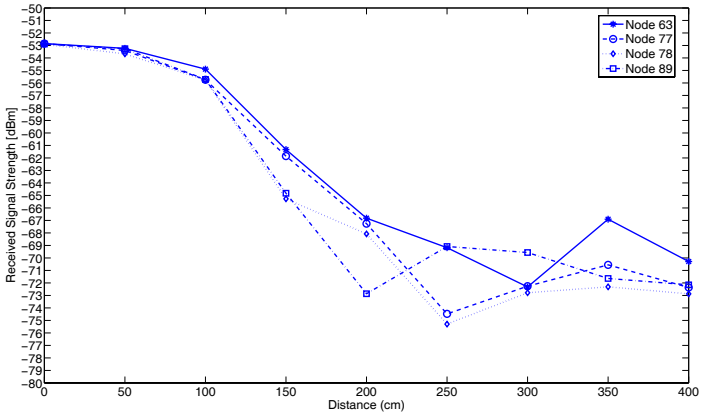

Fig. 4. Indoor received signal strength measurements in the institute's meeting room with our sensor node platform for four different modules.

explanation to two dimensions. Obviously an extension into the third dimension is possible by introducing the additional coordinate parameter z. This would not have any effect on the following calculation in principle. By using the formula of plane trigonometry the method of intersection will be most adequate, which uses the distances between the sensor nodes and the beacons for the calculation of the unknown coordinates. Due to inevitable measurement errors the observations are stochastic (expressed by σ^2). Therefore, an averaging or a statistically optimal estimate with simultaneous use of all observations by adjustment is feasible. Nevertheless the advantages of using the adjustment approach are:

- Unambiguous, statistically optimal coordinate determination
- Statistical tests to detect gross errors and outliers
- Quality evaluation of the results
  - Unambiguous accuracy prediction for the coordinates
  - Information about the reliability of the results

A detailed explanation of the adjustment is given [19].
Adjustment methods can be described as follows. A number of m observations is given, which may vary according to the character and accuracy: e.g. distance measurements, angles, local coordinates. With these values the $(m, 1)$-observation vector L is given. The u-unknown parameters X have to be determined, e.g. the coordinates for the unknown sensor nodes. These parameters are represented by the vector of the unknowns $\mathbf{X}$. The set of observations L have to be mapped onto the set of unknowns X: $L \rightarrow X$. The observations and the unknowns can be distinguished in:

$$\begin{aligned} \widetilde{L}, \widetilde{X} \text{ or } E(L), E(X) &: \text{true, theoretical values or the expectation values} \\ L^0, X^0 &: \text{approximated values, not stochastic} \\ \widehat{L}, \widehat{X} &: \text{stochastic, estimated values} \end{aligned}$$

Due to the fact that there is a very high number of sensor nodes in the WSN, this leads to an overdetermined system, e.g for 5000 nodes and 10% beacons results in a system of equations with $u = 2$ unknowns and $m = 500$ measurements. The cost for the solution is very high and will be described later. We will first explain two methods to set up this system of equations.

3.3 Linearizing the Euclidean Distances

In reality, the relationships between the observations r_i and the unknowns x are non-linear and cannot be described with the linear models of the adjustment. Regression functions will be used for non-linear parameters. In order to apply the estimations, the non-linear approaches must be linearized. In our case the system of equations is given by the Euclidean Distances:

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

This system of equations must be linearized with either Taylor series [20] or a linearization tool [21]. The two different methods will be described in the following.

3.4 Taylor Series

Linearizing a non-linear function with Taylor series is established as a standard technique. This method is precise and simple to implement in software. Taylor series assume that every value of a function f at a point $X^0 + x$ can be represented by a series expansion in a Taylor series. Usually, the approximation is precise enough when truncating after the first element. Taylor series work well if the function f_0 is known in X^0 and a sufficient uniform behavior of a curve is given or the improvement of X is small.
Function $f(X)$ holds at the approximated value X^0 the value $f(X^0)$. A better approximation for the searched value is achieved, if the tangent at the approximated value $[X^0, f(X^0)]$ is determined, which is equal to the first derivation. At the point $X^0 + x$ this method is repeated iteratively. If the approximated values X^0 are known, every non-linear function can be represented by a Taylor series.

$$f\left(X^0 + x\right) = f\left(X^0\right) + \left(\frac{\delta f}{\delta X}\right)_{X=X^0} \cdot x + \frac{1}{2}\left(\frac{\delta^2 f}{\delta X^2}\right)_{X=X^0} \cdot x^2 + O^3 \tag{2}$$

Following we describe a second linearization technique.

3.5 Linearization Tool

Although the linearization tool is not as exact as the Taylor series, it requires no mathematical differentiation. It works without a start value, and it is suitable

for a distributed implementation (discussed later in Section 5). Thus, we use the j'th equation of (1) as a linearization tool. By adding and subtracting x_j and y_j to all other equations this leads to:

$$(x - x_j + x_j - x_i)^2 + (y - y_j + y_j - y_i)^2 = r_i^2 \ (i = 1, 2, \ldots, j-1, j+1, \ldots, m). \tag{3}$$

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:

$$(x - x_j)(x_i - x_j) + (y - y_j)(y_i - y_j) = \frac{1}{2}\left[r_j^2 - r_i^2 + d_{ij}^2\right] = b_{ij}. \tag{4}$$

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 $u = 2$ unknowns.

$$\begin{array}{rcl} (x - x_1)(x_2 - x_1) + (y - y_1)(y_2 - y_1) & = & \frac{1}{2}\left[r_1^2 - r_2^2 + d_{21}^2\right] = b_{21} \\ (x - x_1)(x_3 - x_1) + (y - y_1)(y_3 - y_1) & = & \frac{1}{2}\left[r_1^2 - r_3^2 + d_{31}^2\right] = b_{31} \\ \vdots & & \\ (x - x_1)(x_m - x_1) + (y - y_1)(y_m - y_1) & = & \frac{1}{2}\left[r_1^2 - r_m^2 + d_{m1}^2\right] = b_{m1} \end{array} \tag{5}$$

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

$$A\mathbf{x} = \mathbf{b} \tag{6}$$

with:

$$A = \begin{pmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \\ \vdots & \vdots \\ x_m - x_1 & y_m - y_1 \end{pmatrix}, \mathbf{x} = \begin{pmatrix} x - x_1 \\ y - y_1 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} b_{21} \\ b_{31} \\ \vdots \\ b_{m1} \end{pmatrix}. \tag{7}$$

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

3.6 Solving the Linear Least Square Problem

Due to the fact that overdetermined systems of equations with $m >> n$ have not exactly one solution for $A\mathbf{x} = \mathbf{b}$, we have to apply the L2-norm [19]. This is also called the Euclidean Norm, which minimizes the sum of the squares:

$$\underset{x \in \Re^n}{Minimize} \ ||A\mathbf{x} - \mathbf{b}||^2. \tag{8}$$

To summarize, linear systems of equations can be solved iteratively using splitting techniques or directly either with the normal equations or by orthogonal factorization. Existing techniques are numerous but often the differences between them are small. For this reason, we focus our studies on solving the normal equations. For all others we recommend [22].

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:

$$\|A\mathbf{x} - \mathbf{b}\|^2 \rightarrow A^T A\mathbf{x} = A^T\mathbf{b}. \tag{9}$$

$$\mathbf{x} = \left(A^T A\right)^{-1} A^T\mathbf{b}. \tag{10}$$

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, for one unknown point $n = 2$. This reduces the computation and makes it easy to implement it in software. Next, we will study the described methods in detail.

4 Analysis

4.1 Influence of Noisy Distances to the Linearization Method

We described in Section 3.1 that in reality distances are always influenced by measurement errors. Thus, we studied the influence of an increasing distance error to both linearization methods. To realize this, we replaced the exact distances with random values out of a Gaussian distribution $r_{stochastic} \sim \mathcal{N}(\mu_{exact}, \sigma^2)$. We determined the positions of 500 nodes with an increasing number of beacons to analyze the influence of the graduation. For this, we linearized the system of equations either with Taylor series or a linearization tool, solved the linear system of equations with the linear least squares method and determined the error, i.e. the distance between the exact position and the calculated position. We repeated every series 50 times and averaged the results to avoid a strong influence of outliers. All simulations run with a test field of the size 100×100 where all nodes were placed in a uniform distribution.

The non-linear system of equations was linearized in Figure 5 with a linearization tool and in Figure 6 with Taylor series. The relative localization errors in both graphics are strongly influenced by an increasing variance. Here we can see the disadvantage of a linearization tool, because with many beacons the error increases. However, we are interested to equip as few nodes as possible with additional hardware. Thus, there exist a trade off between high precision and number of beacons. Nevertheless, we focus in this paper on the postulation of our algorithm. Summarized, Taylor series is more precise and the precision can be advanced with more observations. In contrast the distribution of Taylor series needs approximate values and the calculation requires more computation cost.

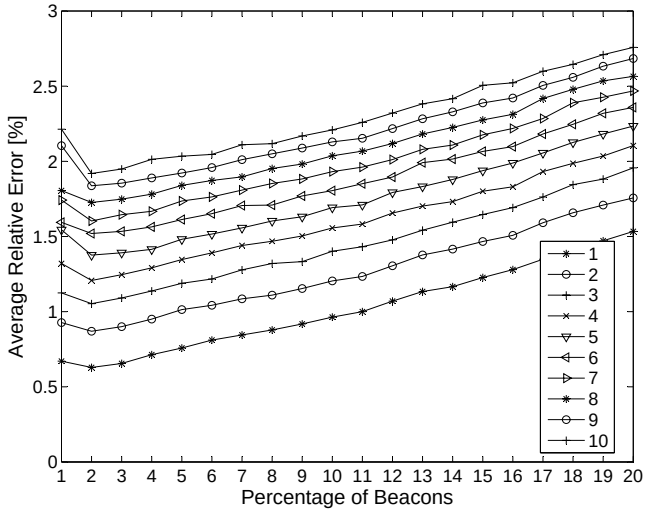

Fig. 5. Localization error over an increasing number of beacons with noisy distances. The non-linear system of equations was linearized with a linearization tool and solved with the linear least squares method. We estimated the positions of 500 sensor nodes in a field with the dimension 100 × 100.

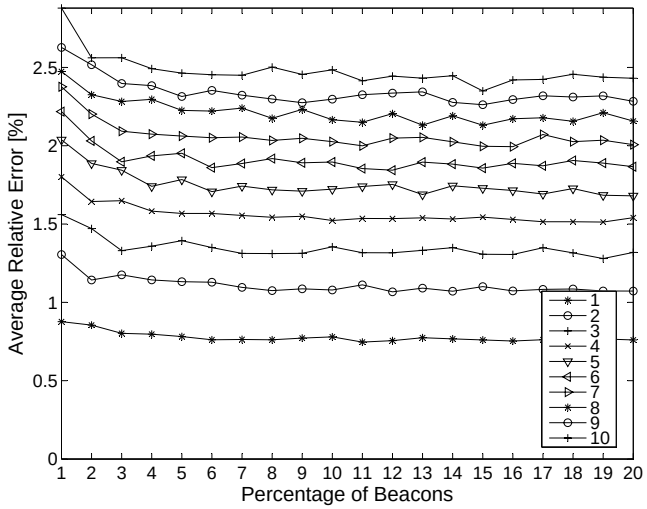

Fig. 6. Localization error with the same simulation settings like in the Figure above but with Taylor linearization.

4.2 Complexity of the Normal Equations

In the following, we will analyze the complexity of the least squares method by solving the normal equations. Although the literature offers numerous specifications, later we will need the details of the calculation to reduce specific parts of it. In order to define the complexity mathematically, we count the number of floating point operations (flops), which is a commonly used method in literature. The required number of computation cycles strongly depends on the used 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: $m = k + 1$.

The linear system of equations:

$$\mathbf{x} = \left(A^T A\right)^{-1} A^T \mathbf{b} \tag{11}$$

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 $A^T A$ by $\frac{n(n+1)}{2}$ flops[4].
2. This $[n \times n]$-matrix $A^T A$ must be inverted[5] to $(A^T A)^{-1}$ with a complexity of n^3.
3. This $[n \times n]$-matrix $(A^T A)^{-1}$ must be multiplied with the $[n \times k]$-matrix A^T, which costs $2n^2k - nk$ flops. This leads to the precalculated Matrix $A_p = (A^T A)^{-1} A^T$.
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. This means, with 100 beacons and the summation of x_1 and y_1 (see (7)) we need 1497 floating point operations. In the next section we will reduce this complexity.

5 Distributing the Least Squares Method

5.1 Topology and Nodes

For optimal self organization wireless sensor networks are structured hierarchically. As already introduced, there are many sensor nodes, some beacons, and

[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 [23]).

a minimum of one base station to which the sensed data is transmitted. Beside resource limited sensor nodes, the beacons and especially the base station are more powerful with more battery capacity. The localization algorithm can be computed centrally on the base station only, shared between the base station and the sensor nodes or completely distributed, being calculated on the sensor nodes only. Since completely central computation results in high network traffic and completely distributed computation exhausts the sensor nodes battery, we decided to develop a hybrid computation of the algorithm. In this hybrid case, the algorithms computation steps are split in parts being distributed over the WSN. There are three methods to distribute the computation load for the localization algorithm:

1. Computation takes place at the base station only.
2. Computation is shared between base station and the sensor nodes.
3. Computation is done at the sensor nodes only.

5.2 Idea and Algorithm Description

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 $\mathbf{b}$ consists of distances between the unknown sensor node and all beacons $r_1..r_m$ and distances $d_{21}..d_{m1}$ between the first beacon and all others. Consequently, we split the complex computation into two parts - a less complex and a very simple part.

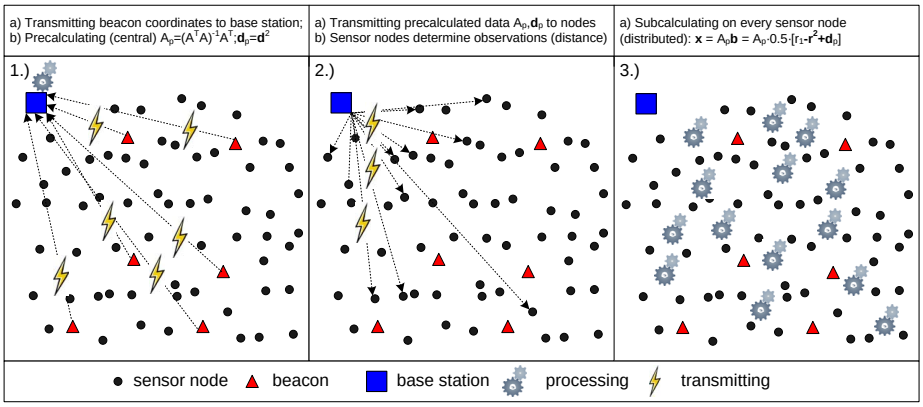

Fig. 7. Schematic of the distributed localization algorithm.

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_p is the same for every sensor node in a static network. This wastes limited resources and produces high redundancy.
Now, with the 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. Before we describe the method in detail, we will postulate the entire algorithm. Figure 7 was created to visualize the following algorithm phases:

1. **Initialization Phase:**
   – All beacons send their position $B(x, y)$ to the base station.
2. **Complex Precalculation Phase (central):**
   – Base station builds matrix A and vector $\mathbf{d}$.
   – Starting the complex precalculation of A_p,$\mathbf{d}_p$.
3. **Communication Phase (distributed):**
   – Base station sends the precalculated matrix A_p and the vector of distances $\mathbf{d}_p$ to all sensor nodes.
4. **Simple Subcalculation Phase (distributed):**
   – Sensor nodes determine the distance to every beacon $r_1..r_m$.
   – Sensor nodes receive the precalculated matrix A_p and vector $\mathbf{d}_p$, built vector $\mathbf{b}$ and estimate their own position P_{est} autonomously.
5. **Communication Phase (distributed)**
   – Sensor nodes send the calculated position back together with the measured signals for quality investigation and outliers tests.

In the next section, we will analyze the performance of the new algorithm regarding complexity, communication and memory effort.

5.3 Performance of the Algorithm

Complexity: In Section 4.2 we already analyzed the complexity of the least squares method. At this point it is important to know what we can save on the sensor nodes without complex precalculations regarding only the remaining subcalculation.

We assume the constellation of the network in Figure 7 with sensor nodes, beacons, and a base station. On the basis of (11), the base station precalculates $A_p = (A^T A)^{-1} A^T$ and $\mathbf{d}_p = \mathbf{d}^2$. The matrix A_p 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, which leads with 100 beacons and the summation of x_1 and y_1 (see (7)) to 791 flops.

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

Here, we focus on a theoretical examination of the algorithm that is, for the moment, independent of protocol definitions and media access operations. Hence, every sensor node must receive the precalculated matrix A_p and vector $\mathbf{d}_p$ with $[n \cdot (m-1) + (m-1)]$ elements. This results in receiving $(3m-3)$ elements, which are 1188 bytes with 100 beacons and floating point representation of every element.[6]
In contrast, in the completely distributed case every beacon would send its coordinates directly to all sensor nodes. Thus, every sensor node has to receive $2(m)$ elements for x and y. In comparison to our algorithm this would result in 800 bytes, which is not much less.

Memory Considerations: 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 $A_p \cdot \frac{1}{2} \cdot (r_1^2 - \mathbf{r}^2 + \mathbf{d}_p)$. In the worst case A_p 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. In floating point representation with $m = 100$ beacons, already 0.796 kb must be allocated for localization only. Normally, the localization task is part of the middleware that has to execute many more tasks. Besides, temporary variables are needed which increase the memory consumption. Given these facts, we studied the critical operations in more detail.

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

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

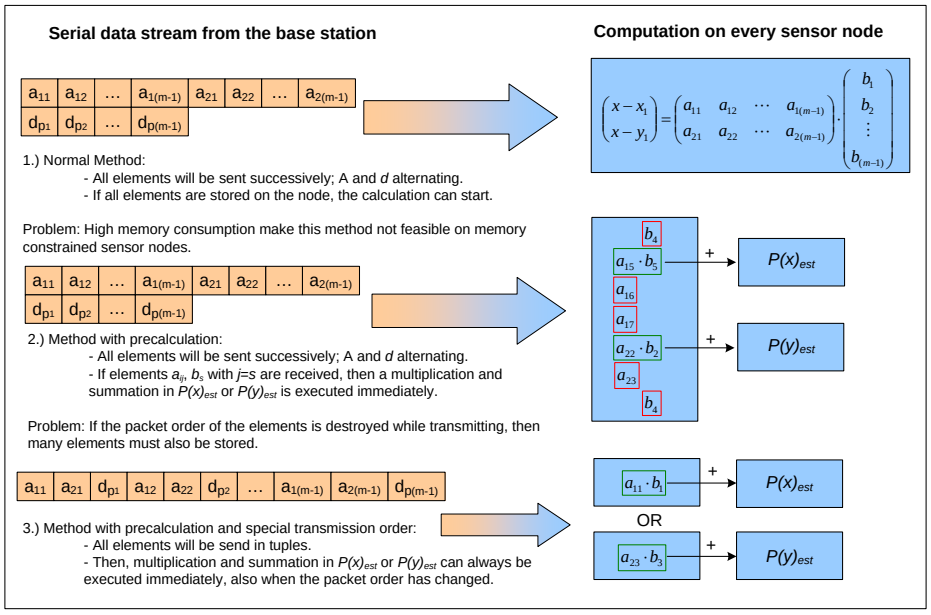

Fig. 8. Three different methods to execute the subcomputation, beginning from the normal method and with optimizations.

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.

With a_{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 $a_{ij} \cdot b_s$ 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 a_{ij}, d_s in appropriate tuples, e.g. $a_{11}, d_1; a_{12}, d_2; \ldots; a_{ij}, d_s$. In the 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. For a clearer understanding, we illustrate all three methods in Figure 8.

By applying the last optimization the algorithm is also memory aware and finally allows implementation on resource constrained sensor nodes.

6 Conclusion

We presented a distributed localization algorithm that allows precise localization on resource limited sensor nodes. Generally, precise localization is achieved by

solving an overdetermined system of equations with the least squares method, which is formed by a multiple trilateration with many beacons and distances. Although the complexity of this solution is relatively high, we showed a method to split the linear least squares method into a complex part, precalculated on the high-performance base station, and a very simple subcalculation on every sensor node. Instead of linearizing the non-linear system of equations with the traditional Taylor series we used a linearization tool that helped to bring the system in the right matrix form. With this approach and based on a network containing 100 beacons we achieved 47.16% savings in computation on every sensor node in comparison to the complete calculation. Moreover, we achieved this with not more communication effort and very little memory consumption with our proposed optimizations. We are currently implementing the algorithm in the sensor network simulation framework J-Sim as well as on our sensor node platform to allow tests under more realistic condistions.

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. 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)
3. 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
4. Gibson, J.: The mobile communications handbook. CRC Press (1996)
5. Alouini, M.: Global Positioning System: An Overview. California Institute of Technology (1996)
6. Stefanidis, A., Nittel, S.: GeoSensor Networks. CRC Press (2004)
7. Bulusu, N.: Gps-less low cost outdoor localization for very small devices. IEEE Personal Communications Magazine **7** (2000) 28–34
8. Delaunay, B.: Sur la sphere vide. In: Bulletin der Akademie der Wissenschaften der UdSSR. (1934) 793–800
9. Mirante, A., Weingarten, N.: The radial sweep algorithm for constructing triangulated irregular networks. In: IEEE Computer Graphics and Applications. (1982) 11–21
10. 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)
11. Savarese, C., Rabaey, J., Langendoen, K.: Robust positioning algorithms for distributed ad-hoc wireless sensor networks. In: USENIX Annual Tech. Conf. (Monterey, CA, June 2002). (2002) 317–327

12. Capkun, S., Hamdi, M., Hubaux, J.P.: Gps-free positioning in mobile ad hoc networks. Cluster Computing **5** (2002) 157–167
13. 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
14. Savvides, A., Han, C.C., Srivastava, M.B.: Dynamic fine grained localization in ad-hoc networks of sensors. In: 7th ACM MobiCom (Rome, Italy,2001). (2001) 166–179
15. 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 (ICDCS'05). (2005) 643–652
16. 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 (WWAN) ICDCSW'05). (2005) 892–898
17. 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 (MobiQuitous'04). (2004) 24–30
18. 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
19. Wolf, P.R., Ghilani, C.D.: Adjustment Computations: Statistics and Least Squares in Surveying and GIS (Surveying & Boundary Control). Wiley-Interscience (1997)
20. Farebrother, R.: Fitting Linear Relationships. Springer (1998)
21. Murphy, W.S., Hereman, W.: Determination of a position in three dimensions using trilateration and approximate distances. (1999)
22. Lawson, C.L., Hanson, R.: Solving Least Squares Problems. Englewood Cliffs, NJ: Prentice-Hall (1974)
23. Golub, G.H., Loan, C.F.V.: Matrix Computations. The Johns Hopkins University Press (1996)