Sensors **2012**, *12*, 839-862; doi:10.3390/s120100839

OPEN ACCESS

sensors

ISSN 1424-8220

www.mdpi.com/journal/sensors

Article

A Low-Complexity Geometric Bilateration Method for Localization in Wireless Sensor Networks and Its Comparison with Least-Squares Methods

Juan Cota-Ruiz [1,⋆], Jose-Gerardo Rosiles [2], Ernesto Sifuentes [1] and Pablo Rivas-Perea [3]

[1] Department of Electrical and Computer Engineering, Autonomous University of Ciudad Juárez (UACJ), Ave. del Charro # 450 Nte. C.P.32310, Ciudad Juárez, Chihuahua, México; E-Mail: esifuent@uacj.mx

[2] Rosiles Consulting, El Paso, TX 79938, USA; E-Mail: rosiles@ieee.org

[3] Department of Computer Science, Baylor University, One Bear Place #97356, Waco, TX 76798, USA; E-Mail: Pablo_Rivas_Perea@baylor.edu

⋆ Author to whom correspondence should be addressed; E-Mail: jcota@uacj.mx; Tel.: +52-656-688-4841.

Received: 12 December 2011; in revised form: 9 January 2012 / Accepted: 10 January 2012 / Published: 12 January 2012

Abstract: This research presents a distributed and formula-based bilateration algorithm that can be used to provide initial set of locations. In this scheme each node uses distance estimates to anchors to solve a set of circle-circle intersection (CCI) problems, solved through a purely geometric formulation. The resulting CCIs are processed to pick those that cluster together and then take the average to produce an initial node location. The algorithm is compared in terms of accuracy and computational complexity with a Least-Squares localization algorithm, based on the Levenberg–Marquardt methodology. Results in accuracy *vs.* computational performance show that the bilateration algorithm is competitive compared with well known optimized localization algorithms.

Keywords: distributed-localization; wireless sensor networks; Least Squares (LS); optimization; bilateration

1. Introduction

Recent advances in microelectronics have led to the development of autonomous tiny devices called sensor nodes. Such devices, in spite of their physical limitations, contain the essential components of a computer, such as memory, I/O ports, sensors, and wireless transceivers which are typically battery-powered. Once deployed (randomly or not) over a certain area, sensor nodes have the ability to be in touch via wireless communications with neighboring nodes forming a wireless sensor network (WSN). The great advantage of using WSNs is that they can be applied in important areas such as disaster and relief, military affairs, medical care, environmental monitoring, target tracking, and so on [1–3]. However, most of WSN applications are based on local events. This means that each sensor node needs to detect and share local phenomenons with neighboring nodes, implying that the location of such events (*i.e.*, sensor locations) are crucial for the WSN application. In this way, sensor self-positioning represents the first startup process in most WSN projects. It is well known that using a GPS in each sensor node represents the primary solution to infer position estimates. However, this option is not suitable to be considered in all nodes if parameters like size, price, and energy-consumption in a sensor node are of concern [4]. In order to optimize such parameters, a good option consists of reducing to a small fraction of sensors with GPS, and the remainder sensors (*i.e.*, unknown sensors), commonly above 90% of total deployed sensors, should use alternatives to estimate its own positions like radio-frequency transmissions or connectivity with neighboring sensors [5–7].

In order to provide position estimates many localization algorithms have been proposed, coming from different perspectives as described in [8,9]. Basically, localization algorithms can be categorized according to range-based *vs.* range-free methods, anchor-based *vs.* anchor-free models, and distributed *vs.* centralized processing [10,11]. Range-based methods consist of estimating node locations (using a localization algorithm) based on ranging information among sensor nodes. Range estimation between pairs of nodes is achieved using techniques of Time-of-Arrival (ToA), Receive-Signal-Strength (RSS), or Angle-of-Arrival (AoA) [12]. This approach has the disadvantage of requiring extra-hardware in each sensor board, increasing the cost per sensor. However, as far as is known, this approach provides the best cost-accuracy performance in localization algorithms. A less expensive but more inaccurate alternative consists of using just connectivity among sensor nodes to estimate node locations, called range-free [13]. On the other hand, if position estimates are obtained by considering absolute references (e.g., sensors with GPS or Anchors), the resulted position estimates (also with absolute positions) will be closely related to such reference positions, called an anchor-based model. By the contrary, if no reference positions are used to estimate locations, relative coordinates will be obtained, called an anchor-free model.

One of the most interesting and relevant aspects in WSN localization is associated with the way to compute the location of sensor nodes. For example, if all pairwise distance measurements among sensor nodes are sent to a central node to compute position estimates, the localization algorithm becomes centralized. This kind of central processing has the advantage of global mapping, but it has basically two important disadvantages which demerit its use in many cases when robustness and saving-energy have high priority in a WSN [14]. Some important centralized schemes are the next. In [15] an iterative descent procedure (*i.e.*, Gauss–Newton method) is used in a centralized way to solve the Non-Linear

Least-Square (NLLS) problem. Another interesting centralized scheme was proposed in [16] where the WSN localization problem is modeled as linear or semidefinite program (SDP), and a convex optimization is used to solve problem.

In contrast, when each sensor node estimates its own location using available information of neighboring nodes (e.g., range, connectivity, location, *etc.*), the localization process becomes distributed. Distributed processing is much less energy consuming in WSNs than centralized processing because centralized schemes need to collect relevant information from all nodes in the network which implies re-transmissions in multi-hop environments. Also, distributed algorithms are tolerant to node failures due to node redundancy. Thus, basically a distributed algorithm allows robustness, saving-energy, and scalability [14,17,18], which overcomes the limitations imposed by the centralized approach. In [19], a robust least squares scheme (RLS) for multi-hop node localization is proposed. This approach reduces the effects of error propagation by introducing a regularization parameter in the covariance matrix. However, the computational cost to mitigate the adverse effects of error propagation is too high at energy-constrained nodes. Similarly, [20] proposes two weighted least squares techniques to gain robustness when a non-optimal propagation model is considered however they failed to introduce a covariance matrix in the localization process that can effectively decrease the computational complexity. On the other hand, the authors of [21] propose a Quality of Trilateration method (QoT) for node localization. This approach provides a quantitative evaluation of different geometric forms of trilaterations. However, it seems to be that the main idea of this methodology depends on the quality or resolution of geometric forms (*i.e.*, like image processing) which is impractical to be implemented in resource-constrained devices with limited memory and processing capabilities (*i.e.*, nodes).

In this paper, we analyze a range-based bilateration algorithm (BL) that can be used in a distributed way to provide initial estimates for unknown sensors in a wireless sensor network (our analysis consider that each unknown sensor can determine its initial position communicating directly with several anchors). In this case, each node uses a set of two anchors and their respective ranges at a time to solve a set of circle intersection problems using a geometric formulation. The solutions from these geometric problems are processed to pick those that cluster around the location estimate and then take the average to produce an initial node location. Finally, we present a computational/accuracy comparison between the BL algorithm, based on closed-formulas, and a classical Least Squares (LS) approach for localization, based on the iterative Levenberg–Marquardt algorithm (LM).

The outline of this paper is as follows. In Section 2 we examine a popular ranging technique for WSNs used in our simulations. In Section 3 we explore the localization problem from the Least Squares point of view. In Section 4 we analyze in detail the BL algorithm. In Sections 5 and 6 we evaluate the accuracy and computational-complexity performance respectively between the bilateration algorithm *vs.* LS schemes. Finally, we present our conclusions.

2. Ranging Techniques

This section presents a brief overview of an existing ranging technique used to estimate the true distance between two sensor nodes using power measurements, called Received Signal Strength (RSS). This technique is popular because sensor nodes do not require special hardware support to estimate distances. As a first approximation, considering the free space path loss model, the distance d_{ij} between

two sensors s_i and s_j can be estimated by assuming that the power signal decreases in a way that is inversely proportional to the square of the distance $(1/d_{ij}^2)$. However, in real environments the signal power is attenuated by a factor $d^{-\eta_p}$. The path-loss factor η_p is closely related to geometrical and environmental factors, and it varies from 2 to 4 for practical situations [22]. In noiseless environments the power signal traveling from a sensor s_j to a sensor s_i can be measured according to the relation [23]

$$P_{ij} = P_0 \left(\frac{d_0}{d_{ij}} \right)^{\eta_p} \tag{1}$$

where the path-loss factor (η_p) depends directly on the environmental conditions. P_0 is the received power at the short reference distance of $d_0 = 1m$ from the transmitter. Also, P_0 can be computed by the Friis free space equation [24]. The log-distance path loss model

$$\overline{P}_L^{(ij)}(dB) = P_0 - 10\eta_p \log \frac{d_{ij}}{d_0} \tag{2}$$

measures the average large-scale path loss between sensors s_i and s_j. The actual path-loss (in dB) is a normally distributed random variable:

$$P_L^{(ij)} \sim \mathcal{N}(\overline{P}_L^{(ij)}, \sigma_{SH}^2) \tag{3}$$

where σ_{SH} is given in dB and reflects the degradations on signal propagation due to reflection, refraction, diffraction, and shadowing. It can be seen that the linear measurements and distance estimates have a log-normal distribution with a multiplicative effect on the measurements. The noisy range measurement R_{ij} can be obtained from Equations (2) and (3) as

$$R_{ij} = 10^{\frac{P_0 - P_L^{(ij)}}{10 \cdot \eta_p}} \tag{4}$$

3. Least-Squares Multilateration Localization Algorithms

In this section, we describe multilateration schemes that provide solutions to the Least-Square (LS) problem for location estimates using noisy ranging information derived from ToA or RSS ranging techniques. Consider a set of N wireless sensor nodes $\mathbf{S} = \{s_1, s_2, \ldots, s_N\}$, randomly distributed over a 2-D region whose locations are unknown. We represent these unknown locations with vectors $\mathbf{z}_i = [x_i, y_i]^T$. Further, we assume the presence of a set $\mathbf{A} = \{a_1, a_2, \ldots, a_M\}$ of M reference or anchor nodes with known position $\mathbf{q}_j = \left[x_j, y_j\right]^T$. Anchor nodes, a_i, are equipped with GPS or a similar scheme to self localize. Also, for practical situations $M \ll N$ with $M > 2$. We develop our discussion assuming a 2-D scenario, but it can be easily generalized to the 3-D case.

Moreover, we assume that any sensor can estimate pairwise ranges with its neighbors using time-of-arrival (ToA) or radio signal strength (RSS) techniques [24]. Denote the range estimate between the node s_i and anchor a_j as

$$R_{ij} = d_{ij} + e_{ij} \tag{5}$$

where d_{ij} is the true distance between a_j and s_i, and e_{ij} represents the measurement error introduced by environmental noise, propagation distortion, and the ranging technique. Then the solution to the localization problem for a node s_i consists of minimizing the sum of certain weighted error-distance function $e_w(\cdot)$ as follows:

$$\mathbf{p}_i = \arg\min_{\mathbf{x}} \sum_{j=1}^{M} e_w \left(\left\| \mathbf{q}_j - \mathbf{x} \right\| - R_{ij} \right) = \arg\min_{\mathbf{x}} \mathcal{F}(\mathbf{x}) \tag{6}$$

where $\mathbf{p}_i = [x_i, y_i]^T$ represents the most likely position for the sensor s_i that minimizes $\mathcal{F}$, $\|\cdot\|$ represents the Euclidean norm, and $e_w(x)$ represents a function that provides a specific weight to the argument x (*i.e.*, error distance). For example, the function $e_2(x) = (x)^2$, the LS formulation, is commonly used to solve Equation (6) due its tractability and efficiency in both mathematical and computational analysis. The LS problem can be solved either by closed-form solutions or by iterative methods. Next we describe both methodologies in detail.

3.1. Closed-Form LS Multilateration

Closed-Form methods have the advantage of fast time processing, which is useful for constrained devices (*i.e.*, motes) where the energy conservation represents one of the major concerns. However, this approach is also subject to inaccurate estimates due to noisy ranging measurements, so in most cases this approach is not a suitable option in real WSN scenarios where current ranging techniques are not able to provide the required accuracy on the ranging measurements. For example, Spherical Intersection (SX), Spherical Interpolation (SI), and Global Spherical Least Squares (GSLS) [25] can solve a nonlinear set of equations using closed-formulas. These approaches provide good accuracy in the estimated positions under conditions like small biases and small standard deviations, but they also provide meaningless estimates under noisy environments [26]. A more robust closed-form scheme consists of using the classical LS multilateration discussed next [19,27,28].

Consider that a sensor s_i with Cartesian position $\mathbf{p}_i = [x_i, y_i]^T$ has already estimated its range R_{ij} to M anchors. For each anchor a_j with position $\mathbf{q}_j = [x_j, y_j]^T$, an equation $\left\| \mathbf{q}_j - \mathbf{p}_i \right\|^2 = R_{ij}^2$ is generated as shown the next formulas:

$$\begin{array}{ccc} \|\mathbf{q}_1 - \mathbf{p}_i\|^2 = R_{i1}^2 & & (x_1 - x_i)^2 + (y_1 - y_i)^2 = R_{i1}^2 \\ \|\mathbf{q}_2 - \mathbf{p}_i\|^2 = R_{i2}^2 & \Leftrightarrow & (x_2 - x_i)^2 + (y_2 - y_i)^2 = R_{i2}^2 \\ \vdots & & \vdots \\ \|\mathbf{q}_M - \mathbf{p}_i\|^2 = R_{iM}^2 & & (x_M - x_i)^2 + (y_M - y_i)^2 = R_{iM}^2 \end{array} \tag{7}$$

The system of Equations (7) can be linearized by subtracting the first equation $(j = 1)$ from the last $M - 1$ equations arriving to a linear system that can be represented in a matrix form as

$$\mathbf{A}\mathbf{p}_i = \mathbf{b} \tag{8}$$

where

$$\mathbf{A} = -2 \begin{bmatrix} x_2 - x_1 & y_2 - y_1 \\ x_3 - x_1 & y_3 - y_1 \\ \vdots & \vdots \\ x_M - x_1 & y_M - y_1 \end{bmatrix}_{(M-1)\times 2} \tag{9}$$

$$\mathbf{b} = \begin{bmatrix} R_{i2}^2 - R_{i1}^2 + x_1^2 + y_1^2 - x_2^2 - y_2^2 \\ R_{i3}^2 - R_{i1}^2 + x_1^2 + y_1^2 - x_3^2 - y_3^2 \\ \vdots \\ R_{iM}^2 - R_{i1}^2 + x_1^2 + y_1^2 - x_M^2 - y_M^2 \end{bmatrix}_{(M-1)\times 1} \tag{10}$$

Now the least squares solution to Equation (8) is to determine an estimate for $\mathbf{p}_i$ that minimizes

$$\begin{aligned} f(\mathbf{p}_i) &= \min_{\mathbf{p}_i} \left\{ \frac{1}{2} \|\mathbf{A}\mathbf{p}_i - \mathbf{b}\|^2 \right\} \\ &= \min_{\mathbf{p}_i} \left\{ \frac{1}{2} (\mathbf{A}\mathbf{p}_i - \mathbf{b})^T (\mathbf{A}\mathbf{p}_i - \mathbf{b}) \right\} \end{aligned} \tag{11}$$

After some manipulations we obtain the following:

$$f(\mathbf{p}_i) = \min_{\mathbf{p}_i} \left\{ \frac{1}{2} \mathbf{p}_i^T \mathbf{A}^T \mathbf{A} \mathbf{p}_i - \mathbf{p}_i^T \mathbf{A}^T \mathbf{b} + \frac{1}{2} \mathbf{b}^T \mathbf{b} \right\} \tag{12}$$

and the gradient of f at $\mathbf{p}_i$ is

$$\bigtriangledown f(\mathbf{p}_i) = \mathbf{A}^T \mathbf{A} \mathbf{p}_i - \mathbf{A}^T \mathbf{b} = 0 \tag{13}$$

which provides the estimate (*i.e.*, normal equations) to Equation (8):

$$\hat{\mathbf{p}}_i = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b} \tag{14}$$

Solving for Equation (14) may not work properly if $\mathbf{A}^T\mathbf{A}$ is close singular, so a recommended approach is to use a Tikhonov regularization as follows:

For $\mu > 0$ (e.g., close to zero)

$$\begin{aligned} f_\mu(\mathbf{p}_i) &= \min_{\mathbf{p}_i} \left\{ \frac{1}{2} \|\mathbf{A}\mathbf{p}_i - \mathbf{b}\|^2 + \frac{\mu}{2} \|\mathbf{p}_i\| \right\} \\ &= \min_{\mathbf{p}_i} \left\{ \frac{1}{2} \mathbf{p}_i^T \mathbf{A}^T \mathbf{A} \mathbf{p}_i - \mathbf{p}_i^T \mathbf{A}^T \mathbf{b} + \frac{1}{2} \mathbf{b}^T \mathbf{b} + \frac{\mu}{2} p_i^T \mathbf{p}_i \right\} \end{aligned} \tag{15}$$

Then the gradient of f_μ at $\mathbf{p}_i$ is

$$\bigtriangledown f_\mu(\mathbf{p}_i) = \mathbf{A}^T \mathbf{A} \mathbf{p}_i - \mathbf{A}^T \mathbf{b} + \mu \mathbf{p}_i = 0 \tag{16}$$

Factorizing we arrive to a robust estimate for the LS problem where the idea is to modify eigenvalues to avoid working with zero eigenvalues [19,29].

$$\hat{\mathbf{p}}_i = (\mathbf{A}^T\mathbf{A} + \mu\mathbf{I})^{-1}\mathbf{A}^T\mathbf{b} \tag{17}$$

3.2. Iterative LS Algorithms

Iterative methods are usually employed either when large-data set of information need to be processed or when an exact solution to a certain problem is not feasible (e.g., non-linear systems of equations) [30]. Optimization techniques represent a good alternative to solve such non-linear equations using an iterative procedure. Optimization algorithms that solve Non-Linear Least-Square (NLLS) problems (*i.e.*, the WSN localization problem) have been extensively proposed where the Newton or Quasi-Newton methods are iteratively used to minimizing some residuals [15,31,32]. The next paragraphs describe two well known iterative algorithms that are used to solve the NLLS problem: the Levenberg–Marquardt (LM) and the Trust-Region-Reflective (TRR).

Assuming that a node denoted $\mathbf{s}_i$, with Cartesian position $\mathbf{p}_i = [x_i, y_i]^T$, estimates its distance R_{ij} to M anchors denoted a_j, with positions $\mathbf{q}_j = [x_j, y_j]^T$, with $j = 1, \ldots, M$. Consider the following residual error vector:

$$\mathbf{R}(\mathbf{p}_i) = \begin{bmatrix} R_{i1} - \|\mathbf{p}_i - \mathbf{q}_1\| \\ R_{i2} - \|\mathbf{p}_i - \mathbf{q}_2\| \\ \vdots \\ R_{iM} - \|\mathbf{p}_i - \mathbf{q}_M\| \end{bmatrix} \tag{18}$$

Therefore, to find the more likely position of $\mathbf{p}_i$, the program

$$\min_{\mathbf{p}_i} f(\mathbf{p}_i) = \min_{\mathbf{p}_i} \left(\frac{1}{2}\mathbf{R}(\mathbf{p_i})^T\mathbf{R}(\mathbf{p_i}) \right) \tag{19}$$

is solved, which is the least squares problem.

To solve Equation (19) we employ the TRR algorithm and the LM algorithm. The TRR algorithm uses a sub-space trust-region method to minimize a function $f(x)$. Here, approximations to f inside of a trust-region are iteratively required. The three main concerns in this algorithm are how to choose and compute the approximation to the function, how to choose and modify the trust region, and, finally, how to minimize over the sub-space trust-region. Even though the TRR algorithm provides an accurate solution for the WSN initial estimates, it is expensive (computationally speaking) for constrained sensor nodes [33].

On the other hand, the LM algorithm uses the search direction approach (a mix between the Gauss–Newton direction and the steepest descent direction) to find the solution to Equation (19). This algorithm outperforms the simple gradient descent methodology [34], and also it avoids dangerous operations with singular matrices as the pure Newton method does, so this methodology represents a good algorithm for comparison with the bilateration approach due to its robustness, speed, and accuracy [35]. Following the procedure presented in [29], Equation (19) can be solved by the Line Search Levenberg–Marquardt methodology shown in Algorithm 1, where $\|\cdot\|$ is the ℓ-2 norm, $\mathbf{I}$ is the

identity matrix, R_{ij} is the estimated distance between the mote s_i and the anchor a_j, $\mathbf{J}(\mathbf{p}_k)$ represents the Jacobian of $\mathbf{R}(\mathbf{p}_k)$ at the iteration k, and $\mathbf{M}_f(\mathbf{p}_k)$ is the merit function given by

$$\frac{1}{2}\mathbf{R}^T(\mathbf{p}_k)\mathbf{R}(\mathbf{p}_k) \tag{20}$$

The derivative of the merit function at the iteration k is

$$\mathbf{M}_f^{'}(\mathbf{p}_k) = \mathbf{J}^T(\mathbf{p}_k)\mathbf{R}(\mathbf{p}_k) \tag{21}$$

Δ_{LM} is the Levenberg–Marquardt direction,

$$\mu_k = \rho \left\|\mathbf{J}^T(\mathbf{p}_k)\mathbf{R}(\mathbf{p}_k)\right\| \tag{22}$$

where $\rho \in (0,1)$, and finally

$$\mathbf{p}_0 = \frac{1}{M}\sum_{j=1}^{M}\mathbf{q}_j \tag{23}$$

provides the initial guess required for the TRR and LM iterative algorithms.

Algorithm 1 Levenberg–Marquardt methodology.

Require: an initial position $\mathbf{p}_0$
Ensure: a solution $\mathbf{p}_{k+1}$
1: **Initialize:** k = 0,τ = Threshold, $\rho = 0.05$
2: **do**
3: Solve: $(\mathbf{J}^T(\mathbf{p}_k)\mathbf{J}(\mathbf{p}_k) + \mu_k\mathbf{I})\Delta_{LM} = -\mathbf{J}^T(\mathbf{p}_k)\mathbf{R}(\mathbf{p}_k)$
4: Find the sufficient decrease (Armijo's condition):
5: such that $\alpha_k = (\frac{1}{2})^t for\ t = 0,1,\ldots$
6: satisfies $\mathbf{M}_f(\mathbf{p}_k + \alpha_k\Delta_{LM}) \leq \mathbf{M}_f(\mathbf{p}_k) + 10^{-4}\alpha_k\mathbf{M}_f^{'}(\mathbf{p})^T\Delta_{LM}$
7: Update position: $\mathbf{p}_{k+1} = \mathbf{p} + \alpha_k\Delta_{LM}$
8: Update μ_k: $\mu_k = \rho\|\mathbf{J}^T(\mathbf{p}_k)\mathbf{R}(\mathbf{p}_k)\|$
9: **while**$(\|\mathbf{p}_{k+1} - \mathbf{p}_k\| \leq \tau$ or $k \leq 100)$

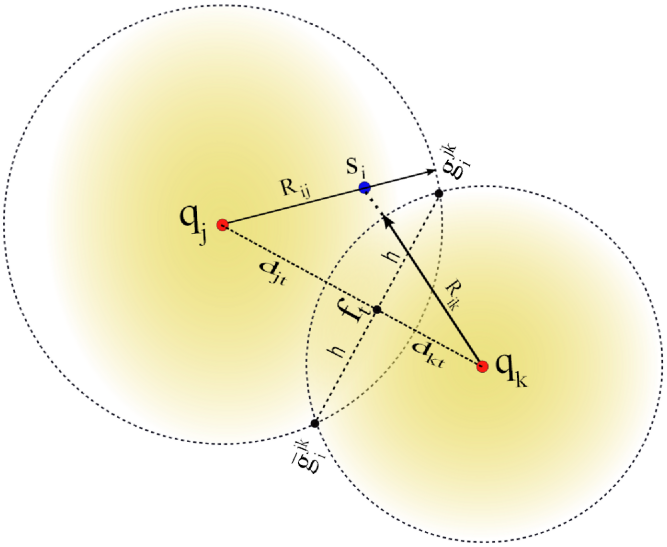
4. A Bilateration Localization Method

In this section we present the bilateration method for WSN localization which can be used as the initialization step for iterative localization schemes. This algorithm avoids iterative procedures, gradient calculations, and matrix operations that increase the internal processing in a constrained device. This research was done independently of the work presented in [36]. Even though both schemes share the same idea (*i.e.*, bilateration), the procedure and the scope of both works are different as shown in Subsection 4.1. We show that it is possible to obtain a position estimate by solving a set of bilateration problems between a sensor node and its neighboring anchors, and then fusing the solutions according to the geometrical relationships among the nodes. Our aim is to find a scheme that can be deployed on a computationally constrained node. We argue that bilateration is an attractive option as the localization problem is divided on smaller sub-problems which can be efficiently solved on a mote. Next we

start our development by introducing the typical assumptions and definitions considered in a WSN localization problem.

Let us define anchor subsets $\mathbf{A}_{jk} \subset \mathbf{A}$ such that $\mathbf{A}_{jk} = \{a_j, a_k\}$ with $j \neq k$. Hence, there is a total of $Q = \binom{M}{2}$ anchor subsets. Without loss of generality, consider the case for one node s_i that receives from a subset A_{jk} the anchor positions $\mathbf{q}_j$ and $\mathbf{q}_k$, and computes the respective ranges R_{ij} and R_{ik} using RSS or ToA measurements. A possible geometrical scenario for this configuration is shown in Figure 1. We can appreciate from this example that the range estimates R_{ij} is larger than d_{ij} and R_{ik} is shorter than d_{ik}. Now, consider the two range circles shown in the figure; one with its origin at $\mathbf{q}_j$ and radius R_{ij}, and the second with center in $\mathbf{q}_k$ and radius R_{ik}. Next, define the two circle intersection points as $\mathbf{g}_i^{jk}$ and $\overline{\mathbf{g}}_i^{jk}$, where $\overline{\mathbf{g}}_i^{jk}$ is the reflection of $\mathbf{g}_i^{jk}$ with respect to the (imaginary) line that connects $\mathbf{q}_j$ and $\mathbf{q}_k$. In this case, the superscript jk represents the anchor subset A_{jk}. To simplify our discussion, we drop the superscripts, and only use them when more than one anchor subset is involved in our discussion.

Figure 1. Sensor s_i finding its two feasible solutions $(\mathbf{g}_i, \overline{\mathbf{g}}_i)$ based on the anchors locations $(\mathbf{q}_k, \mathbf{q}_j)$ and their respective anchor range measurements (R_{ij}, R_{ik}).



In our approach, node s_i determines the circle-circle intersections (*CCI*) $\mathbf{g}_i$ and $\overline{\mathbf{g}}_i$ by solving the closed-form expression reported in [37]. For instance, consider the two right triangles formed by the coordinates $(\mathbf{q}_j, \mathbf{g}_i, \mathbf{f}_t)$ and $(\mathbf{q}_k, \mathbf{g}_i, \mathbf{f}_t)$ in Figure 1, which satisfy the following relationships:

$$d_{jt}^2 + h^2 = R_{ij}^2 \tag{24a}$$

$$d_{kt}^2 + h^2 = R_{ik}^2 \tag{24b}$$

respectively. The distance d_{jt} can be obtained by solving for h^2 in Equations (24a) and (24b):

$$R_{ij}^2 - d_{jt}^2 = R_{ik}^2 - d_{kt}^2 \tag{25}$$

and letting $d = \|\mathbf{q}_j - \mathbf{q}_k\| = d_{jt} + d_{kt}$ resulting in

$$R_{ij}^2 - d_{jt}^2 = R_{ik}^2 - (d - d_{jt})^2 \tag{26a}$$

$$R_{ij}^2 = R_{ik}^2 - d^2 + 2 \cdot d \cdot d_{jt} \tag{26b}$$

$$d_{jt} = \frac{R_{ij}^2 - R_{ik}^2 + d^2}{2 \cdot d} \tag{26c}$$

where the position $\mathbf{f}_t = [x_t, y_t]^T$ is obtained as follows:

$$\mathbf{f}_t = \mathbf{q}_j + \frac{d_{jt}}{d}\left(\mathbf{q}_k - \mathbf{q}_j\right) \tag{27}$$

Finally, the circle intersection $\mathbf{g}_i = [x_i, y_i]^T$ is computed as

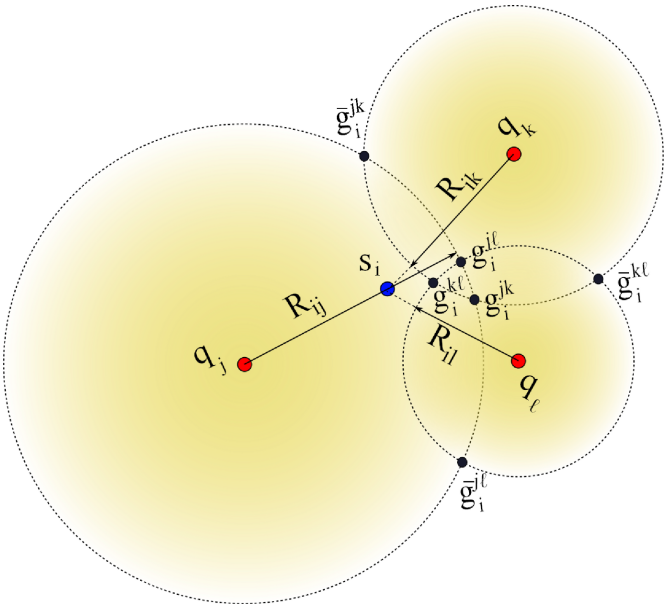
$$x_i = x_t \pm \frac{h}{d}(y_k - y_j) \tag{28a}$$

$$y_i = y_t \mp \frac{h}{d}(x_k - x_j) \tag{28b}$$

where $\mathbf{q}_j = [x_j, y_j]^T, \mathbf{q}_k = [x_k, y_k]^T$, and h is easily obtained from Equation (24). The complementary signs of Equations (28a) and (28b) are used to obtain the solution for $\bar{\mathbf{g}}_i$.

Each node s_i applies the *CCI* procedure using all Q subsets $\mathbf{A}_{jk}$. For instance, $\mathbf{g}_i^{jk}$ and $\bar{\mathbf{g}}_i^{jk}$ are obtained from the subset $\mathbf{A}_{jk}$, $\mathbf{g}_i^{j\ell}$ and $\bar{\mathbf{g}}_i^{j\ell}$ are obtained from the subset $\mathbf{A}_{j\ell}$, and so on. Hence, a sensor node will have $2Q$ possible initial position estimates where half are considered mirror solutions which should be eliminated through the selection process described next. Geometrically, we expect that the true location will be located around the region where solutions form a cluster (*i.e.*, half of the circle intersections should ideally intersect at the solution). Let us to consider the example shown in Figure 2.

Figure 2. Sensor s_i getting its initial estimation P_i^0 from three non-collinear anchors (a_j, a_k, a_ℓ).



There are three anchors named a_j, a_k and a_ℓ and a node s_i that needs to be localized. The range estimate R_{ij} is larger than d_{ij}, the range estimate R_{ik} is shorter than d_{ik}, and the range estimate $R_{i\ell}$ is shorter than $d_{i\ell}$. Hence, s_i computes a set of of six location candidates given by $\{\mathbf{g}_i^{jk}, \bar{\mathbf{g}}_i^{jk}, \mathbf{g}_i^{j\ell}, \bar{\mathbf{g}}_i^{j\ell}, \mathbf{g}_i^{k\ell}, \bar{\mathbf{g}}_i^{k\ell}\}$.

As seen in the figure, all the mirror circle intersection estimates will tend to be isolated while the correct circle intersections will tend to cluster around the node location. For example, to decide between $\mathbf{g}_i^{jk}$ and $\overline{\mathbf{g}}_i^{jk}$ candidate positions, generated using the anchors (a_j, a_k), the sensor s_i obtains the minimum Square Euclidean sum from the location $\mathbf{g}_i^{jk}$ to each pair of candidate positions as follows:

$$\psi = \min\left(\left\|\mathbf{g}_i^{jk} - \mathbf{g}_i^{j\ell}\right\|^2, \left\|\mathbf{g}_i^{jk} - \overline{\mathbf{g}}_i^{j\ell}\right\|^2\right) + \min\left(\left\|\mathbf{g}_i^{jk} - \mathbf{g}_i^{k\ell}\right\|^2, \left\|\mathbf{g}_i^{jk} - \overline{\mathbf{g}}_i^{k\ell}\right\|^2\right) \tag{29}$$

On the other hand, the sensor s_i also obtains the minimum Square Euclidean sum from the location $\overline{\mathbf{g}}_i^{jk}$ to each pair of candidate positions as follows:

$$\varphi = \min\left(\left\|\overline{\mathbf{g}}_i^{jk} - \mathbf{g}_i^{j\ell}\right\|^2, \left\|\overline{\mathbf{g}}_i^{jk} - \overline{\mathbf{g}}_i^{j\ell}\right\|^2\right) + \min\left(\left\|\overline{\mathbf{g}}_i^{jk} - \mathbf{g}_i^{k\ell}\right\|^2, \left\|\overline{\mathbf{g}}_i^{jk} - \overline{\mathbf{g}}_i^{k\ell}\right\|^2\right) \tag{30}$$

Finally, the lowest value of ψ and φ helps to decide between choosing $\mathbf{g}_i^{jk}$ or $\overline{\mathbf{g}}_i^{jk}$. The process is repeated for all Q solution pairs to generate a set of disambiguated locations.

Algorithm 2 General code used by every sensor s_i to get its initial position estimate $\mathbf{p}_i^0$. [1]

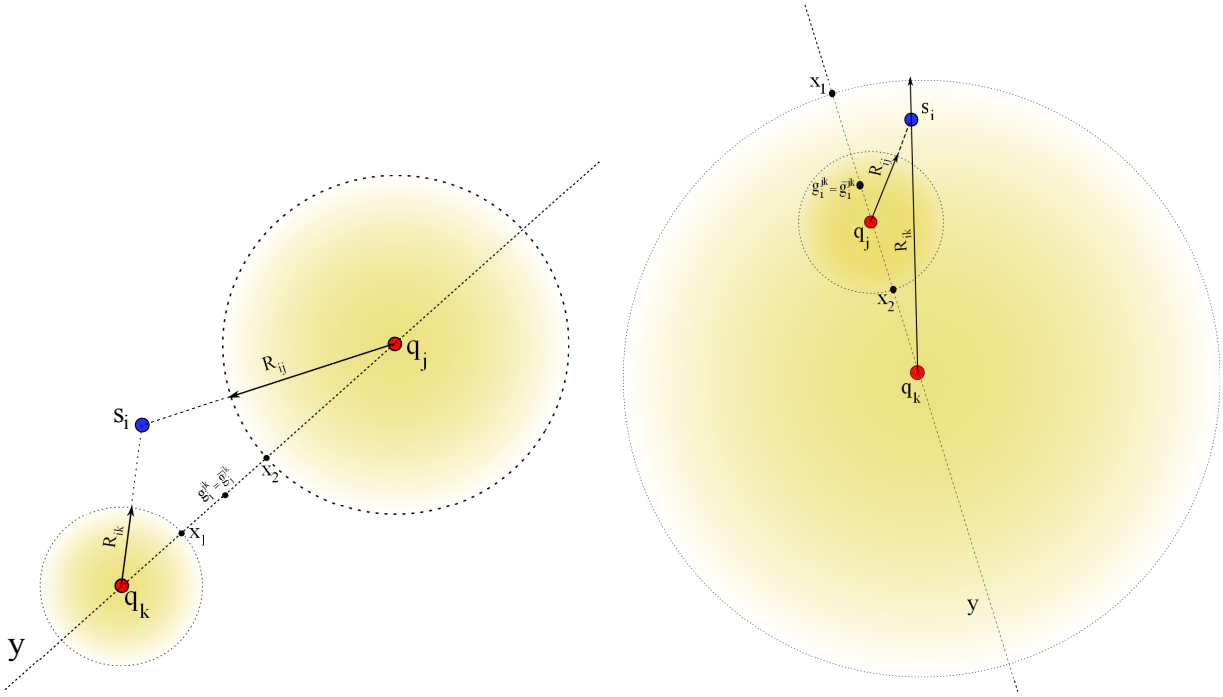
Require: $\mathbf{q}_k, R_{ik}, with\,\{k \leftarrow 1,\ldots,M\}$, and $Q \leftarrow \binom{M}{2}$
Ensure: $\mathbf{p}_i^0$
Initialize: $\mathbf{T} \leftarrow [0,0]^T$
for each subset $\mathbf{A}_{jk} \in \binom{M}{2}$ two-anchor subsets **do**
  $\psi \leftarrow 0$
  $\varphi \leftarrow 0$
  $\left(\mathbf{g}_i^{jk}, \overline{\mathbf{g}}_i^{jk}\right) \leftarrow CCI\left(\mathbf{q}_j, \mathbf{q}_k, R_{ij}, R_{ik}\right)$ {Return the two circle intersections}
  for each subset $\mathbf{A}_{\ell m} \neq \mathbf{A}_{jk} \in \binom{M}{2}$ two-anchor subsets **do**
    $\left(\mathbf{g}_i^{\ell m}, \overline{\mathbf{g}}_i^{\ell m}\right) \leftarrow CCI\left(\mathbf{q}_\ell, \mathbf{q}_m, R_{i\ell}, R_{im}\right)$ {Return the two circle intersections}
    $v_1 \leftarrow \left\|\mathbf{g}_i^{jk} - \mathbf{g}_i^{\ell m}\right\|^2$
    $v_2 \leftarrow \left\|\mathbf{g}_i^{jk} - \overline{\mathbf{g}}_i^{\ell m}\right\|^2$
    $\psi \leftarrow \psi + \min\left(v_1, v_2\right)$ {Return the minimum between v_1 and v_2}
    $w_1 \leftarrow \left\|\overline{\mathbf{g}}_i^{jk} - \mathbf{g}_i^{\ell m}\right\|^2$
    $w_2 \leftarrow \left\|\overline{\mathbf{g}}_i^{jk} - \overline{\mathbf{g}}_i^{\ell m}\right\|^2$
    $\varphi \leftarrow \varphi + \min\left(w_1, w_2\right)$ {Return the minimum between w_1 and w_2}
  end for
  if $(\psi < \varphi)$ **then**
    $\mathbf{T} \leftarrow \mathbf{T} + \mathbf{g}_i^{jk}$
  else
    $\mathbf{T} \leftarrow \mathbf{T} + \overline{\mathbf{g}}_i^{jk}$
  end if
end for
$\mathbf{p}_i^0 \leftarrow \frac{\mathbf{T}}{Q}$

[1] The algorithm omits the special cases where there are no circle intersections. The procedure for these instances is described further in the text.

Referring to our example, once node s_i removes the mirror locations, then an estimate of the node position can be formed by taking the *average* of the disambiguated set $\mathbf{G} = \{\mathbf{g}_i^{jk}, \mathbf{g}_i^{j\ell}, \mathbf{g}_i^{k\ell}\}$.

The complete bilateration scheme is described in Algorithm 2. This is a distributed localization algorithm in the sense that each node can implement it and determine its position estimate, given the anchor positions and the range estimates R_{ij} between each node and all the anchors. Since Algorithm 2 uses only anchor measurement, it can be used as an initialization step to generate a set of position estimates that can be used with algorithms that integrate more information from anchor and non-anchor nodes (*i.e.*, iterative distributed algorithms).

Figure 3. Possible cases where the triangle inequality is not satisfied. (**a**) Case A; (**b**) Case B.



There are some anomalous cases which should be considered in the bilateration algorithm. In order to get its initial estimation $\mathbf{p}_i^0$, it is essential that every sensor s_i gets the two location estimations from each one of the Q subsets even if the solutions are not feasible. For example, assume the two special cases shown in Figure 3. If we consider the left-side case on the figure, R_{ik} is shorter than d_{ik}, and R_{ij} is shorter than d_{ij}, clearly the triangle inequalities are not satisfied since

$$\begin{array}{lcl} R_{ij} + R_{ik} & > & \left\|\mathbf{q}_j - \mathbf{q}_k\right\| \\ R_{ij} + \left\|\mathbf{q}_j - \mathbf{q}_k\right\| & > & R_{ik} \\ R_{ik} + \left\|\mathbf{q}_j - \mathbf{q}_k\right\| & > & R_{ij} \end{array} \tag{31}$$

As a consequence, the sensor s_i will not be able to find any solution to Equation (28). In other words, if the two circles do not intersect with each other, it will not be feasible to find the circle intersections $\mathbf{g}_i$ and $\overline{\mathbf{g}}_i$. Therefore, a relaxed estimation should be generated as described next. Considering that $\left\|\mathbf{q}_j - \mathbf{q}_k\right\|$ is a constant distance between the anchors in set $\mathbf{A}_{jk}$, the node s_i takes two steps to estimate the locations

$\mathbf{g}_i^{jk}$ and $\bar{\mathbf{g}}_i^{jk}$. First, a location $\mathbf{x}_1$ is obtained by fixing R_{ik} and making $R_{ij} = \left|\left\|\mathbf{q}_j - \mathbf{q}_k\right\| - R_{ik}\right|$ in order to satisfy the triangle inequality. Next, the sensor s_i should use the *CCI* procedure to solve for $\mathbf{x}_1$. Similarly, a second location estimate $\mathbf{x}_2$ is obtained by fixing R_{ij}, choosing $R_{ik} = \left|\left\|\mathbf{q}_j - \mathbf{q}_k\right\| - R_{ij}\right|$ to satisfy the triangle inequality and solving the problem through the *CCI* procedure. Finally, both $\mathbf{g}_i^{jk}$ and $\bar{\mathbf{g}}_i^{jk}$ are generated as the average $\frac{\mathbf{x}_1+\mathbf{x}_2}{2}$ implying that when the triangle inequality is not satisfied, there will be a single solution that falls over the line y. A similar procedure can be derived for the second case as depicted in Figure 3.

4.1. Comparison with the Previous Bilateration Scheme

As described before, the research reported in [36] is focused on a distributed bilateration scheme that finds initial estimates. Using two anchors at a time, each sensor node s_i finds two possible candidates (*i.e.*, circle intersections). If sufficient anchors are available, the sensor node s_i averages the cloud of candidates which tend to be close to each other. The average of such candidates provides the initial estimate.

As can be seen the general idea for this approach is quite similar to our bilateration approach. However, there are relevant differences between the two schemes that should be taken into account. These differences make that our bilateration approach be an alternative for the scheme proposed in [36]. For example, one of the differences is that [36] does not take into account special cases when a sensor s_i is not able to compute circle intersections of two anchors (*i.e.*, the circles are not in touch) as shown Figure 3. Therefore, under this perspective this scheme is limited to naive scenarios in which estimated distances between sensors and anchors should have good accuracy. Thus, noisy RSS measurements, commonly used in realistic scenarios, may not provide useful information for this scheme. Hence, if a sensor s_i is not able to find sufficient circle intersections from two neighboring pair of anchors at time, the localization process will fail. In our case, the bilateration scheme is able to obtain initial estimates under the most severe scenarios (*i.e.*, not circle intersections).

Another important aspect to consider in [36], is the use of a threshold, δ, which reduces the number of possible candidate positions, making this approach more selective. However, the value of δ is hard to determine in practice and also it does not guarantee good results in noisy environments. Additionally, in [36] each sensor node s_i should create a table of its neighboring anchors. All anchors have a specific position inside of the table, and they are weighted by the sensor s_i according to the candidate positions that they generate. The value of δ is used to select a certain group of candidate positions. The anchors are weighted according to the candidates that they generated. Finally, all tables are broadcast by sensors. Once all sensors have received the anchor tables of their neighbors, they run a post-processing stage to determine which anchors are more reliable than others. These anchors are used to obtain initial estimates. As can be seen, the drawback of this approach are extra wireless transmissions required to share anchor tables among sensors. In our case we present an extension of the earlier BL algorithm which avoids any kind of wireless transmission with the goal to save energy. Finally, we should remark that we are using a sorting algorithm (lines 10,13, and 15–18 of the Algorithm 2) to determine initial positions. Analysis results shown in next section demonstrate that the alternative BL algorithm is competitive in comparison with well known accurate and efficient algorithms based on least-squares methodologies.

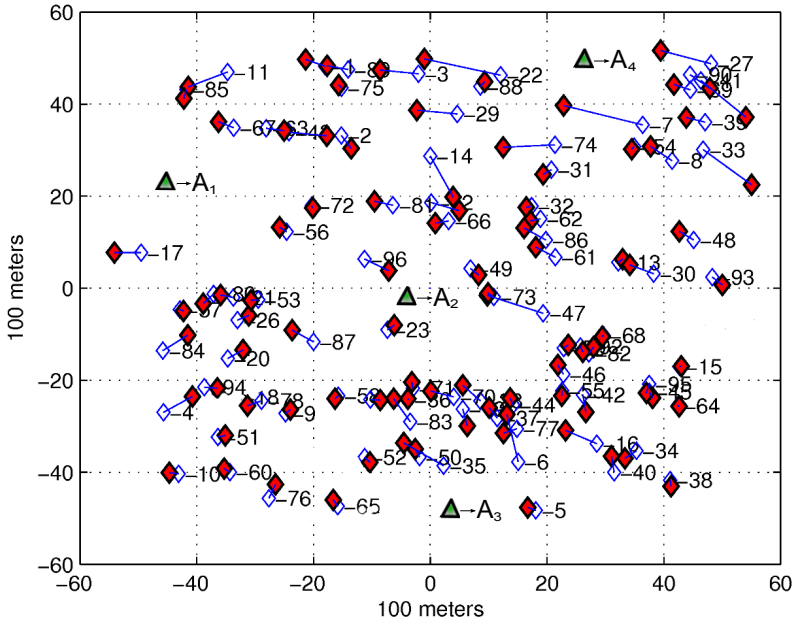
5. Accuracy Performance Between Closed-Formulas and Iterative Procedures in the WSN Localization Problem

In this section we analyze the accuracy performance of both methodologies, optimized and closed-formula schemes. Even though the strength of a closed-formula for solving the WSN localization problem is its low complexity compared with an iterative algorithm, closed-formulas can present large errors in the presence of inaccurate ranging measurements. However, in many cases it is desirable to sacrifice accuracy to save energy (*i.e.*, increase battery lifetime). On the other hand, the weakness for closed-form methods (*i.e.*, noise sensitivity) represents the strong point for iterative methods and vice versa. The goal of both methodologies seems to be in opposite directions. However, the main effort in WSN localization research is focused on developing an strategy that can join the strength of both methodologies to create an efficient algorithm that can save energy providing the best accuracy in the estimated positions.

Next we present an evaluation of accuracy between closed-formulas and iterative methodologies. For the former methods we are considering the classical LS Multilateration, the Min-max method (The Min-max approach is based on the intersection of rectangles instead of circle intersections. It provides a more simple technique than lateration schemes to obtain position estimates at the expenses of accuracy) [38,39], and the bilateration algorithm. For iterative methodologies we are considering two algorithms to solve the NLLS: the LM and the TRR algorithms.

For the simulations that follow, we consider 20 different sensor networks where each one is composed by $N = 96$ unknown sensors, randomly distributed over 100 m by 100 m area. Also, we add $M = 4$ non-collinear anchors with full-connectivity on every realization as shown in Figure 4.

Figure 4. A typical WSN composed by 96 unknown sensors with true positions='◇', 4 non-collinear anchors ='▲', and 96 initial estimates='⧫'. Each unknown sensor, using an initialization algorithm, estimates its initial position by using four reference positions $(A_1, A_2, A_3$, and $A_4)$ and their respective estimated distances.



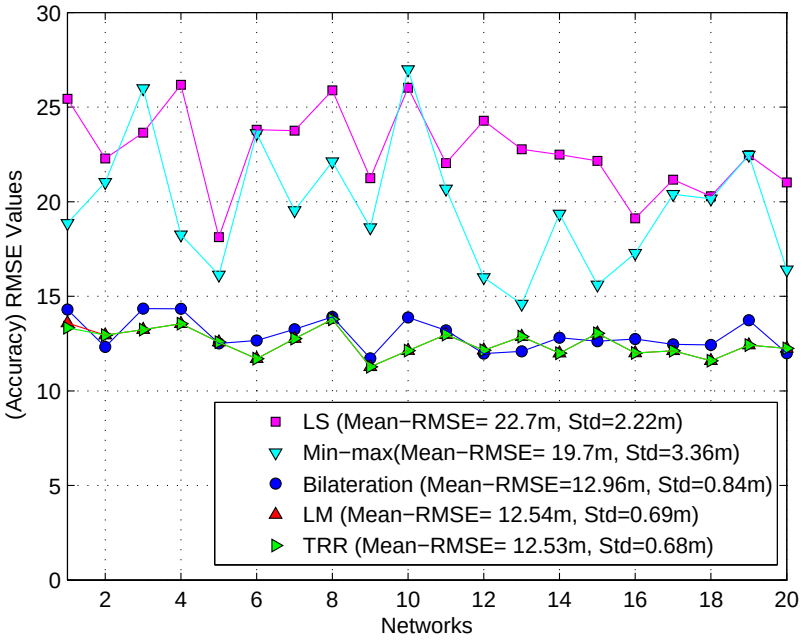
For each network, we add noise to the true distances between anchors and nodes using the log-distance path loss model described in Equation (4). The estimated distances are simulated using $\sigma_{SH} = 6_{dB}$ and $\eta_P = 2.6$, typical parameters for the propagation models on outdoors scenarios, and $P_0 = -52_{dB}$ is selected according to current commercial specifications for wireless motes [40]. Finally, we assume that all nodes have the sensitivity to detect any RF signal coming from anchors.

To compare the accuracy performance between both methodologies it was necessary to use the same set of range measurements for each closed-form method and iterative algorithm. Figure 5 summarizes the initial estimates obtained by both methodologies using the RMSE metric as shown the next equation:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N} \left\| \mathbf{p}_i^0 - \mathbf{z}_i \right\|} \tag{32}$$

where $\mathbf{p}_i^0$ represents an initial position estimate for a sensor s_i and $\mathbf{z}_i$ its true position.

Figure 5. Position estimates provided by different initialization algorithms.



As can be seen, the closed-form LS approach provides the least accurate initial estimates (mean = 22.7 m and standard deviation = 2.22 m) compared with iterative algorithms as expected due to the noisy ranging measurements. In a similar way, the Min-max scheme also provides large errors (mean = 19.7 m and standard deviation = 3.36 m). On the other hand, we can appreciate that both iterative algorithms, the LM and the TRR, provide practically the best and similar results for initial estimates (mean = 12.54 m and standard deviation = 0.69 m) as expected, and finally the bilateration algorithm presents very acceptable initial estimates compared with the last two algorithms (mean = 12.96 m and standard deviation = 0.84 m). However, we should consider that the computational complexity for the LM and the TRR algorithms is significantly larger than the bilateration algorithm. This discussion will be expanded in the next section.

Also, we tested the SX, SI and GSLS algorithms [25] using the same set of networks. The estimated positions presented large errors under this scheme as indicated by [26]. Then, these results were disregarded in our analysis.

6. Computational Complexity Analysis between the Bilateration and the LM Algorithm

The efficiency of an algorithm can be described in terms of the time or space complexity [41]. Time complexity refers to the relation between the number of required computations for a given input. The space used for a given input provides the space complexity of an algorithm. The computational complexity of an algorithm could be described as the number of operations that it takes to find a solution [42].

In this section we provide an operation count on the number of additions (ADDs), multipliers (MULs), divides (DIVs), and square roots (SQRTs) exactly in the way that DSP algorithms are described [43]. This will allow an "apple-to-apple" comparison. Moreover, an accurate description lends itself to a cycle accurate description for any microprocessor and more significantly, the use of energy models based on computing cycles to estimate the energy consumption for a given algorithm. An energy analysis will be a discussion of a future work. In next subsections we present the computational complexity analysis for the iterative LS and the bilateration algorithm.

6.1. Computational Analysis of the LM Algorithm

The LM algorithm could be considered as too expensive for motes given its iterative nature and the need to estimate first and second order information (*i.e.*, gradients, Jacobians, and Hessians). The number of iterations K is highly dependent on the initial point $\mathbf{x}_0$ and could be considered a random variable. On the other hand, if a good initial estimate, $\mathbf{x}_0$, is provided, then the number of iterations is expected to be low given the convergence properties of LM.

We are interested in providing an algorithmic analysis that provides a detailed description in terms of additions and subtractions (jointly referred as ADDs), multiplications (MULs), divisions (DIVs), and square roots (SQRTs). For simplicity in the next paragraphs we let $\mathbf{J}_k \equiv \mathbf{J}(\mathbf{x}_k)$ and $\mathbf{R}_k \equiv \mathbf{R}(\mathbf{x}_k)$.

The square root is a relevant operation as the error function $\mathbf{R}_k$ and the Jacobian estimate requires ℓ_2 norms to compute distances between sensor and anchors. We also note that the complexity of the operations is not the same in terms of the processing resources (hardware and software) they take. Abusing notation we have

$$\text{ADD} < \text{MUL} < \text{DIV} < \text{SQRT} \tag{33}$$

This analysis also focuses on the most efficient implementation in terms of the proper operation sequencing in order to favor reuse of terms (*i.e.*, avoid computing the same quantity twice).

We perform the analysis for a single iteration of the LM algorithm, and the total cost for each operation is multiplied by K. We also note that K can be modeled as a random variable; the usefulness of this approach is discussed later. We assume there are M anchors which have broadcast their position to all the nodes. Each node will run the LM algorithm to find its initial position as described in Subsection 3.2. We identify three core operations: ℓ_2 or Euclidean norm, the error vector $\mathbf{R}_k$ and an estimate of $\mathbf{J}_k$.

The ℓ_2 norm will be used to compute the magnitude of the difference between two vectors $\mathbf{a}, \mathbf{b} \in \mathbb{R}^2$ given by $\|\mathbf{a}-\mathbf{b}\| = \sqrt{(a_x-b_x)^2+(a_y-b_y)^2}$. This requires three ADDS, two MULs and one SQRT. The norm is used to compute $\mathbf{R}_k$ given in Equation (18) and to estimate the Jacobian as follows:

$$\mathbf{J}_k = \begin{bmatrix} \frac{-(x_1-x_k)}{\sqrt{(x_1-x_k)^2+(y_1-y_k)^2}} & \frac{-(y_1-y_k)}{\sqrt{(x_1-x_k)^2+(y_1-y_k)^2}} \\ \vdots & \vdots \\ \frac{-(x_M-x_k)}{\sqrt{(x_M-x_k)^2+(y_M-y_k)^2}} & \frac{-(y_M-y_k)}{\sqrt{(x_M-x_k)^2+(y_M-y_k)^2}} \end{bmatrix}_{M\times 2} \tag{34}$$

For $\mathbf{R}_k$ we see that we require M ADDs and M ℓ_2-norms. Accounting for the norms, the error function requires $4M$ ADDs, $2M$ MULs, and M SQRTs. These numbers are recorder in Table 1. A similar analysis follows for $\mathbf{J}_k$. A direct look at Equation (34) indicates that we have the same norm across rows, so we can compute them first and then we would need an additional $2M$ ADDs and $2M$ DIVs. However, a better approach would be to compute the terms $1/\|\mathbf{x}_j-\mathbf{x}_k\|$ first so that we would require M DIVs, $2M$ MULs, and $2M$ ADDs. We exchange M DIVs by $2M$ MULs under the typical case that MULs have a much lower complexity than DIVs, particularly for the case of floating point operations. The complexity for the Jacobian estimate is also shown on Table 1.

Table 1. LM Cost Functions.

Title	ADD	MUL	DIV	SQRT
$\mathbf{R}_k$	$4M$	$2M$	0	M
$\mathbf{J}_k$	$5M$	$4M$	M	M
$\mathbf{H}_k$	$3M-3$	$3M$	0	0
M'_f	$2M-2$	$2M$	0	0
M_f	$M-1$	$M+1$	0	0
μ_k	3	3	0	1
$\mathbf{H}_k^{-1}$	3	6	1	0
$\triangle_{LM}$	2	4	0	0
Sufficient Decrease	$T(M+4)$	$T(M+2)$	0	0
Update	2	2	0	2
Stopping Condition	3	2	0	1
Total	$(M+4)T+15M+7$	$(M+2)T+12M+18$	$M+1$	$2M+4$

Once these two quantities have been evaluated, their use trickles down through the algorithm. The costs for the different steps or operations is presented in the remaining part of Table 1. We just make two more remarks on the algorithm complexity. First, note that the approximation to the Hessian matrix $\mathbf{J}_K^T\mathbf{J}_K$

$$\nabla^2 f(\mathbf{x}_k) = \mathbf{J}_k^T\mathbf{J}_k = \begin{bmatrix} \sum_{j=1}^{M}\left(\frac{(x_j-x_k)^2}{\left((x_j-x_k)^2+(y_k-y_j)^2\right)}\right) & \sum_{j=1}^{M}\left(\frac{(x_j-x_k)(y_j-y_k)}{\left((x_j-x_k)^2+(y_j-y_k)^2\right)}\right) \\ \sum_{j=1}^{M}\left(\frac{(x_j-x_k)(y_j-y_k)}{\left((x_j-x_k)^2+(y_j-y_k)^2\right)}\right) & \sum_{j=1}^{M}\left(\frac{(y_j-y_k)^2}{\left((x_j-x_k)^2+(y_k-y_j)^2\right)}\right) \end{bmatrix}_{2\times 2} \tag{35}$$

is of size 2×2 which makes its inversion trivial when computing the LM step Δ_{LM}, as shown in Algorithm 1.

$$\triangle_{LM} = \left(\mathbf{J}_k \mathbf{J}_k^T + \mu_k \mathbf{I}\right)^{-1} \mathbf{J}_k^T \mathbf{R}_k \tag{36}$$

where the gradient of the function $\mathbf{J}_k^T \mathbf{R}_k$ is given by

$$\nabla f(\mathbf{x}_k) = \mathbf{J}_k^T \mathbf{R}_k = \left[\begin{array}{c} \sum_{j=1}^{M} \left(\frac{(x_j - x_k) \cdot \left(R_{kj} - \sqrt{(x_j - x_k)^2 + (y_j - y_k)^2} \right)}{\sqrt{(x_j - x_k)^2 + (y_j - y_k)^2}} \right) \\ \sum_{j=1}^{M} \left(\frac{(y_j - y_k) \cdot \left(R_{kj} - \sqrt{(x_j - x_k)^2 + (y_j - y_k)^2} \right)}{\sqrt{(x_j - x_k)^2 + (y_j - y_k)^2}} \right) \end{array} \right]_{2 \times 1} \tag{37}$$

Second, satisfying the sufficient decrease condition is also an iterative procedure where different values of α_k are tested. We identify T as the number of iterations needed to satisfy this condition. As we discuss later, we will model T as a random variable.

The last row of the table provides the total which we identify as T_{ADD}, T_{MUL}, T_{DIV} and T_{SQRT} respectively. These numbers are the operations for a single iteration of the LM algorithm. Then, for K iterations we have the total number of operations to be $K_{ADD} = K \cdot T_{ADD}$, $K_{MUL} = K \cdot T_{MUL}$, $K_{DIV} = K \cdot T_{DIV}$, and $K_{SQRT} = K \cdot T_{SQRT}$.

Since the values of T and K are random variables, then a more convenient approach to quantify the number of operations would be to look at the average number of operations, *i.e.*, the expected value. It is intuitive to assume that T and K are independent, and that for a given network their distributions will be identical. Hence, we define

$$\overline{K}_{ADD} = \varepsilon\{K_{ADD}\} = \varepsilon\{K\}\varepsilon\{T_{ADD}\} = \varepsilon\{K\}(\varepsilon\{T\}(M+4) + 15M + 7) \tag{38}$$

$$\overline{K}_{MUL} = \varepsilon\{K_{MUL}\} = \varepsilon\{K\}\varepsilon\{T_{MUL}\} = \varepsilon\{K\}(\varepsilon\{T\}(M+2) + 10M + 15) \tag{39}$$

$$\overline{K}_{DIV} = \varepsilon\{K_{DIV}\} = \varepsilon\{K\}(M+1) \tag{40}$$

$$\overline{K}_{SQRT} = \varepsilon\{K_{SQRT}\} = \varepsilon\{K\}(2M+2) \tag{41}$$

where $\varepsilon\{x\}$ represents the expected value of the random variable x. Finally, we can quantify the total complexity of the LM algorithm by converting operations to a common denominator and compute a single representative number that can be used for comparison with other algorithms. The typical way to quantify operations is to use the number of processor cycles (on the average) required to complete each type of operation. Let us define N_{ADD}, N_{MUL}, N_{DIV}, and N_{SQRT} as the number of cycles required for floating addition (or subtraction), a multiplication, a division, and square root, respectively. We should note that these numbers depend on the micro-processor and hardware used by the mote and the compiler tools used to develop the software. Hence, in practice the best way to obtain these values is through code profiling using a cycle-accurate simulator. Moreover, as discussed in [44], the number of task cycles can be used as part of models that measure energy consumption. Hence, as final measure of complexity for the LM algorithm we compute the total number of cycles as

$$N_{LM} = N_{ADD}\overline{K}_{ADD} + N_{MUL}\overline{K}_{MUL} + N_{DIV}\overline{K}_{DIV} + N_{SQRT}\overline{K}_{SQRT} \tag{42}$$

6.2. Computational Analysis of the Bilateration Algorithm

The bilateration algorithm is very simple and non-iterative. For M anchors, a sensor node picks $\binom{M}{2}$ pairs of sensors and computes the intersections of the imaginary circles around each anchor with a radius given between the anchor and the sensor node. These intersections are computed using geometry with a procedure described by Equations (24–28). Then, a cluster with half of the computed intersections is found, providing an indication of the area where the node position is located. The number of operations required to compute two intersections is presented in Table 2.

Table 2. Bilateration Cost Operations.

Operations	ADD	MUL	DIV	SQRT
d	3	2	0	1
d_{jt}	1	5	1	0
h	1	1	0	1
$r = \frac{h}{d}$	0	0	1	0
$\mathbf{f}_t$	2	2	1	0
$\mathbf{x}_i$	1	1	0	0
$\overline{\mathbf{x}}_i$	1	0	0	0
$\mathbf{y}_i$	1	1	0	0
$\overline{\mathbf{y}}_i$	1	0	0	0
Total (2 circle intersections)	11	12	3	2
Total Q node combinations	$11Q$	$12Q$	$3Q$	$2Q$

Since this process is repeated Q= $\binom{M}{2}$ times, then the final row reflects the total operations multiplied by this factor. As the intersections are computed, the search for the cluster is performed by Equations (29) and (30). Since there are $2Q$ intersections, we need to select the Q that cluster together (*i.e.*, eliminate mirrors). The clustering is based on looking at the distance between all possible pairs of intersections and selecting those that exhibit the closets distances among themselves. This requires the calculation of $S = \frac{2Q(2Q-1)}{2}$ squared norms, and the use of a clustering or sorting algorithm to find the smallest Q elements from the list of S norm values. Taking advantage of the structure of the location points (*i.e.*, the two intersections from the same anchor pair are not compared), we can expect an average complexity of $O(S)$ sorting steps using an algorithm like Quickselect algorithm [45]. Hence the final computational cost for the bilateration algorithm is presented in Table 3.

Table 3. Final Computational Cost for the Bilateration Scheme.

Action	ADD	MUL	DIV	SQRT	SORT
Circle Intersections	$11Q$	$12Q$	$3Q$	$2Q$	0
Squared Norms	3S	2S	0	0	0
Number of Comparisons	0	0	0	0	$O(S)$

As with the LM algorithm, we close this subsection by providing an expression in terms of processor cycles. Using the same characterization for all main operations of the algorithm, we can provide a total

cycle count that can be directly compared with other algorithms. Obviously, a lower cycle implies lower complexity when the hardware and software development tools are identical. The expression for total cycles is

$$N_{BL} = N_{ADD} \cdot (11Q + 3S) + N_{MUL} \cdot (12Q + 2S) + N_{DIV} \cdot (3Q) + N_{SQRT} \cdot (2Q) + N_{SORT} \tag{43}$$

It is easy to see that the bilateration scheme uses a significantly less number of cycle for all operations. Experimental data in [45] indicates that the cycle count for the complete sorting step with the *QuickSelect* algorithm with a pipelined architecture can be achieved within 2500 and 3000 clock cycles.

To complete the computational complexity analysis, we need the number of CPU cycles required for the four basic operations as floating point operations. These values are highly dependent on the architecture of the mote processor. An extensive study in [46] provides good representative values for processors with some level of hardware support. The values are summarized on Table 4, and it shows the relation between basic operations and CPU cycles.

Table 4. Operation cycle counts.

ADD	MUL	DIV	SQRT
11	25	112	119

Next, we use Tables 1 and 2 to obtain the number of CPU cycles required by each initialization stage, BL and LM respectively. For the LM initialization stage we are using M = 4 anchors and the random variables T and k. We recall that k is the number of iterations spent by the LM algorithm to find a solution. These values are obtained through simulations where $\varepsilon\{T\} = 2$ and $\varepsilon\{k\} = 13$. In this way the total cycles required by the LM algorithm according to Equation (42) is given by

$$\begin{aligned} K_{LM} = & \varepsilon\{k\}\,[((M+4)\,\varepsilon\{T\} + 15M + 7)\,(11) + \\ & ((M+2)\,\varepsilon\{T\} + 12M + 18)\,(25) + \\ & (M+1)(112) + \\ & (2M+4)(119)\,] = 63063\,\text{cycles} \end{aligned} \tag{44}$$

Similarly, the total number of cycles used by the BL stage is given by Equation (43) as

$$\begin{aligned} K_{BL} = & (11)(11Q + 3S) + \\ & (25)(12Q + 2S) + \\ & (112)(3Q) + \\ & (119)(2Q) + \\ & (N_{SORT}) = 14198\,\text{cycles} \end{aligned} \tag{45}$$

where $Q = 6, S = 66$, and $N_{SORT} = 2750$. The value for N_{SORT} represents the total number of cycles required to perform the sorting step of the BL algorithm. This step can be performed using efficiently the Quickselect algorithm [47]. As expected, the LM algorithm consumes more energy in the initialization

process than the BL scheme. However, the former represents a better choice when accuracy is required. Therefore, the BL can be an alternative localization scheme when a good tradeoff between accuracy and energy consumption is required on the initial estimates.

7. Conclusions

In this research, we analyzed a localization algorithm that can be realistically deployed over real WSNs which can provide good accuracy performance with low computational complexity. The bilateration algorithm is a distributed scheme that can be used as an initialization stage to find an initial set of locations.

Most initialization algorithms demand very high computing power to provide a set of initial estimates for an N-node WSN. The analyzed algorithm is capable to provide competitive initial estimates at low processing power. This approach is basically formed by two stages. The first stage consists of finding all circle intersections formed by anchor positions and their respective range estimates to a sensor node, obtained by ranging techniques like *ToA* or *RSS*. The great advantage of this approach is to use "closed-formulas" to find all circle intersections (*i.e.*, candidate positions) using two anchors at a time. In the second stage, the algorithm uses a sorting algorithm to find the cluster of candidate positions that tend to be closer to each other around the true location. The cluster with the nearby candidate positions is averaged to finally obtain the initial location. This scheme can be used by any WSN localization algorithm that needs initial approximations. Also, it is implementable in constrained devices with low processing and memory capabilities (*i.e.*, motes). Results show that this initialization algorithm is well behaved (e.g., computational and accuracy performance) in comparison with other well known algorithms like LS methodologies.

Finally, we are interested in exploring iteratively, at the refinement process, the Levenberg–Marquardt approach for node localization. We believe that this methodology can play a crucial role in producing excellent position estimates with high accuracy and low energy consumption due to the rate of convergence associated with this optimization technique.

Acknowledgements

The authors wish to express their gratitude to the anonymous reviewers for their invaluable comments.

References

1. Zhong, Z.; Wang, D.; He, T. Sensor Node Localization Using Uncontrolled Events. In *Proceedings of the 28th International Conference on Distributed Computing Systems (ICDCS '08)*, Beijing, China, 17–20 June 2008; pp. 438–445.
2. Youssef, A.; Youssef, M. A Taxonomy of Localization Schemes for Wireless Sensor Networks. In *Proceedings of the International Conference on Wireless Networks*, Las Vegas, NV, USA, 25–28 June 2007.
3. Chan, F.; So, H. Accurate distributed range-based positioning algorithm for wireless sensor networks. *IEEE Trans. Signal Process.* **2009**, *57*, 4100–4105.

4. Niculescu, D.; Nath, B. DV based positioning in *ad hoc* networks. *Telecommun. Syst.* **2003**, *22*, 267–280.
5. He, T.; Huang, C.; Blum, B.; Stankovic, J.; Abdelzaher, T. Range-free localization and its impact on large scale sensor networks. *ACM Trans. Embed. Comput. Syst.* **2005**, *4*, 877–906.
6. Stoleru, R.; Stankovic, J. Probability Grid: A Location Estimation Scheme for Wireless Sensor Networks. In *Proceedings of the 2004 1st Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (IEEE SECON '04)*, Santa Clara, CA, USA, 4–7 October 2004; pp. 430–438.
7. Goldenberg, D.; Bihler, P.; Cao, M.; Fang, J.; Anderson, B.; Morse, A.; Yang, Y. Localization in Sparse Networks Using Sweeps. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*, Los Angeles, CA, USA, 24–29 September 2006; pp. 110–121.
8. Akyildiz, I.; Su, W.; Sankarasubramaniam, Y.; Cayirci, E. A survey on sensor networks. *IEEE Commun. Mag.* **2002**, *40*, 102–114.
9. Gezici, S. A survey on wireless position estimation. *Wirel. Pers. Commun.* **2008**, *44*, 263–282.
10. Li, M.; Liu, Y. Rendered path: Range-free localization in anisotropic sensor networks with holes. *IEEE/ACM Trans. Netw.* **2010**, *18*, 320–332.
11. Jordt, G.; Baldwin, R.; Raquet, J.; Mullins, B. Energy cost and error performance of range-aware, anchor-free localization algorithms. *Ad Hoc Netw.* **2008**, *6*, 539–559.
12. Mao, G.; Fidan, B.; Anderson, B. Wireless sensor network localization techniques. *Comput. Netw.* **2007**, *51*, 2529–2553.
13. Stoleru, R.; He, T.; Stankovic, J. Range-Free Localization. In *Secure Localization and Time Synchronization for Wireless Sensor and Ad Hoc Networks*; Springer: Berlin, Germany, 2007; pp. 3–31.
14. Yu K., Guo, Y. Robust Localization in Multihop Wireless Sensor Networks. In *Proceedings of the Vehicular Technology Conference (VTC Spring 2008)*, Singapore, 11–14 May 2008; pp. 2819–2823.
15. Moses, R.; Krishnamurthy, D.; Patterson, R. A self-localization method for wireless sensor networks. *EURASIP J. Appl. Signal Process.* **2003**, *4*, 348–358.
16. Doherty, L.; El Ghaoui, L. Convex Position Estimation in Wireless Sensor Networks. In *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '01)*, Anchorage, AK , USA, 22–26 April 2001; Volume 3, pp. 1655–1663.
17. Chan, F.; So, H. Efficient weighted multidimensional scaling for wireless sensor network localization. *IEEE Trans. Signal Process.* **2009**, *57*, 4548–4553.
18. Weng,Y.; Xiao, W.; Xie, L. Diffusion-based em algorithm for distributed estimation of gaussian mixtures in wireless sensor networks. *Sensors* **2011**, *11*, 6297–6316.
19. Liu, J.; Zhang, Y.; Zhao, F. Robust Distributed Node Localization with Error Management. In *Proceedings of the 7th ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Florence, Italy, 22–25 May 2006; pp. 250–261.
20. TarrÃo, P.; Bernardos, A.M.; Casar, J.R. Weighted least squares techniques for improved received signal strength based localization. *Sensors* **2011**, *11*, 8569–8592.

21. Yang, Z.; Liu, Y. Quality of trilateration: Confidence-based iterative localization. *IEEE Trans. Parallel Distrib. Syst.* **2009**, *21*, 631–640.
22. Patwari, N.; Ash, J.; Kyperountas, S.; Hero, A.; Moses, R.; Correal, N. Locating the nodes. *IEEE Signal Process. Mag.* **2005**, *22*, 54–69.
23. Andersen, J.; Rappaport, T.; Yoshida, S. Propagation measurements and models for wireless communications channels. *IEEE Commun. Mag.* **1995**, *33*, 42–49.
24. Rappaport, T.S. *Wireless Communications: Principles and Practice*; Prentice Hall PTR: Upper Saddle River, NJ, USA, 1996; Volume 207.
25. Huang, Y.; Benesty, J.; Chen, J. *Acoustic MIMO Signal Processing*; Springer: Berlin, Germany, 2006.
26. Huang, Y.; Benesty, J.; Elko, G.; Mersereati, R. Real-time passive source localization: A practical linear-correction least-squares approach. *IEEE Trans. Speech Audio Process.* **2002**, *9*, 943–956.
27. Chen, H.; Sezaki, K.; Deng, P.; So, H. An Improved DV-Hop Localization Algorithm for Wireless Sensor Networks. In *Proceedings of the 3rd IEEE Conference on Industrial Electronics and Applications (ICIEA '08)*, Singapore, 3–5 June 2008; pp. 1557–1561.
28. Verdone, R.; Dardari, D.; Mazzini, G.; Conti, A. *Wireless Sensor and Actuator Networks: Technologies, Analysis and Design*; Elsevier: Amsterdam, The Netherlands, 2008; p. 211.
29. Dennis, J.; Schnabel, R. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*; Society for Industrial Mathematics: Englewood Cliffs, NJ, USA, 1996; p. 227.
30. Nocedal, J.; Wright, S. *Numerical Optimization*; Springer: Berlin, Germany, 2006; p. 8.
31. Cheng, B.; Vandenberghe, L.; Yao, K. Distributed algorithm for node localization in wireless *ad-hoc networks*. *ACM Trans. Sens. Netw.* **2009**, *6*, 1–20.
32. Costa, J.A.; Patwari, N.; Hero, A.O., III. Distributed weighted-multidimensional scaling for node localization in sensor networks. *ACM Trans. Sensor Netw.* **2006**, *2*, 39–64.
33. Delbos, F.; Gilbert, J.; Glowinski, R.; Sinoquet, D. Constrained optimization in seismic reflection tomography: A Gauss–Newton augmented Lagrangian approach. *Geophys. J. Int.* **2006**, *164*, 670–684.
34. Roweis, S. *Levenberg-Marquardt Optimization*; University Of Toronto: Toronto, ON, Canada, 1996.
35. Ye, N. *The Handbook of Data Mining*; Lawrence Erlbaum: Mahwah, NJ, USA, 2003.
36. Li, X.; Hua, B.; Shang, Y.; Xiong, Y. A robust localization algorithm in wireless sensor networks. *Front. Comput. Sci. China* **2008**, *2*, 438–450.
37. Bourke, P. Intersection of Two Circles, 1997. Available online: http://local.wasp.uwa.edu.au/∼pbourke/geometry/2circle/ (accessed on 9 January 2010).
38. Savvides, A.; Park, H.; Srivastava, M. The Bits and Flops of the N-Hop Multilateration Primitive for Node Localization Problems. In *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, Atlanta, GA, USA, 28 September 2002; pp. 112–121.
39. Langendoen, K.; Reijers, N. Distributed localization in wireless sensor networks: A quantitative comparison. *Comput. Netw.* **2003**, *43*, 499–518.
40. XST-AN019a, A.N. XBee and XBee-PRO OEM RF Module Antenna Considerations, 2005. Available online: http://www.digi.com (accessed on 9 January 2010).

41. Cormen, T. *Introduction to Algorithms*; The MIT Press: Cambridge, MA, USA, 2001; p. 23.
42. Schörghofer, N. *The Third Branch of Physics: Essays in Scientific Computing*; Nobert: Hamburg, Germany, 2005; pp. 53–71.
43. Wang, A.; Chandrakasan, A. Energy-efficient DSPs for wireless sensor networks. *IEEE Signal Process. Mag.* **2002**, *19*, 68–78.
44. Sinha, A.; Chandrakasan, A. Energy Aware Software. In *Proceedings of the 13th International Conference on VLSI Design*, Calcutta, India, 3–7 January 2000; pp. 50–55.
45. TI-Algorithms. Optimized Sort Algorithms for DSP, 2011. Available online: http://processors.wiki.ti.com/index.php/Optimized_Sort_Algorithms_For_DSP (accessed on 9 January 2012).
46. Dietz, H.; Dieter, B.; Fisher, R.; Chang, K. Floating-Point Computation with Just Enough Accuracy. In *Proceedings of the Computational Science (ICCS '06)*, Reading, UK, 28–31 May 2006; pp. 226–233.
47. Mansi, R. Enhanced quicksort algorithm. *Int. Arab J. Inform. Technol.* **2010**, *7*, 161–166.