Macrothink Institute™

Network Protocols and Algorithms
ISSN 1943-3581
2011, Vol. 3, No. 3

A Novel Two-Class Localization Algorithm in Wireless Sensor Networks

Linqing GUI, Thierry VAL

Laboratory CNRS-IRIT-UT2, University of Toulouse

IUT Blagnac-dpt R&T, 1 Place Georges Brassens, 31703 Blagnac (France)

Tel: +33562747584 Email: gui@irit.fr, val@irit.fr

Anne WEI

Laboratory CEDRIC, CNAM-Paris

292 Rue St Martin, 75141 Paris (France)

Tel: +33140272296 Email: anne.wei@cnam.fr

Received: August 11, 2011 Accepted: November 22, 2011 Published: December 18, 2011

DOI: 10.5296/npa.v3i3.863 URL: http://dx.doi.org/10.5296/npa.v3i3.863

Abstract

For most of the applications in wireless sensor networks, localization is fundamental and essential. The localization systems can be categorized into two types: range-based and range-free. In this paper, we propose a new range-free localization algorithm. The basic principle of this algorithm is to define two classes of nodes according to the number of anchors in the neighborhood, and then to apply the proposed localization methods to each class of nodes. The computation complexity of related algorithms has also been estimated and analyzed. The simulation results prove that our proposed two-class algorithm has acceptable complexity, and achieves better precision than the existing ones, such as Centroid, CPE and DV-hop.

Keywords: localization, range-based, range-free, wireless sensor networks.

1. Introduction

Recent years, wireless sensor networks have been hot topics in worldwide research and industrial fields, for their vast applications such as medical care, disaster relief, environmental monitoring, and military surveillance. For these applications, localization is fundamental and essential. Without precise knowledge of the location, many applications of wireless sensor networks cannot work [1].

The existing localization techniques can be generally categorized into two types: range-based and range-free.

In the range-based scheme, the distance or angle between nodes must first be precisely measured. This can achieved using Received Signal Strength Indicator (RSSI) [2], Time of Arrival (TOA) [3], Time Difference of Arrival (TDOA) [4], and Angle of Arrival (AOA) [5]. Then, the position can be estimated by the trilateration or triangulation [2-5], which is the process to determine the node position, based on its distances or angles to at least 3 other nodes. The range-based scheme usually has two main drawbacks. First, additional ranging devices are needed, which consumes more energy and increases the cost. Second, range information is very easily affected by multi-path fading, noise and environment variations.

While the range-based scheme uses the distance or angle between nodes, the range-free scheme uses connectivity information between nodes. So the range-free scheme doesn't need the additional ranging devices. This scheme could be very helpful for very simply constructed sensor nodes. Another advantage of range-free scheme is its robustness, because the connectivity between nodes is not so easily affected by the environment. So, we focus our research on the range-free scheme. In this scheme, the nodes which are aware of their position are called anchors, while others are called normal nodes. Normal nodes first gather the connectivity information as well as the position of anchors, and then calculate their own positions. The existing typical range-free localization algorithms, such as Centroid [6] [7], CPE (Convex Position Estimation) [8] and DV-hop (Distance Vector-Hop) [9-11], are not accurate enough. Therefore, the localization accuracy of range-free methods should be further improved.

In this paper, we propose a new range-free algorithm. In this algorithm, according to the number of accessible anchors, the normal nodes are divided into two groups. Normal nodes of the first class are those which have less than three anchors in their neighborhood, while those of the second class have at least three anchors. For the normal nodes of the first class, a Checkout DV-hop method is proposed. For the normal nodes of the second class, we propose a Mid-Perpendicular method.

We analyze the computation complexity of the related algorithms. Both the theoretical analysis and simulation results prove that Checkout DV-hop has a negligible increase in complexity, while Mid-Perpendicular has an acceptable increase. Simulation results show that

our proposed algorithm has better precision than the existing ones such as Centroid, CPE and DV-hop.

The rest of this paper is organized as follows. Section 2 presents a survey of related range-free works. Section 3 presents our proposed algorithm. Although part of Section 3 has already been included in our previous work [12], we here will give more detailed explanation and also add the exceptional cases. In Section 4, the computation complexities of related algorithms are estimated. Section 5 presents the simulation results and analysis. Finally we give our conclusion and future prospects in Section 6.

2. Related Work

In this section, the most relevant range-free research works are reviewed. They were developed in the aim of different goals including accuracy, cost, and scalability. Some of them, such as Centroid and CPE, are very simple, but demand that normal nodes have at least three neighbor anchors. Others, like DV-hop, can handle the case where a normal node has less then three neighbor anchors. But this is achieved at the cost of heavy communication and computation.

Centroid and CPE are two well-known range-free localization algorithms. They are designed for normal nodes which have at least three neighbor anchors. These methods assume that the normal node N has m neighbor anchors A_1, $A_2 \ldots A_m$, whose coordinates are respectively (x_1, y_1) (x_2, y_2) … (x_m, y_m), and that all nodes have identical communication range. The principle of Centroid algorithm is as follows: when the normal node N wants to determine its position, it listens to the neighbor anchors' beacons, and from the information gathered, it computes its position, denoted as $N_{centroid}$ $(x_{centroid}, y_{centroid})$, using equation (1).

$$x_{centroid} = (\sum_{i=1}^{m} x_i)/m \ , \quad y_{centroid} = (\sum_{i=1}^{m} y_i)/m \tag{1}$$

The Centroid algorithm has a low computation cost, and doesn't increase the network traffic. It can also get relatively good accuracy when the distribution of anchors is regular. However, when the distribution of anchors is not even, the estimated position derived from the Centroid algorithm will be inaccurate [13].

CPE has slightly higher localization accuracy than Centroid. The basic principle of CPE is to define the estimative rectangle (ER), which bounds the overlapping communication region of A_1 $A_2 \ldots A_m$. Then, the centre of the estimated rectangle, denoted as N_{CPE}, is regarded as the estimated position for N. The position of N_{CPE}, that is $[x_{CPE}, y_{CPE}]$, can be calculated as:

$$\begin{cases} x_{CPE} = [\max\limits_{i=1}^{m}(x_i) + \min\limits_{i=1}^{m}(x_i)]/2 \\ y_{CPE} = [\max\limits_{i=1}^{m}(y_i) + \min\limits_{i=1}^{m}(y_i)]/2 \end{cases} \tag{2}$$

The DV-hop algorithm was proposed by Niculescu [9]. Although its complexity is higher than that of Centroid and CPE, it is a suitable solution for normal nodes having less than three neighbour anchors. As shown in Figure 1, although the normal node N_x has only one neighbour or reachable anchor A_1, N_x can use DV-hop for localization. The algorithm consists of the following three steps. First, each anchor A_i broadcasts its position; this information is relayed by the normal nodes so that N_x knows every anchor A_i's positions as well as the minimal hop count from N_x to A_i (denoted as $hop_{i,Nx}$). Second, each anchor A_i calculates dhp_i, which is the average distance per hop, and then broadcastes it. Third, N_x obtains its distance to each anchor A_i by multiplying $hop_{i,Nx}$ by dhp_i, and then N_x finally uses this distance to each anchor to calculate its estimated position.

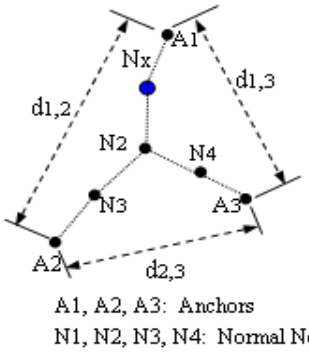

Figure 1. Example of DV-hop

In [10], an improved DV-hop is proposed, which makes all anchors share the same average distance per hop. The average distance per hop defined in [10] is the average of all dhp_i. In [11], a robust weighted algorithm is presented, which uses the weighted method to calculate the average distance per hop. It demands complex calculation but results in only slight improvement of performance.

From the previous literature review, we find that different proposed algorithms have respective qualities and drawbacks, in particular in terms of precision. This encouraged us to propose a range-free localization algorithm in order to reach a satisfying trade-off.

3. A Novel Localization Algorithm

According to the number of neighbour anchors, we divide the normal nodes into two classes. Normal nodes in class one are those which have less than three neighbour anchors, while those in class two have at least three. We propose two different localization methods for the two classes respectively.

3.1 Checkout DV-hop Method for Class One Nodes

3.1.1 Principle of the Checkout DV-hop Method

For normal nodes of class one, DV-hop is a frequently-used localization method. The key issue of DV-hop is to calculate the approximate distance between the normal node N_x and each anchor A_i, by multiplying minimal hop number by average distance per hop. This means that:

$$d_{i,\mathrm{Nx}} = hop_{i,\mathrm{Nx}} \times dhp_i, i = 1,2....m \quad (3)$$

Where $d_{i,\mathrm{Nx}}$ is the approximate distance between N_x and A_i, $hop_{i,Nx}$ is the minimal hop number between N_x and A_i, and dhp_i is the approximate average distance per hop for A_i. The calculation of dhp_i is shown as:

$$dhp_i = (\sum_{k(k\neq i)} d_{i,k}) / (\sum_{k(k\neq i)} hop_{i,k}) \quad (4)$$

Where $d_{i,k}$ is the distance between A_i and A_k, $hop_{i,k}$ is the minimal hop number between A_i and A_k.

Since $d_{i,\mathrm{Nx}}$ is an important and basic element for calculating the position of the normal node N_x [9][10], it has a considerable influence on the accuracy of DV-hop. We denote the true distance from N_x to A_i as $d_{i,Nx\mathrm{True}}$, and the difference between $d_{i,Nx\mathrm{True}}$ and $d_{i,\mathrm{Nx}}$ as $\Delta d_{i,Nx}$, where obviously $\Delta d_{i,Nx}$ is the main reason for the inaccuracy of DV-hop. If we denote Δdhp_i as the difference between dhp_i and its true value, then from equation (3), we have that:

$$\Delta d_{i,\mathrm{Nx}} = hop_{i,\mathrm{Nx}} \times \Delta dhp_i \quad (5)$$

Equation (5) indicates that $\Delta d_{i,Nx}$ is proportional to $hop_{i,Nx}$. So when $hop_{i,Nx}$ increases, $\Delta d_{i,Nx}$ also increases, and the accuracy of DV-hop decreases. If A_{near} is the nearest anchor to N_x among all anchors $A_1\ A_2\ \ldots\ A_m$, then correspondingly $hop_{near,Nx}$ is the smallest, so that $\Delta d_{near,Nx}$ is the smallest position error. So we can conclude that, compared to other anchors, the distance from the normal node N_x to its nearest anchor A_{near}, denoted as $d_{near,\mathrm{Nx}}$, has the highest reliability in terms of precision. Based on this conclusion, our proposed Checkout DV-hop method tries to make best use of the relatively more reliable value $d_{near,\mathrm{Nx}}$.

Now we illustrate the principle of our algorithm for class one nodes. Our method adds a checkout step to DV-hop, as shown in Figure 2. For the purpose of comparison, Figure 2(a) shows the result of DV-hop without "checkout", while Figure 2(b) shows our checkout step. As shown in Figure 2(a), the normal node N_x uses DV-hop algorithm to obtain its estimated position (x', y') at $N_{DV\text{-}hop}$, and then calculate the distance between $N_{DV\text{-}hop}$ and A_{near} (here, A_{near} is A_1), denoted as $d_{DV\text{-}hop}$. N_x has already used equation (3) to calculate its approximate distance to the nearest anchor A_{near}, denoted as $d_{near,\mathrm{Nx}}$.

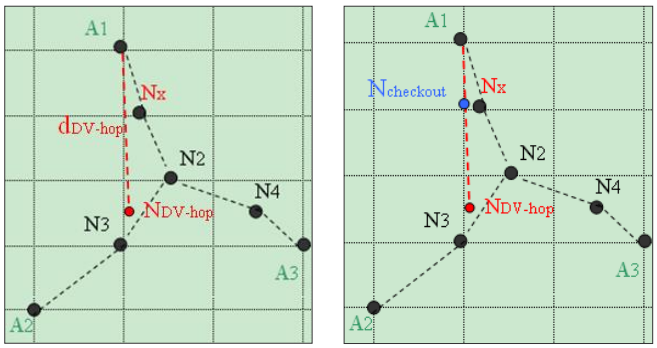

Figure 2(a). DV-hop Figure 2(b). Checkout DV-hop

Figure2. Principle of Checkout DV-hop

At this point, it is time to perform our proposed checkout step. The purpose of the checkout step is to change the estimated position from $N_{DV\text{-}hop}$ (see Figure2(b)) to a new one $N_{checkout}$, whose distance to A_{near} is $d_{near,Nx}$. To achieve this, the easiest and quickest way is to change the position along the line connecting $N_{DV\text{-}hop}$ and A_{near}. $N_{checkout}$ is on the line from $N_{DV\text{-}hop}$ to A_{near}, and the distance between $N_{checkout}$ and A_{near} is $d_{near,Nx}$. The position of A_{near} is (x_{Anear}, y_{Anear}) and of $N_{DV\text{-}hop}$ is (x', y'), then the position of $N_{checkout}$, denoted as $(x_{checkout}, y_{checkout})$ can be derived as follows. $N_{checkout}$ is chosen as our node estimated position.

$$\begin{cases} x_{checkout} = x' - \left(\dfrac{d_{\text{DV-hop}} - d_{\text{near,Nx}}}{d_{\text{DV-hop}}}\right) \times (x' - x_{Anear}) \\ y_{checkout} = x' - \left(\dfrac{d_{\text{DV-hop}} - d_{\text{near,Nx}}}{d_{\text{DV-hop}}}\right) \times (y' - y_{Anear}) \end{cases} \quad (6)$$

3.1.2 Procedure of the Checkout DV-hop Method

Our method comprises four steps. The fourth step, which is checkout step, is proposed by us, while the first three steps are the same with DV-hop. The procedure of our method is presented as follows.

Step One: Initially, the system installer ensures that each anchor *Ai* is aware of its own position (x_i, y_i). Every node N_h, including anchors, holds a variable $hop_{i,Nh}$, which represents the minimal hop number from N_h to *Ai*. N_h initializes $hop_{i,Nh}$ as -1(if N_h is not *Ai*), or 0(if N_h is *Ai*). Then, *Ai* broadcasts a beacon "Beacon*Ai*" to its neighbors. Beacon*Ai* contains *Ai*'s position (x_i, y_i) and its hop-number value 0. When *Ai*'s neighbor node N_q receives Beacon*Ai*, N_q memorizes (x_i, y_i), and changes $hop_{i,Nq}$ from 0 to 1. Then N_q also broadcasts to its neighbors a new beacon "Beacon$_{Nq}$", containing (x_i, y_i) and $hop_{i,Nq}$. When N_q's neighbor N_r receives Beacon$_{Nq}$, N_r memorizes (x_i, y_i) and compares $hop_{i,Nr}$ to $hop_{i,Nq}$+1, and also to -1. To maintain the minimal hop number to *Ai*, if $hop_{i,Nr}$=-1 or $hop_{i,Nr}$>$hop_{i,Nq}$+1, N_r changes $hop_{i,Nr}$ to $hop_{i,Nq}$+1, and N_r broadcasts "Beacon$_{Nr}$" , containing (x_i, y_i) and $hop_{i,Nr}$. Otherwise, N_r just ignores Beacon$_{Nq}$. The

rest of the nodes do the same operation as N_r. Through this mechanism, all nodes in the network get the minimal hop number to every anchor.

Step Two: at this point, each normal node *Nx* knows its $hop_{i,Nx}$ (minimal hop number from *Ai* to *Nx*). Each anchor *Ai* has also obtained $hop_{i,k}$, which is the minimal hop number to any other anchor A_k. So *Ai* can calculate its average distance per hop dhp_i, and then broadcasts dhp_i through the network. After receiving dhp_i, the normal node *Nx* can use equation (3) to get $d_{i,Nx}$, which is the approximate distance between *Nx* and each anchor *Ai*. If *Anear* is the nearest anchor to *Nx*, according to the conclusion in section 3.1.1, $d_{near,Nx}$ will be used in the fourth step (our checkout step).

Step Three: The normal node *Nx* use the approximate distance to each anchor to calculate its estimated position $N_{DV\text{-}hop}(x', y')$. The details of the third step can be found in [9][10].

Step Four: Finally, with our proposed checkout step, the normal node *Nx* calculates the distance between $N_{DV\text{-}hop}$ and A_{near}, denoted as $d_{DV\text{-}hop}$. Because *Nx* already knows $d_{i,Nx}$, *Ai*'s position (x_A, y_A), $N_{DV\text{-}hop}$'s position (x', y'), and $d_{DV\text{-}hop}$, *Nx* uses equation (6) to calculate $N_{checkout}$, which is the final estimated position of *Nx*.

3.2 Mid-Perpendicular Method for Class Two Nodes

For normal nodes of class two, Centroid and CPE are popular methods because of their low communication and computation cost, regardless of their inaccuracy. However, our proposed Mid-Perpendicular method is able to achieve higher accuracy.

We assume that the communication ranges of anchors are all the same as displayed by the circles of Figure 3. The normal node *N* has three neighbour anchors ($A_1 A_2 A_3$). It means that *N* locates in the overlapping communication region of $A_1 A_2 A_3$.

Figure 3 shows how to derive the centre of overlapping region. Line A_2A_3 connects anchors A_2 and A_3, and the mid-perpendicular of line A_2A_3 is "*Line1*". According to the symmetry, *Line1* go through the centre of the overlapping region. If the same observation is made for line A_1A_2 and line A_1A_3, there will be a total of three mid-perpendiculars: *Line1*, *Line2*, and *Line3*. Since each mid-perpendicular goes through the centre of the overlapping region, the cross point of the three mid-perpendiculars "*P*" can be regarded as the centre. In fact, in order to calculate the cross point *P*, only two mid-perpendiculars are needed, for example, *Line1* and *Line2*.

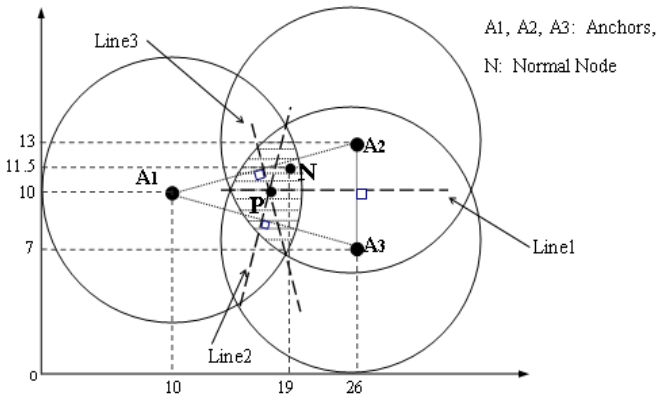

Figure 3. Diagram of Mid-Perpendicular Method

If the coordinates of three anchors A_1 A_2 A_3 are (x_1, $y1$), (x_2, $y2$), and (x_3, $y3$), then *Line1*, which is the mid-perpendicular of line A_2A_3, can be expressed as:

$$y-\frac{y_2+y_3}{2}=(x-\frac{x_2+x_3}{2})\frac{x_2-x_3}{y_3-y_2} \tag{7}$$

and *Line2*, which is the mid-perpendicular of line A_1A_3, can be expressed as:

$$y-\frac{y_1+y_3}{2}=(x-\frac{x_1+x_3}{2})\frac{x_1-x_3}{y_3-y_1} \tag{8}$$

The cross point *P* of the above two mid-perpendiculars, can then be calculated as displayed by equation (9). The centre of the overlapping region, *P* finally becomes *N*'s estimated position, and is the result of our proposed method. That is why our method is called Mid-Perpendicular.

$$\begin{cases} x_p=\dfrac{(x_1^2-x_2^2)(y_3-y_1)+(x_1^2-x_3^2)(y_1-y_2)+(y_1-y_2)(y_2-y_3)(y_3-y_1)}{2\left[y_1(x_2-x_3)+y_2(x_3-x_1)+y_3(x_1-x_2)\right]} \\ y_p=\dfrac{(y_1^2-y_2^2)(x_3-x_1)+(y_1^2-y_3^2)(x_1-x_2)+(x_1-x_2)(x_2-x_3)(x_3-x_1)}{2\left[x_1(y_2-y_3)+x_2(y_3-y_1)+x_3(y_1-y_2)\right]} \end{cases} \tag{9}$$

The process of our Mid-Perpendicular is summarized as follows. First, any normal node *N* that can be considered as a class 1 node sends a localization request to its neighbour anchors A_1 A_2 A_3. As a response, A_1 A_2 and A_3 send their respective positions to *N*. Finally, *N* calculates its estimated position *P* using equation (9).

The exceptional case is: when the triangle formed by three neighbor anchors is an obtuse triangle (that is, one angle of the triangle is greater than 90°), then we choose the middle of the longest side of the triangle as the estimated position. Because in this case, the overlapping communication region by three neighbor anchors is the same as the overlapping region by two of the three anchors, which forms the longest side of the triangle.

4. Algorithm Complexity Estimates

In this section, we analyze the computation complexity of the localization algorithms. The algorithms considered in this section include Centroid, CPE, DV-hop, our proposed Mid-Perpendicular and Checkout DV-hop.

The study of an algorithm's complexity involves determining the amount of resources (such as time and storage) necessary to execute it. Theoretically, it is commonly expressed using "O" notation, which suppresses multiplicative constants and lower order terms [14]. For example, if the number of elementary operations required by an algorithm on all inputs of size m is at most $5m^3 + 3m$, then its calculation complexity is $O(m^3)$. The following is the detailed analysis of calculation complexity for the related algorithms.

4.1 Centroid Complexity

We find that the computation for $x_{centroid}$ involves two elementary operations: "+" and "/". The number of "+" operation is $m-1$, and the number of "/" operation is 1. For $y_{centroid}$, the same result can be obtained. So, the total amount of elementary operations for Centroid is $2\times m$. Finally, we can conclude that the calculation complexity for Centroid is $O(m)$.

4.2 CPE Complexity

From equation (7), we find that the computation for x_{CPE} demands three elementary operations: compare, "+" and "/". In order to get $\max_{i=1}^{m}(x_i)$ and $\min_{i=1}^{m}(x_i)$, we first compare x_1 and x_2, and without loss of generality, assume that $x_1 > x_2$. We set the current maximum value *"max"* as x_1, and the current minimum value *"min"* as x_2. Then, for each x_i from x_3 to x_m, we first compare x_i to *"max"*. If x_i is greater than the current value of *"max"*, then x_i is assigned to *"max"*. Otherwise, another comparison will be performed with *"min"*. So, for each x_i from x_3 to x_m, 1 or 2 compare operations are needed, and the average number is 3/2. As a result, in order to obtain x_{CPE}, the number of compare operations should be $1+3/2\times(m-2)=3/2\times m-2$. The number of "+" operation is *1* as well as that of "/" operation. For y_{CPE}, the same result can be obtained. So, the total amount of elementary operations for CPE is $2\times3/2\times m=3\times m$. Finally, we can conclude that the calculation complexity for CPE is $O(m)$, which is similar to the results for Centroid.

4.3 Mid-Perpendicular Complexity

When m is 3, the calculation of our proposed Mid-Perpendicular method is shown as equation (9).

When m>3, from these m neighbor anchors $A_1, A_2, \ldots, A_m$, we select any three of them (for example, A_i, A_j, A_k) to form a 3-anchor group $\{ A_i, A_j, A_k \}$ (any two groups shouldn't be the

same). For each group, we use equation (8) to calculate its estimated position. The average of all these positions is the final estimated position for *N*. Since equation (9) has a constant amount of operations, the complexity of Mid-perpendicular method is dominated by the selection of all the 3-anchor groups.

The selection can be fulfilled in three steps. First, we sequentially select one anchor among the total *m* anchors, which brings in the complexity of O(m). Second, among the *m-1* anchors, we continue to select one anchor, with the complexity of O(m-1). Third, among the *m-2* anchors, the selection of the last anchor brings in the complexity of O(m-2). As a result, the total complexity is O(m×(m-1)×(m-2))= O(m^3).

4.4 DV-hop Complexity

[9][10] give the equation to calculate the estimated position of DV-hop algorithm, $N_{\text{DV-hop}}$ (*x*', *y*'). That is:

$$N_{DV-hop}: \begin{bmatrix} x' \\ y' \end{bmatrix} = (A^T A)^{-1} A^T B \tag{10}$$

where

$$A = -2 \times \begin{bmatrix} x_1 - x_m & y_1 - y_m \\ x_2 - x_m & y_2 - y_m \\ \vdots & \vdots \\ x_{m-1} - x_m & y_{m-1} - y_m \end{bmatrix} \tag{11}$$

$$B = \begin{bmatrix} d_1^2 - d_m^2 - x_1^2 + x_m^2 - y_1^2 + y_m^2 \\ d_2^2 - d_m^2 - x_2^2 + x_m^2 - y_2^2 + y_m^2 \\ \vdots \\ d_{m-1}^2 - d_m^2 - x_{m-1}^2 + x_m^2 - y_{m-1}^2 + y_m^2 \end{bmatrix} \tag{12}$$

where *di* is the estimated distance between N_x and A_i.

The amount of elementary operations ("—" and "×") in matrix *A* is *2×2×(m-1) =4×(m-1).* The amount of elementary operations ("+", "—" and "×") in matrix *B* is *3×m+5×(m-1) =8×m-5.* *A* is a *(m-1)* by *2* matrix, and A^T is a *2* by *(m-1)* matrix. The multiplication of these two matrix, $A^T A$ demands *4×[(m-1) + (m-2)] = 8×m-12* elementary operations ("×" and "+"). Since $A^T A$ is a *2* by *2* matrix, its inverse $(A^T A)^{-1}$ only needs 9 elementary operations ("/" and "—"). Then the multiplication of $(A^T A)^{-1}$ and A^T needs *2×(m-1)×3=6×(m-1)* elementary operations ("×" and "+"). The multiplication of $(A^T A)^{-1} A^T$ and *B* needs *2×[(m-1) + (m-2)] = 4×m-6*

elementary operations ("×" and "+"). As a result, equation (10) totally demands $30 \times m - 24$ elementary operations ("+", "—", "×", and "/"). So, the calculation complexity for DV-hop is O(m).

4.5 Checkout DV-hop Complexity

While traditional DV-hop algorithm has three steps, our proposed Checkout DV-hop method adds the fourth step. Equation (6) indicates that the additional step has a constant amount of operations. So, totally, the calculation complexity for Checkout DV-hop is O(m), the same as DV-hop.

5. Simulation and Analysis

Using Matlab simulation tools, we set two simulation scenarios with different area sizes. Scenario 1 consists of a $50 \times 50 m^2$ area with 100 sensor nodes in it, and the communication range of sensor nodes is 10 meters. As for Scenario 2, the size of area is $200 \times 200 m^2$, number of sensor nodes is 500, and the radio range is 50m. For both these two scenarios, all the sensor nodes are uniform-randomly distributed inside the simulation area.

The simulation sets different anchor ratios from 5% to 90%. The anchor ratio is defined as the ratio of anchors among the network nodes. For each anchor ratio, the simulations are performed 5000 times. Every time, the anchors are randomly selected from the network nodes.

5.1 Localization Accuracy Simulation

We consider two metrics for evaluating the performance of localization accuracy. One is location error (expressed in percentage of radio range), which is defined as distance(estimated position, true position)/RadioRange×100%. Here, distance(estimated position, true position) is the distance between a node's estimated position and its true position. RadioRange is the communication range of the sensor nodes. Another metric is the number of cases when the estimated position is so close to the true position that their distance is less than 10%× RadioRange. This metric, denoted as "number of correct cases" in short, can be regarded as a kind of standard deviation, and can evaluate the precision of localization.

In order to evaluate the localization performance, we simulated five algorithms: our algorithm and four existing algorithms, which are DV-hop[9], Improved DV-hop[10], Centroid+DVhop, CPE+DVhop. Here, Centroid+DVhop means using Centroid for class two nodes and DV-hop for class one nodes; this configuration is necessary since Centroid was not developed to work for class 1 nodes.

Figure 4 shows that our algorithm achieves better accuracy than Centroid, CPE, and DV-Hop. The location error decreases as anchor ratio increases. For the same anchor ratio, location error is smaller when our scheme is applied in same simulation environment. For example, when

anchor ratio is 5%, our scheme has an average location error of about 86%, whereas others have the error of about 97%. In this condition, since most normal nodes belong to class 1, the improvement is mainly fulfilled by Checkout DV-hop method. When anchor ratio is 90%, our scheme has an average location error of about 30%, whereas others have more than 36%. In this context, since most nodes belong to class 1, the improvement is mainly fulfilled by Mid-Perpendicular method. Thus, the proposed localization system adapts itself to the context configuration.

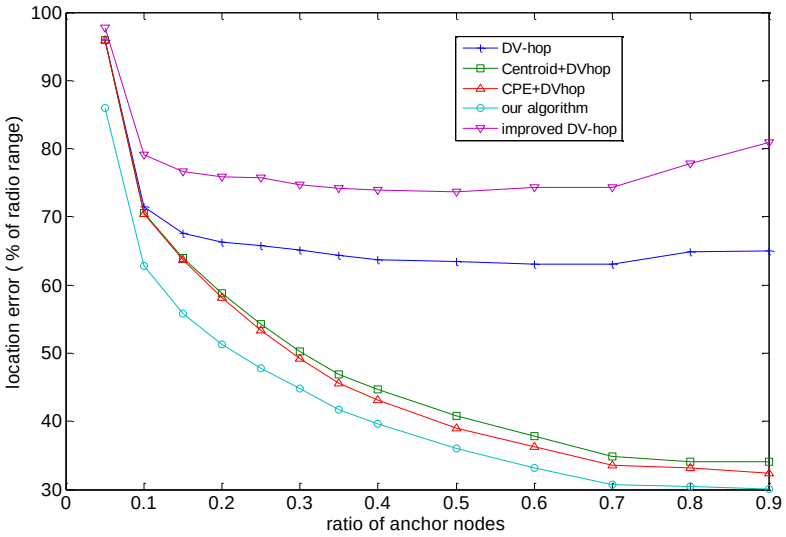

Figure 4. Location Error in Simulation Scenario 1

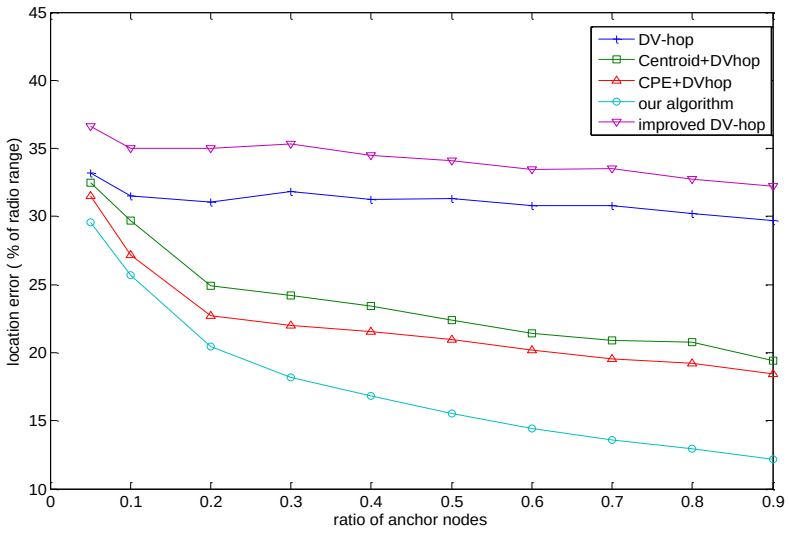

Figure 5. Location Error in Simulation Scenario 2

The following tables list the number of cases when the estimated position is close to the true position, for different algorithms in different scenarios. Since the communication range of nodes in Scenario 1 is 10 meters, the correct cases are those when the distance between estimated position and true position is less than 1 meter. From Table 1, it can be seen that our algorithm has the most correct cases. This shows that our algorithm has better precision than Centroid, CPE and DV-hop algorithms. The same conclusion can be obtained from Table 2 which includes the simulation results of Scenario 2. Since the communication range in Scenario 2 is 50

meters, the correct cases in Table 2 are those when the distance between estimated position and true position is less than 5 meter.

Table 1: Comparison of the Number of Correct Cases for Scenario 1

anchor ratio	Centroid+DV-hop	CPE+DV-hop	DV-hop	Improved DV-hop	our algorithm
5%	9402	9577	9451	9285	10361
20%	16710	16898	15407	13663	19026
40%	17561	17952	15253	12898	19511
60%	25980	25616	18049	12396	28058
80%	22017	23162	16946	14663	25587

Table 2: Comparison of the Number of Correct Cases for Scenario 2

anchor ratio	Centroid+DV-hop	CPE+DV-hop	DV-hop	Improved DV-hop	our algorithm
5%	93170	93518	92213	92058	99471
20%	123705	123112	110487	105739	128623
40%	154288	155906	127633	109046	175128
60%	159169	162956	113503	111478	193272
80%	145283	146754	128651	110629	185974

5.2 Computation Complexity Simulation

The computation of an algorithm takes certain amount of runtime when it is simulated with Matlab on computers. We use this runtime as the metric for evaluating the computation complexity. Hence, the longer the runtime of an algorithm, the higher its complexity.

Generally, sensors have limited computation capability, while the computers that we used for simulation possess high-speed powerful CPUs. The computation capacity of device has influence on the runtime of algorithm. In order to present this influence, the same simulation is done by two computers, which have different computation strength. Computer A has a 3.07GHz CPU and 24GB RAM, while computer B has a 1.6GHz CPU and 0.99GB RAM. In Matlab, the default data type is double-precision floating point, which requires 64 bits for storage. The simulation results are shown in the following figures.

Figure 6 shows the comparison on the runtime of DV-hop and our proposed Checkout DV-hop. We can find that these two algorithms have nearly the same computation complexity. That means, the additional complexity brought in by our proposed method can be neglected. Figure 7 presents the comparison on the runtime of Centroid, CPE and our proposed Mid-perpendicular.

When anchor ratio increases, the complexity of the proposed method increases obviously. The influence of device computation capacity can also be observed from both figures.

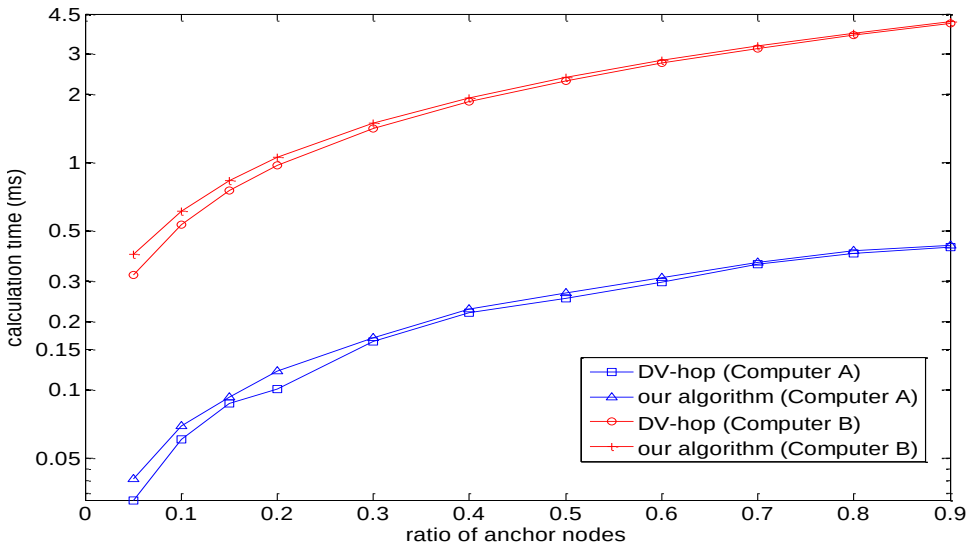

Figure 6. Runtime of Class One Algorithms in Simulation Scenario 1

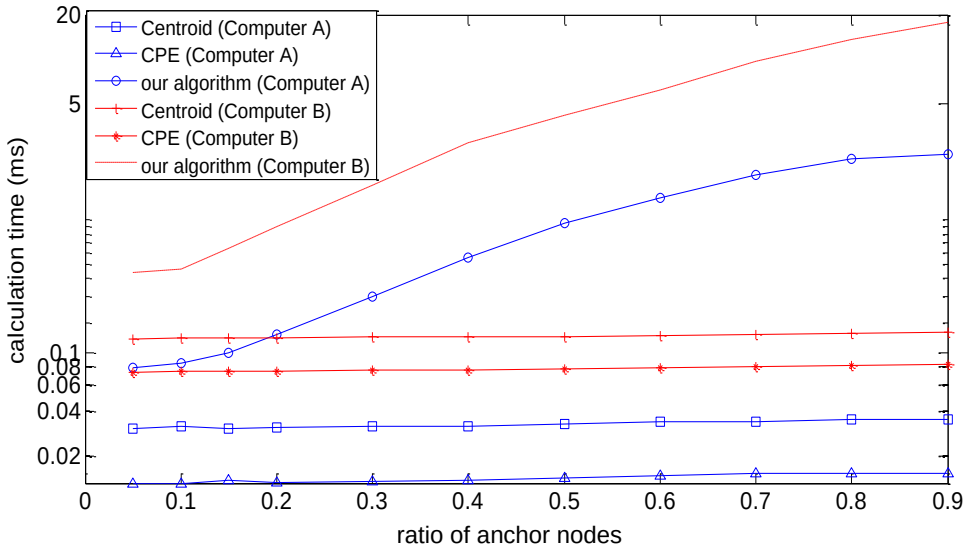

Figure 7. Runtime of Class Two Algorithms in Simulation Scenario 1

As shown in Table 3, the theoretical analysis in section 4 is compared with simulation results in this section. The theoretical analysis includes both big O notation and the total amount of operations, while simulation results are presented as runtime in millisecond. From the table, we can conclude that the theoretical analysis fits well with simulation results.

Table 3: Comparison of Theoretical Analysis and Simulation results

	Theoretical Analysis	Runtime* (ms)
Centroid	O(m): 2×m	0.128
CPE	O(m): 3×m	0.0766
Mid-perpendicular	$O(m^3)$	1.374
DV-hop	O(m): 30×m-24	1.4112
Checkout DV-hop	O(m): 30×m-12	1.4886

* : The simulation is fulfilled by Computer B, and the anchor ratio is 30%.

6. Conclusion

In this paper, we present a novel range-free localization algorithm for wireless sensor networks. Our algorithm works with two classes where the class one with less than three neighbor anchors and the class two with at least three neighbor anchors. Checkout DV-hop method is proposed for class one nodes, while Mid-Perpendicular method is proposed for class two nodes. Simulation results confirm that our proposed algorithms have better accuracy than the existing ones (Centroid, CPE and DV-hop). We can conclude that our two class algorithms could be a good candidate in improving the location accuracy for any ratio of anchor nodes. In addition, we analyze the computation complexity of the related algorithms. Both the theoretical analysis and simulation results prove that our Checkout DV-hop has a negligible increase in complexity, while Mid-Perpendicular has an acceptable increase.

References

[1] Martusevicius V., Kazanavicius E., “Self-localization System for Wireless Sensor Network”. ELEKTRONIKA IR ELEKTROTECHNIKA, vol:7, issue:103, pages:17-20, 2010.

[2] Kumar P., Reddy L., Varma S., “Distance measurement and error estimation scheme for RSSI based localization in Wireless Sensor Networks”, Fifth IEEE Conference on Wireless Communication and Sensor Networks (WCSN), Allahabad, India, Dec. 2009, pages:1-4. http://dx.doi.org/10.1109/WCSN.2009.5434802

[3] Van N. A., Wyffels J., JP G., “Time of Arrival Based on Chirp Pulses as a means to Perform Localization in IEEE 802.15.4a Wireless Sensor Networks”, ADVANCES IN ELECTRICAL AND COMPUTER ENGINEERING, vol:10, issue:2, pages: 65-70, 2010. http://dx.doi.org/10.4316/AECE.2010.02011

[4] Kovavisaruch L., Ho K.C., “Alternate source and receiver location estimation using TDOA with receiver position uncertainties”, IEEE International Conference on Acoustics, Speech, and Signal Processing 2005 (ICASSP'05), pages: iv/1065 - iv/1068, March 2005. http://dx.doi.org/10.1109/ICASSP.2005.1416196

[5] Rong P., Sichitiu M.L., “Angle of Arrival Localization for Wireless Sensor Networks”, 3rd Annual IEEE Communications Society on Sensor and Ad Hoc Communications and Networks 2006 (SECON '06), vol:1, pages: 374 - 382, Sept. 2006.
http://dx.doi.org/10.1109/SAHCN.2006.288442

[6] Bulusu N., Heidemann J., Estrin D., "GPS-less Low-Cost Outdoor Localization for Very Small Devices", IEEE Personal Communications, vol: 7, issue: 5, pages: 28-34, 2000. http://dx.doi.org/10.1109/98.878533

[7] Patro R.K., "Localization in wireless sensor network with mobile beacons", Proceedings of 23rd IEEE Convention of Electrical and Electronics Engineers in Israel, pages:22-24, Sept. 2004. http://dx.doi.org/10.1109/EEEI.2004.1361078

[8] Essoloh M., Richard C., Snoussi H., "Anchor-based distributed localization in wireless sensor networks", 14th IEEE/SP Workshop on Statistical Signal Processing, vol:1, pages:393-397, Aug 26-29, 2007. http://dx.doi.org/10.1109/10.1109/SSP.2007.4301287

[9] Niculescu D., Nath B., "Ad hoc positioning system (APS)", Proc. Global Telecomm. Conf., vol:5, pages:2926-2931, Nov. 2001. http://dx.doi.org/10.1109/GLOCOM.2001.965964

[10] Chen H., Sezaki K., Deng P., Hing C., "An improved DV-hop localization algorithm for wireless sensor networks", 3rd IEEE Conference on Industrial Electronics and Applications, pages:1557-1561, June 2008. http://dx.doi.org/10.1109/ICIEA.2008.4582780

[11] Lee J., Chung W., Kim E., Hong I., "Robust DV-hop Algorithm for Localization in Wireless Sensor Network", International Conference on Control Automation and Systems (ICCAS), Korea, pages: 2506 – 2509, October 2010.

[12] Linqing G., Anne W., Thierry V., "A Two-level Range-free Localization Algorithm for Wireless Sensor Networks", 6th international conference on wireless communications, networking, and mobile computing (WICOM 2010), Chengdu, China, pages: 1-4, Sept 2010. http://dx.doi.org/10.1109/WICOM.2010.5601377

[13] Sheu J.P., Chen P.C., Hsu C.S., "A distributed localization scheme for wireless sensor networks with improved grid-scan and vector-based refinement", IEEE Transsactions on Mobile Computing, vol:7, issue:9, pp:1110-1123, 2008. http://dx.doi.org/10.1109/TMC.2008.35

[14] Sanjeev A., Boaz B., "Computational Complexity: A Modern Approach". Cambridge University Press, 2009.