An efficient approach for trilateration in 3D positioning

E. Doukhnitch\textsuperscript{a,1}, M. Salamah\textsuperscript{a},*, E. Ozen\textsuperscript{b,2}

\textsuperscript{a}Department of Computer Engineering, Eastern Mediterranean University, Gazimagusa, K.K.T.C., Mersin 10, Turkey
\textsuperscript{b}Department of Information Technology, School of Computing and Technology, Eastern Mediterranean University, Gazimagusa, K.K.T.C., Mersin 10, Turkey

\textbf{A R T I C L E I N F O}

Article history:
Received 3 January 2008
Received in revised form 21 August 2008
Accepted 22 August 2008
Available online xxxx

Keywords:
Trilateration
3D location determination
Time of arrival (TOA)
Vector rotation

\textbf{A B S T R A C T}

Trilateration is a common operation to find the object location using its distances or range measurements to three other known points or stations. Traditionally, this problem has been solved either by algebraic or numerical methods. These methods involve long and complex geometric computations which are usually implemented in software. An approach that avoids this complexity is proposed here. Simulation results show improvements of the proposed approach in terms of computational cost and implementation simplicity over the conventional methods used. In addition, the proposed approach is based on vector rotations and uses only simple add and shift operations and therefore can be easily implemented in the hardware (or firmware) of the mobile object.

© 2008 Elsevier B.V. All rights reserved.

\section{1. Introduction}

Mobile positioning has recently become popular and many of the existing location determination technologies have been applied to cellular wireless networks. Value-added services (tracking, finding, guidance and navigation) also make positioning applications very attractive for the market\cite{1}. The problem gained more importance when the Federal Communications Commission asked for estimates of vertical position which requires a 3D positioning solution that is usually achieved by using trilateration. In fact, trilateration is a method to determine the position of an object based on simultaneous range measurements from three stations located at known sites using the time of arrival (TOA) signaling approach. This method is a common operation used in robot localization, kinematics, aeronautics, crystallography, and computer graphics\cite{2}.

The main contribution of this paper is twofold. We derived a simple 3D trilateration algorithm and this algorithm may be implemented as software or hardware.

The conventional geometric algorithm\cite{1,3} uses standard trigonometric complex computation methods that usually need a relatively long execution time. Moreover, it is obvious that a simpler and faster algorithm always reduces the complexity in terms of time, chip area, and power consumption\cite{4}. This improves the quality of automatic control of a mobile object. Motivated with these arguments, we proposed an efficient algorithm for location determination in 3D space. Since it is possible to use the algorithm in hardware implementation for example, which makes it cheap, it could be used for 3D sensor networks, as described by Ou and Ssu\cite{5}, along with many other low power wireless applications.

The paper is organized as follows. Section 2 presents the related work and the conventional geometric algorithm used for 3D location determination. We suggest a simplified geometric algorithm in Section 3. To implement this algorithm, our proposed approach which is based on vector rotation is given in Section 4. The same section also includes different ways of implementing our fast and dynamic methods. The convergence and error analysis of the proposed approach are explained in Section 5. Simulation results and the achieved improvements are shown in Section 6. Finally, Section 7 summarizes the main contributions.

\section{2. Related work}

Trilateration can be trivially expressed as the problem of finding the intersection of three spheres which involves a system of quadratic equations. There are many algebraic and numerical methods to solve this problem in 2D\cite{6–18} and in 3D space\cite{1,2,19–23}. Most (if not all) of these approaches use software-oriented complex geometric methods that usually need relatively long execution time and they are not suitable for simple hardware (or firmware) implementation. The recent work of Thomas and Ros\cite{2} represents an alternative approach in order to avoid the algebraization of the problem using a few number of Cayley–Meger determinants which is considered as the computationally most efficient. However, it is not suitable for simple implementation as well.
Generally, the mobile object (MO) location can be estimated using a statistical approach [like least-square (LS) method] or geometric approach. The LS solution obtained by iterative algorithm could converge to one of the local minima instead of the global minima, and may not be suitable for hardware implementation [1]. For the geometric approach, the problem is usually simplified by expressing the station coordinates according to a specific coordinate frame. This can be achieved by using the XY plane of the reference frame as the base plane, or making one coordinate axis coincide with the baseline between two stations, and locating the origin at one station [24]. In this case, for 3D space we can generalize a simple hardware-oriented approach as in Salamah and Doukhnitch’s [4] where a new computational method was adopted using simple add-shift operations to find the position of a mobile object in 2D space.

Though we have used a similar approach [4], it was for a 2D trilateration system. To solve a 3D trilateration problem with the greater number of possible variants of conditions to control this process as will be clear from Section 4.1. In order to improve further the performance of the proposed algorithm, we apply parallel vector rotation together with a dynamically changing rotation angle. In contrast to Doukhnitch and Salamah [25] we customized this idea for a 3D case.

2.1. The conventional geometric algorithm

Usually the conventional positioning algorithm uses four base stations (BS-s) to determine the mobile location [1,3]. Suppose we have four BS-s with known locations where these stations are not in the same plane (can always provide a solution) and a MO located in the range of these BS-s. From these four BS-s it is necessary to form three spheres in order to estimate the location of MO [3]. The MO located at \( x_m, y_m, z_m \), BS-s located at \( x_i, y_i, z_i \) and the distance from BS-s to MO is \( d_i \) where \( i = 1,2,3,4 \). The intersection of spheres where the sphere of location is determined using range measurements [1] in the 3D Euclidian space is the location of MO. The intersection of two of these spheres will form a circle of location. Using these circles of locations, planes of location can be obtained. So, we will have independent linear equations of the planes of locations [1] as

\[
p = G^{-1}h,
\]

where \( p = [x_m, y_m, z_m]^T \),

\[
G = \begin{bmatrix}
  x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\
  x_4 - x_1 & y_4 - y_1 & z_4 - z_1 \\
  x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \\
\end{bmatrix}
\]

and

\[
h = \frac{1}{2} \begin{bmatrix}
  x_1^2 + y_1^2 + z_1^2 - x_2^2 - y_2^2 - z_2^2 + d_1^2 \\
  x_1^2 + y_1^2 + z_1^2 - x_4^2 - y_4^2 - z_4^2 + d_4^2 \\
  x_1^2 + y_1^2 + z_1^2 - x_3^2 - y_3^2 - z_3^2 + d_3^2 \\
\end{bmatrix}
\]

If \( |G| \) which depends on the location of BS-s is not equal to zero, then Eq. (1) has a real root and an estimation location could be calculated [3]. In order to simplify this solution, we can use a special coordinate system [24].

3. A simplified geometric algorithm

In Fang’s ‘simple solution for hyperbolic and related position fixes’ [24], the positioning algorithm uses three base stations to determine the mobile location in a special coordinate system. Assume that the coordinates of BS1, BS2 and BS3 in a local right-handed orthogonal coordinate system are \( (x_1, y_1, z_1) \), \( (x_2, y_2, z_2) \) and \( (x_3, y_3, z_3) \), respectively. The origin is at one of the stations, one axis is along a station baseline, and another axis is orthogonal to the two station baseline, or station plane. Therefore according to the TOA from each of the three base stations, the MO position \( (x_p, y_p, z_p) \) is the intersection of the three spheres centered at \( BS_1(0,0,0), BS_2(x_2,0,0) \) and \( BS_3(x_3,y_3,0) \) with radiuses \( d_1, d_2, d_3 \), respectively. The conventional algorithm for 3D positioning can be decomposed into a sequence of 2D rotations as illustrated in Fig. 1 below.

To illustrate the algorithm, Fig. 2 shows a detailed drawing. Using the formula of a circle on the plate XOY, the equation for \( BS_1 \) is

\[
(X - X_1)^2 + (Y - Y_1)^2 = d_1^2,
\]

and the equation for \( BS_2 \) is

\[
(X - X_2)^2 + (Y - Y_2)^2 = d_2^2.
\]

Since \( Y_1 = Y_2 \), Eq. (5) can be rewritten as

\[
(X - X_2)^2 + (Y - Y_1)^2 = d_2^2.
\]

Combining (4) and (6), we can find the X coordinates of the intersection points as

\[
X = \frac{d_2^2 - d_1^2 + X_2^2 - X_1^2}{2(X_1 - X_2)}
\]

Substituting (7) in (6), the Y coordinates of the intersection points will be

\[
Y = \pm \sqrt{d_2^2 - X_1^2 - X_2^2 + 2XX_1 + Y_1}.
\]

So, the results of the first step of the algorithm are: \( X_0 = X \) and radius of the circle of two spheres intersection \( R_1 = Y \).

For the second step, we can calculate the projection of vector \( d_3 \) on the circle plane: \( R_2 = \sqrt{d_3^2 - (X_3 - X_p)^2} \).

The third step is used to calculate the coordinates of two intersection points for two circles with radiuses \( R_1 \) and \( R_2 \):

\[
Y_p = \frac{(R_1^2 - R_2^2 + Y_3^2)/2Y_3}{Z_p = \pm \sqrt{R_1^2 - Y_p^2}}.
\]

The twofold ambiguity of \( Z_p \) in (9) can often be resolved if some knowledge of the general location of the MO (like altitude above station plane) is available, and hence the MO position \( P \) can be estimated [24].

![Fig. 1. The simplified algorithm for 3D positioning.](image-url)
4. The proposed approach

Our rotation method with fixed angle (ODS) algorithm is based on TOA and it uses the same source of information and steps as the simplified algorithm in Fig. 4. However, the steps are performed in a completely different way. The main idea is to use vectors rotations [4].

4.1. Rotation method with fixed angle (ODS)

Rotations in the proposed algorithm can be performed using either the known CORDIC algorithm [26], or the fast rotation methods [27]. As it is shown in Salamah and Doukhnitch’s algorithm for mobile objects localization [4], rotations with CORDIC algorithm have low performance due to the need of multiple iterations and factor compensation for each of the iterations. The fast vector rotation method uses a fixed step rotation angle \( \sigma = \arcsin 2^{-k} \), where \( k > 4 \) depends on the needed accuracy, and executes the rotation step-by-step recursively [27].

The rotation matrix for step \( j \) has the following approximated elements:

\[
M(j) = \begin{pmatrix}
\cos \sigma & \sin \sigma \\
-\sin \sigma & \cos \sigma
\end{pmatrix} = \begin{pmatrix}
1 - 2^{-2k+1} & 2^{-k} \\
-2^{-k} & 1 - 2^{-2k+1}
\end{pmatrix}
\]

Therefore, vector coordinates are recursively rotated at each step \( j \) as follows:

\[
V(j + 1) = M(j) \cdot V(j) = \begin{pmatrix}
x_{j+1} = x_j - x_{j,a} + s_j b \\
y_{j+1} = y_j - y_{j,a} - s_j b
\end{pmatrix}
\]

where \( a = 2^{-2k+1} \), \( s = \pm 1 \) the operator of rotation direction, and \( b = 2^{-k} \).

So, it is clearly seen from the above equations that no trigonometric calculations are performed. Instead, simple add, subtract and shift operations are used which reduce the computational cost and are necessary requirements for hardware implementation.

The first step of the algorithm in Fig. 1 is to find the intersection point of two circles with radiuses \( d_1 \) and \( d_2 \) and can be implemented by rotating vectors \( d_1 \) and \( d_2 \), until their heads intersect each other [4]. Before starting vector rotations, we have to check which radius is larger. The smaller radius needs more rotations than the other one. If we assume that \( B_3 \) has a larger radius (i.e. \( d_1 > d_2 \)), the conditions of rotation should be as follows:

\[
\begin{align*}
\text{While } & (x_{1,j} + x_{2,j} - X > \varepsilon) \vee (y_{1,j} - d_2 > \varepsilon) \\
\text{   Rotate } & d_1 \text{ by one step } j \text{ with angle } \sigma \\
\text{While } & (y_{1,j} - y_{2,j}) > \varepsilon \\
\text{   Rotate } & d_2 \text{ by one step with angle } \sigma \\
\text{End while }
\end{align*}
\]

where \( x_{1,0} = d_1 \) and \( y_{1,0} = 0; x_{2,0} = d_2 \) and \( y_{2,0} = 0; \varepsilon \) determines the machine accuracy.

The intersection point \( (x_{1,j}, y_{1,j}) \) can be found as a result of these rotations \( (s = 1) \).

The result of this first step is the positioning of the circle of location with coordinate \( X_0 = x_{1,1} \) and radius \( R_1 = y_{1,1} \).

To implement the second step of ODS algorithm we can rotate the vector \( (d_1,0)^t \) step-by-step (with step size \( \sigma \) and \( s = 1 \) until its \( x_1 \) coordinate becomes \( X_3 - X_0 \) and \( y_1 = R_2 \). That is, the conditions of rotations should be as follows:

\[
\begin{align*}
\text{While } & (x_{0} - (X_3 - X_0)) > \varepsilon \\
\text{   Rotate } & d_1 \text{ by one step } j \text{ with angle } \sigma \\
\text{End while }
\end{align*}
\]

The third step can be implemented with parallel vector rotation similar to the first step but on the plane YOZ \( (X = X_0) \) with different condition for the external While-loop in contrast to Salamah and Doukhnitch [4]. We rotate vector \( (R_1,0)^t \) and vector \( (R_2,0)^t \) up to the equality: \( a \) \( y_{1,i} + y_{2,i} - |Y_3| \) or \( b \) \( |y_{1,i} - y_{2,i}| - |Y_3| \). The process is illustrated in Fig. 3: (a) for stop condition (a) where \( |Y_3| > R_1 \) and (b) for stop condition (b) where \( |Y_3| < R_1 \). In both cases, vectors \( \mathbf{R}_1 \) and \( \mathbf{R}_2 \) (or even \( \mathbf{R}_2 \)) are rotated in the first quadrant only.
The longest vector is rotated first. Assuming that $R_1 > R_2$, the conditions of rotations should be as follows:

\[
\text{While } (y_{1j} + y_{2j} > |y_3|) \land (|y_{1j} - y_{2j}| < |y_3|) \land (z_{1j} - R_2 > e) \\
\quad \text{Rotate } R_1 \text{ by one step } j \text{ with angle } \sigma \\
\quad \text{While } (z_{1j} - z_{2j}) > e \\
\quad \quad \text{Rotate } R_2 \text{ by one step with angle } \sigma \\
\text{End while}
\]

The result of the third step is the following:

If stop condition (a) is true, then $Y_p = (\text{sign}(y_3))y_{1j}$.
If stop condition (b) is true, then $Y_p = (\text{sign}(y_{1j} - y_{2j})) (\text{sign}(y_3))y_{1j}$.
For both cases $Z_p = \pm z_{1j}$.

Finally, we can use some additional information to fix the coordinate $Z_p$ of the object [24].

For the purpose of decreasing the number of rotations, we modified the ODS algorithm. Unlike the ODS, where the step angle is fixed, the step angle may vary from one rotation to another in DODS depending on distance criteria.

4.2. Dynamic rotation method (DODS)

For the first step of the algorithm the angle for parallel rotations of the two vectors $(d_1, 0)^T$ and $(d_2, 0)^T$ depends on the distance between their current heads’ positions. If the distance is large, rotation is done with a bigger step angle, i.e. $k$ is small. As the distance gets smaller, the two vectors are rotated with smaller step angles. To guarantee convergence, the approximation of $a$, in each iteration $j$ of (11), is taken as follows:

\[
a = \sin(\sigma_j) = \begin{cases} 
2^{-k-1} & \text{if } \Delta x_j > 2^{-k} \\
2^{-m-1} & \text{if } 2^{-m} \leq \Delta x_j < 2^{-m-1}, \quad k < m \leq n, 
\end{cases}
\]  

(15)

where $\Delta x_j = x_{1j} + x_{2j} - x_j$, $k = 4$, $m = 5$, and $n = 11$ (see Section 5).

Although only $a$ is shown in (15), $b$ is changed accordingly.

Note that the distances are normalized, and the largest possible distance is taken as 1. Consequently, other distances are less than 1, thus $\Delta x_j < 1$.

To implement the second step of ODS algorithm we can rotate the vector $(d_3, 0)^T$ step by step (with step size $s$ and $s = 1$) until its $x_i$ coordinate becomes $(X_1 - X_0)$ and coordinate $y_i = R_3$. As the $x$ coordinate of the vector decreases, rotation with a smaller step angle is performed. Approximation of $a$ in each rotation of (15) is chosen as follows:

\[
a = \sin(\sigma_j) = 2^{-k} \quad \text{if } x_j - x_1 + x_p \geq 2^{-k} \quad (16)
\]

for $k = 5, 6, \ldots, 11, b$ is changed accordingly.

The third step is implemented as the first step using the condition (15) with $\Delta x_j = \text{Min}[(y_{1j} + y_{2j} - Y_3), (y_j + y_{1j} - y_{2j})]$.

5. Convergence and quantization effects of the proposed algorithm

Convergence of our algorithms is guaranteed because of the following facts:

- Since the nearest three BS-s are used for the MO positioning, the largest radius of a rotation circle is always less than the distance between BS-s.
- In the parallel rotation of vectors we rotate the larger one first.
- The parallel rotation of vectors is always performed in the first quadrant only.
- To avoid the infinite internal While-loop in (12), the external While-loop stops when the vertical coordinate of the larger vector exceeds the length of the small one.

To prove the convergence of our algorithms, we can prove the convergence of its three steps separately.

**Theorem 1.** For algorithm (12) if $d_1 > d_2$

\[
d_1 \sin \left( \sum_{j=0}^{i-1} \sigma_j \right) = d_2 \sin \left( \sum_{j=0}^{i-1} \sigma_j \right) G_i \sigma_j,
\]

(17)

where $\sigma_j = \arcsin(2^{-k})$, $t = x_j / \sigma_j$ (see Fig. 2) and

\[
G_j = \left( (2x_{j+1} - x_j) / \sigma_j \right)
\]

\[
= \left( \arcsin \left( \frac{d_1 \sin(x_{j+1} + \sigma_j)}{d_2} \right) - \arcsin \left( \frac{d_1 \sin(x_j)}{d_2} \right) \right) / \sigma_j.
\]

(18)

**Proof.** Theorem can be proven by mathematical induction. Obviously, it is true for $t = 0$. Assume it is true for $t = r$. It follows from (12) that

\[
y_{t,r+1} = y_{t,r} \cos \sigma_r + x_{t,r} \sin \sigma_r - y_{2r+1}
\]

\[
y_{t,r+1} = y_{t,r} \cos(G_r \sigma_r) + x_{t,r} \sin(G_r \sigma_r)
\]

(19)

or

\[
d_1 \left( \sin \sum_{j=0}^{i-1} \sigma_j \right) \cos \sigma_i + d_1 \left( \cos \sum_{j=0}^{i-1} \sigma_j \right) \sin \sigma_i
\]

\[
d_1 \left( \sin \sum_{j=0}^{i-1} G_i \sigma_j \right) \cos(G_r \sigma_r) + d_1 \left( \cos \sum_{j=0}^{i-1} G_i \sigma_j \right) \sin(G_r \sigma_r)
\]

Please cite this article in press as: E. Doukhnitch et al., An efficient approach for trilateration in 3D positioning, Comput. Commun. (2008), doi:10.1016/j.comcom.2008.08.019
or
\[ d_1 \sin \sum_{j=0}^{r} \sigma_j = d_2 \sin \sum_{j=0}^{r} G_j \sigma_j. \]

So, from mathematical induction the theorem holds for any integer \( t \).

Algorithm (13) is a simple rotation of a single vector and the algorithm (14) is similar to (12) with a different coordinate system.

The only difference between ODS and DODS algorithms is that in DODS the value of \( \sigma_j \) varies from step to step (see (15) and (16)).

Quantization effects of the ODS and DODS algorithms: Following Hu’s ‘the quantization effects of the CORDIC algorithm’ [28], we can analyze two types of quantization error: an approximation error due to the quantized representation of rotation angle \( \sigma = \arcsin 2^{-k} \) and a rounding error due to the finite precision representation in fixed point arithmetic.

For a cellular system the total rotated angle may be bounded as
\[ |A| \leq \frac{\pi}{3}. \quad (20) \]

Its angle approximation error \( \delta \) will be bounded by the smallest rotation angle \( \sigma \). That is
\[ |\delta| \leq \sigma = \arcsin 2^{-k}. \quad (21) \]

If \( V_0(t) \) is the ideal result vector (11) of total rotation when \( \delta = 0 \), it can be computed as
\[ V_0(t) = D V(t) = \begin{pmatrix} \cos \delta & -\sin \delta \\ \sin \delta & \cos \delta \end{pmatrix} V(t), \quad (22) \]

where \( V(t) = [x(t), y(t)] \) is a vector with the final coordinates. Using \( I \) as \( 2 \times 2 \) identity matrix, we have
\[ V_0(t) - V(t) = (D - I) V(t). \quad (23) \]

Hence, the relative approximation error is bounded by [28]
\[ |V_0(t) - V(t)|/|V(t)| \leq \|D - I\|. \quad (24) \]

The spectral norm of matrix difference in (24) can be estimated as
\[ \|D - I\| = \sqrt{(\cos \delta - 1)^2 + \sin^2 \delta} = 2 \sin \frac{\delta}{2} \leq |\delta| \leq \sigma = \arcsin 2^{-k}. \quad (25) \]

As for DODS algorithm, the value of \( k \) varies in the range \( k = 4 \ldots 11 \) and the approximation error corresponds to the average value \( k = 7 \).

The rounding error can be described with a quantization operator \( Q[V(j)] \) [28] such that \( Q[V(j)] = V(j) + e(j) \), where \( e(j) = [e_x(j), e_y(j)] \) is an error due to rounding. For fixed point arithmetic, the absolute rounding error will be bounded by
\[ |e_x(j)| \leq \frac{\epsilon}{2} \quad \text{and} \quad |e_y(j)| \leq \epsilon, \quad (26) \]

where the magnitude of ‘\( \epsilon \)’ depends on the machine accuracy (for \( h \) bits accuracy \( \epsilon = 2^{-h-1} \)). Hence, an upper bound for \( |e(j)| \) is
\[ |e(j)| = \sqrt{e_x^2(j) + e_y^2(j)} \leq 2 \epsilon. \quad (27) \]

Taking into account two error components (the rounding error propagated from the previous rotation step and the new rounding error introduced in the present step) we can use the overall rounding error propagation formula from [28] as
\[ f(t) = Q[V(t)] - V(t) = e(t) + \sum_{j=0}^{r-1} [M(j), e(j)]. \quad (28) \]

And the worst case rounding error is equal to
\[ |f(t)| \leq \sqrt{2} \epsilon (1 + \sum_{j=0}^{r-1} |M(j)|). \quad (29) \]

The norm \( ||M(j)|| \) approaches unity and for cellular systems
\[ t_{\text{max}} \leq \pi/(3 \cdot 2^{-k}). \]

Hence, the worst case rounding error
\[ |f(t)| \leq \sqrt{2} \cdot 2^{-h-1} \pi/(3 \cdot 2^{-k}) = 2^{k-h-1} \pi \sqrt{2}/3. \quad (30) \]

As for DODS algorithm, the number of steps \( t \) is always less than for ODS and the rounding error is smaller.

To estimate the accuracy of our ODS and DODS algorithms with the assumption that \( d_1 \) is the larger radius, we can obtain the accuracy as follows:
\[ \max |\Delta y| = \max |d_1| \sin \sigma = \max |d_1| 2^{-k}. \quad (31) \]

where \( \Delta y \) is the increment of coordinate \( y \) in one step of rotation.

For example, a macro cell of a cellular system usually has a radius of 30 km, and if the required accuracy is approximately 15 m, we have
\[ 2^k = \max |d_1| \frac{|\Delta y|}{15} = 2 \times 10^4. \quad (32) \]

Therefore, \( k_{\text{max}} = n = \log_2(2 \times 10^4) \approx 11 \). From (30) with \( k = 11 \), we can find that \( h \geq 20 \). It is worth mentioning that \( k \) can be increased if larger precision or less computational error is needed.

6. Simulation results

In our analysis the Matlab 7.0 package was used. We wrote programs for conventional and our algorithms. We ran our programs to find the location of a mobile in the coverage area of three BSs. The experiments were repeated for many arbitrary positions of the mobile to guarantee the 95% confidence level.

For the purpose of comparing our algorithms against the conventional and simplified geometric algorithms, we calculated a computational cost as the number of required operations to find the location of the mobile object in each case. We used the weights of the operations as shown in Table 1 in terms of addition and shift for \( h = 20 \) bits accuracy [26].

Fig. 4 shows the average number of operations needed by each of the algorithms versus \( \sin \sigma \) (which explicitly specifies the step rotation angle \( \sigma \), and implicitly specifies the accuracy level). The computational cost of ODS increases as the step angle decreases (i.e. accuracy level increases), because more rotations are required with small angles. The DODS algorithm shows almost a constant performance, because the rotation starts with the largest possible angle and ends with the smallest one. DODS algorithm demonstrates the lowest computational cost. The improvements of the DODS algorithm over the conventional one and its simplified version in terms of computational cost are 67% and 33%, respectively.

Fig. 5 shows the average absolute error in determining the mobile location against \( \sin \sigma \). For ODS algorithm, the absolute error decreases when the step rotation angle decreases. For \( \sin \sigma = 2^{-11} \), a very good level of accuracy can be achieved. It is equal to approximately 65 m which coincides with the E-911 standards. The average error of DODS algorithm is constant since all rotations end with the minimal step size (\( \sin \sigma = 2^{-11} \)).

7. Conclusion

This paper presents a new simple DODS algorithm based on TOA measurements to determine the position of a mobile object.

<table>
<thead>
<tr>
<th>Table 1</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Weights of operations</strong></td>
</tr>
<tr>
<td>Operation</td>
</tr>
<tr>
<td>Weight</td>
</tr>
</tbody>
</table>

in 3D space of a wireless network. Although the 3D positioning algorithm proposed here is a generalization of the 2D one [25] that had been previously proposed by the same authors, the addition of the 3rd dimension raises difficulties such as finding the projection of third distance which we explicitly solved. The main contribution of this study could be achieved using the combination of three ideas: (1) We simplified the conventional 3D geometric algorithm (due to the special coordinate system); (2) To solve the trilateration problem, vector rotations were used. (3) To implement vector rotations, the simple fast rotation method [27] was used. The main benefit of the algorithm is that it avoids the calculations of complex arithmetic operations (multiplications and divisions) and trigonometric functions. Since all operations in our algorithm are simple add and shift operations, it reduces the computational cost by 67% and (if needed) the implementation in hardware can be achieved simply. We have analyzed all components of the computational error and we proved by mathematical induction the convergence of the algorithm. The algorithm will conceivably be efficient in terms of time, chip area, and power consumption (by inference). This algorithm is suitable for ASIC, FPGA implementation and for a RISC processor also. The results show that for an acceptable accuracy level satisfying E-911 standards, our algorithm outperforms the conventional one in terms of both software and hardware implementations.

References