Journal of Applied Technology and Innovation
vol. 1, no. 1, (2017), pp. 10-27

Essential Functions for Localization in Wireless Sensor Networks Using Geographic Coordinates

Thomas O'Daniel
Faculty of Computing, Engineering, & Technology
Asia Pacific University of Technology & Innovation
57000 Kuala Lumpur, Malaysia
dr.thomas.odaniel@apu.edu.my

Abstract- A variety of localization protocols have been proposed in the literature which allow Wireless Sensor Network (WSN) nodes to interpolate their location from their neighbors as an alternative to deploying more expensive WSN nodes with GPS receivers or other dedicated localization hardware. This paper presents a set of efficient functions applied to three base cases where a WSN node calculates an initial estimate of its location and a finite set of alternate points that could be its actual location, given the GPS coordinates and nominal transmission radius of two or three neighbors. The process of narrowing the set of possible actual locations through iterative refinement as more nodes join the network is discussed, along with the limits on the accuracy of the overall network map.

Index Terms- Wireless sensor network (WSN), global positioning system (GPS); localization protocols

1. Introduction

Wireless Sensor Networks (WSNs) are a fundamental aspect of ubiquitous systems and the Internet of Things (IoT). WSNs are composed of tiny devices with constrained processing and memory resources that are typically battery powered. Networks of these devices are characterized by small packet payload size, minimum bandwidth, unreliable radio connectivity, ad hoc deployment, dynamic topology changes, and nodes running in a power conservation mode to prolong battery lifetime.

Many industrial applications consist of a large number of randomly distributed nodes, so it is advantageous if the network is able to autonomously build the communication links and control the communication between nodes [1]. WSN deployments for environmental surveillance and disaster management in particular could benefit from constant reporting of the location where data was sensed. Nodes can be equipped with Global Positioning System (GPS), but this is a costly solution in terms of both money and energy consumption [2] [3], and GPS typically fails inside buildings and under heavy vegetative cover [4].

This paper describes the fundamental calculations necessary for a node to estimate its position given the GPS coordinates of some neighbors and an indication of their transmission radius. The most basic principle of triangulation is that given two points and the distance between them, a third point can be found. Ancient texts record the use of triangulation to estimate distances. Two common examples: to measure the distance from shore to a remote ship, mark two points on the shore with a known distance between them and calculate the angles between this baseline and the location of the ship; to measure the height of a mountain or lighthouse, use the distance between two ground points and the angles to the top.

In surveying *trilateration* is the process of determining absolute or relative locations of points purely by measurement of distances, while the term *triangulation* is reserved for the process that involves only angle measurements, The use of both angle and distance measurements is referred to as *triangulateration* by those who find these distinctions meaningful. *Multilateration* is a technique based on measuring power levels and antenna patterns, commonly used with radio navigation systems. Unlike measurements of absolute distance or angle, using a radio signal to measure the distance between two stations at known locations emitting broadcast signals at known times results in an infinite number of locations that satisfy the "time difference of arrival" metric. Multilateration requires at least three synchronized emitters for determining location in two dimensions, and at least four for three dimensions.

Many WSN localization techniques reported in the literature use various combinations of metrics to develop measures of link quality, but inferring relative location from these measures is subject to assumptions about decrease in signal strength due to the distance between transmitter and receiver, type and height of antennas, and the presence of obstacles that disrupt the line-of-sight path [1] [5] [6]. The techniques presented here simply require each node to have the ability to transmit its actual or presumed location, and its nominal transmission radius. Exactly how this is achieved (through beaconing, addressing, or some type of protocol for example) is not important for the calculations. The calculations are done with locations expressed as decimal GPS coordinates and distances in kilometers; other coordinate systems and distance measurements could be used.

The first section of this paper reviews the basic terms and concepts related to solving triangles and geolocation. The second section presents the essential formulae expressed as functions in the C programming language, which can be easily ported to another. The third section shows how the essential functions can be used by a WSN node to establish an initial estimate of its location given minimal information, along with a finite set of alternate points that could be its actual location. This is followed by an examination of the process for refining the initial estimate using several of the sets of alternate points, and consideration of the limits on overall accuracy.

2. Materials and Methods

2.1 Basic Principles of Triangulation

2.1.1 *Characteristics of Triangles*

Triangles have several interesting properties:

The shortest side is always opposite the smallest interior angle

- The longest side is always opposite the largest interior angle
- The interior angles of a triangle always add up to 180°
- The exterior angles of a triangle always add up to 360° - thus given three points, it is possible to draw a circle that passes through all three (the circumcircle of the triangle)
- Any side of a triangle is always shorter than the sum of the other two sides; in other words, a triangle cannot be constructed from three line segments if any of them is longer than the sum of the other two. This is known as the Triangle Inequality Theorem.

Solving a triangle means finding the unknown lengths and/or angles. The classic problem is to specify three of the six characteristics (3 sides, 3 angles) and determine the other three. Any combination except 3 angles allows determination of the other side lengths and angles - three angles alone determines the shape of the triangle, but not the size. The actual solution depends on the specific problem, but the same tools are always used:

- The knowledge that the sum of the angles of a triangle is 180º.
- The Pythagorean theorem, the essence of which is that for any triangle a line can be drawn that divides it into two right triangles, and the relationship between the sides of a right triangle is such that the square of the length of the longest side equals the sum of the squares of the lengths of the other two sides ($c^2 = a^2 + b^2$ in the notation explained below).
- The trigonometric functions that relate a given angle measure to a given side length.

Essential Terminology

It is usual to name each vertex (angle) of a triangle with a single upper-case letter, and name the sides with the lower-case letter corresponding to the opposite angle, as illustrated in Fig. 1. Alternatively, the sides of a triangle can be labeled for the vertices they join, so side b would be called line segment AC.

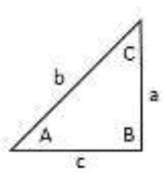


Fig. 1

The height or altitude of a triangle depends upon which side is selected as the base. An altitude of a triangle is a line through a vertex of a triangle that meets the opposite side at right angles. This point will be inside the triangle when the longest side is the base; if one of the angles opposite the chosen vertex is obtuse (greater than 90°), then this point will lie outside the triangle. The area of a triangle is one-half the product of its base and its perpendicular height; in the case of a right triangle, this is the product of the sides that form the right angle.

A special set of terms is used to describe right triangles: the hypotenuse is the longest side, an "opposite" side is the one across from a given angle, and an "adjacent" side is next to a given angle. There are six trigonometric functions that take an angle argument and return the ratio of two of the sides of a right triangle that contain that angle. For any given angle L

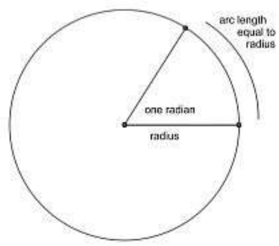
1. Sine: sin(L) = Opposite / Hypotenuse
2. Cosine: cos(L) = Adjacent / Hypotenuse
3. Tangent: tan(L) = Opposite / Adjacent

atan(), asin(), and acos() are the respective inverses of tan(), sin(), and cos().

In C, C++, Java, python, and other programming languages the trigonometric functions take a parameter and return a value expressed in radians. The radian is the standard unit of angular measure, used in many areas of mathematics. One radian is the angle at the center of a circle where the arc is equal in length to the radius, as illustrated in Fig. 2 (a). More generally, the magnitude in radians of an angle is the arc length divided by the radius of the circle. As the ratio of two lengths, the radian is a "pure number" that needs no unit symbol.

The number pi is a mathematical constant, the circumference divided by the diameter of any circle. One radian is equivalent to 180 / pi (57.29578) degrees; Fig. 2 (b) illustrates this relationship.

The trigonometric functions actually work with a "unit circle" centered at (0,0) with a radius of one unit, so it intersects the X and Y axes at (1,0), (0,1), (-1,0), and (0,-1). They return a value between 1 and -1, and multiplying this number by the length of the vector yields the exact Cartesian coordinates of the vector.



(a)

(b)

Fig. 2 [7] [8]

Solving Triangles

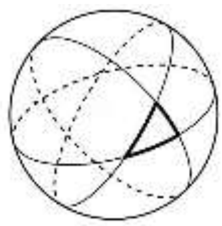
As noted above, solving a triangle means finding the unknown lengths and/or angles. Given any three of the six parameters (except 3 angles without a side length), any triangle can be solved using three equations:

4. $A + B + C = 180°$ [Angles sum to 180]
5. $c^2 = a^2 + b^2 - 2*a*b*\cos(C)$ [The Law of Cosines]
6. $a / \sin(A) = b / \sin(B) = c / \sin(C)$ [The Law of Sines]

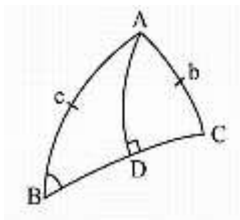
Points worthy of mention are (a) the Law of Cosines reduces to the Pythagorean Theorem in the case of right triangles, and (b) determination of an angle or side directly from its sine will lead to ambiguities since sin(x) = sin(pi - x), while determination from cosine or tangent will be unambiguous. Many formulae have been derived to avoid the sine ambiguity, but the simplest is to use the half angle formula which yields an unambiguous positive or negative result (by symmetry there are similar expressions for angles B and C).

7. $\sin(A / 2) = \sqrt{(1 - \cos(A)) / 2}$

For geolocation, plane triangles are adequate under certain circumstances (explained below) but the general case involves solving "spherical triangles". A spherical triangle is fully determined by three of its six characteristics (3 sides and 3 angles), and the basic relations used to solve a problem are similar to those above. However, the key differences are that the sides of a spherical triangle are measured in angular units (radians) rather than linear units, and the sum of the interior angles of a spherical triangle is greater than 180°.



(a)



(b)

Fig. 3

[9, pp.183,196]

Fig. 3 (a) shows how the intersection of three planes through a sphere forms two spherical triangles, one from the solid lines (foreground) and one from the dotted lines (background). The triangle degenerates into three points with the sum of the angles equal to 3*pi and the sum of the sides equal to 2*pi on the unit sphere. Euclid (300BC) Book 11, Proposition 21 provides a rigorous proof, with a corollary that the sum of the angles of a spherical triangle is greater than pi [9, pp.184]. The amount by which the sum of the three angles exceeds pi is referred to as the "spherical excess".

Labeling points and angles on a spherical triangle follows the normal conventions, as shown in Fig. 3 (b). The basic relations used to solve a spherical triangle are similar to those for a planar triangle: modifications to account for the curvature of the sides and the spherical excess lead to analogous formulae for side lengths and area, a Spherical Law of Cosines, and a "Spherical Pythagorean Theorem" (amongst Napier's Rules). Relevant examples are provided in section III.

GPS and Geolocation

The Earth is only approximately spherical, so no single value serves as its natural radius. However, the Earth deviates from a perfect sphere by only a third of a percent, making the sphere model adequate in many contexts. Using the polar minimum of 6,357.75 km and the equatorial maximum of 6,378.14 km, several different ways of modeling the Earth as a sphere yield a mean radius of 6,371 km [10].

GPS coordinates are based on dividing this perfect sphere of the world into 360 degrees of horizontal longitude and 180 degrees of vertical latitude. Each degree of latitude and longitude is divided into sixty minutes, and each minute is divided into sixty seconds, with fractions of a second offering finer-grained specification of a location. There are 3 common and equivalent formats for expressing location, ddd°mm'ss.ss", ddd°mm.mmm', and ddd.ddddd°, where d, m, and s stand for degrees, minutes, and seconds.

Degrees are expressed as a number between -180 and +180 for longitude, and a number between -90 and +90 for latitude. Zero degrees longitude is an arbitrary line, locations to the west of which are negative, and locations to the east are positive. Zero degrees latitude is the equator, with locations to the north as a positive number, and to the south as a negative number.

On the sphere of the world the longitude lines, also known as meridians, are the same distance apart at the equator and converge at the poles. The meter was originally defined such that ten million of them would span the distance from the equator to a pole, so at the equator each degree of both latitude and longitude represents approximately 111.32 km. Because the meridians get closer together moving from the equator toward either pole, one degree of longitude is multiplied by the cosine of the latitude, decreasing the indicative physical distance as illustrated in Table 1 for coordinates expressed as decimal degrees.

Table 1: Precision of GPS decimal places and indicative locations at particular latitudes [11]

Decimals	0	1	2	3	4	5	6	7
Equator	111 km	11 km	1 km	111 m	11 m	1 m	11 cm	1 cm
Quito, Ecuador; Maqcapa, Brazil; Kampala, Uganda; Thinadhoo, Maldives; Pontianak, Borneo								
23 N // S	102. 5 km	10.25 km	1 km	102.5 m	10.25 m	1 m	10.25 cm	1 cm
Havana, Cuba; Muscat, Oman; Shantou, China // Sao Paulo, Brazil; Windhoek, Namibia; Alice Springs, Australia								
45 N // S	78.7 km	7. 9 km	787 m	78.7 m	7. 9 m	78.7 cm	7. 9 cm	7. 9 mm
Portland OR USA; Limoges, France; Harbin, China // Rio Mayo, Argentina; Dunedin, New Zealand								
67 N // S	43. 5 km	4.35 km	435 m	43.5 m	4.35 m	43.5 cm	4.35 cm	4.35 mm
Coldfoot, AK USA; Repulse Bay, Canada; Inari Finland; Tomtor, Siberia // Adelaide, Casey Station, Antarctica								

It is worth noting that the fourth decimal place is comparable to the typical accuracy of an uncorrected GPS unit with no interference, while accuracy to the fifth decimal place requires differential correction with commercial GPS units. The seventh decimal place is near the limit of what GPS-based techniques can achieve with painstaking measures [12].

2.2 Essential Functions

To calculate the relative location of a point on the globe, we consider a spherical triangle given point A as longitude xA, latitude yA and point B as longitude xB, latitude yB, and derived point xC, yC. The distance between points A and B is easily calculated from their coordinates, so we need either the distance or the angles between point C and these points to determine its coordinates. Since we know the radius of the earth sphere, the characteristics of the triangle can be expressed as radians and the calculations done on the unit sphere.

The essential formulae are below as functions in the C programming language, which can be easily ported to another. One "peculiarity" of C is its lack of a built-in operator for exponentiation, because exponentiation not a primitive operation for most CPUs. Thus a function or a preprocessor macro such as `#define SQ(v) ((v)*(v))` is necessary to improve clarity. It is also convenient to have a static constant or preprocessor macro such as `#define D2R 0.017453293` for converting decimal degrees to radians (this is a library function in Java and python). Similarly, useful constants are `#define K2R 0.00015696` for converting kilometers to radians and `#define R_KM 6371` for the earth radius.

Indispensable references for the spherical earth formulae are Williams [13] where they are presented in a manner that facilitates practical calculation, and Osborn [9] which has the full proofs. For those who are interested, Veness [14] has implemented them in Javascript, along with additional calculations based on an elliptical earth model.

2.2.1 Distance Between Points

The planar linear distance between points A and B given their longitude (x) and latitude (y) is calculated as

```
double lenplnr(double xA, double yA, double xB, double yB) {
  return sqrt( (SQ(xB - xA)) + (SQ(yB - yA)) );     }
```

The Haversine formula returns the geodistance between the points in radians, accurate to around 0.3% because it is based on a spherical earth model. It is preferred to the spherical law of cosines because it maintains its accuracy at very small earth distances.

```
double lenhsine(double xA, double yA, double xB, double yB) {
    double sinlon = sin( ( (xB - xA) * D2R )/2 );
    double sinlat = sin( ( (yB - yA) * D2R )/2 );
    return 2 * asin(sqrt(
```

```
(SQ(sinlon)) + cos(xB*D2R) * cos(xA*D2R) * (SQ(sinlat))      )); }
```

2.2.2 Area of a Triangle Given Side Lengths

Heron's (aka Hero's) formula is used for the planar triangle

```
double areapt(double lenA, double lenB, double lenC) {
  double s = (a + b + c) /2;
  return sqrt( s * (s-a) * (s-b) * (s-c) );   }
```

l'Huiller's formula for a spherical triangle is analogous to Heron's for a plane triangle, and maintains its accuracy with small triangles. Argument and return values are in radians, multiply the returned value by SQ(R_KM) for the surface area enclosed by the triangle.

```
double areast(double lenA, double lenB, double lenC) {
  double s = (lenA + lenB + lenC) / 2;
  return 4 * atan(sqrt(
       tan(s/2) * tan((s-lenA)/2) * tan((s-lenB)/2) * tan((s-lenC)/2))   ));   }
```

2.2.3 Side Length of a Right Triangle

For the planar right triangle in Fig. 1 above, given the length of sides a and c the Pythagorean Theorem yields the length of the hypotenuse (side b) as

```
double lenrpthyp(double lenA, double lenC)  {
  return sqrt( (SQ(lenA)) + (SQ(lenC)) );   }
```

Alternatively, given the length of the hypotenuse and one other side, the length of the third side is

```
double lenrptside(double lenB, double lenC) {
  return sqrt( fabs( (SQ(lenB)) – (SQ(lenC)) ) );   }
```

For a spherical triangle with one right angle, there are ten relations (Napier's rules) that allow computing any unknown side or angle in terms of any two of the others. One of these uses the lengths of the sides that form the right angle: a Spherical Pythagorean Theorem.

```
double lenrsthyp(double lenA, double lenC)  {
  return cos(lenA) * cos(lenC);    }
```

or given the length of the hypotenuse and one other side, the length of the third side is

```
double lenrstside(double lenB, double lenC) {
  return cos(lenB) / cos(lenC);    }
```

2.2.4 The Special Case of Small Spherical Triangles

Spherical triangles with side lengths much less than the radius have a spherical excess so small they may be treated as planar. Legendre's theorem shows the angles of the spherical triangle exceed the corresponding angles of the planar triangle by approximately one third of the spherical excess when the side lengths of the spherical triangle are much less than 1 radian. For those who want to check, an intermediate point in the proof of Legendre's theorem presented in [9, equation D48, pp.201] is calculation of the area of the counterpart triangle using the side lengths.

```
/* Legendre’s theorem - convert the area of a spherical triangle
     to the area of a planar triangle with sides of the same length */
   double areaptst(double starea, double lenA, double lenB, double lenC) {
     return (starea / (1 + (( (SQ(lenA)) + (SQ(lenB)) + (SQ(lenC)) )/24)));   }


   /* Legendre’s theorem - planar to spherical */
   double areastpt(double ptarea, double lenA, double lenB, double lenC) {
     return (ptarea * (1 + (( (SQ(lenA)) + (SQ(lenB)) + (SQ(lenC)) )/24)));   }
```

2.2.5 Additional Formulae: Intersection of Circles

This calculation [15] saves a lot of work for these scenarios relative to using the triangle formulae above, which could be used to get the same result. Arguments are the radius of the two circles and the distance between their center points, the coordinates of the center points, and two-point (x,y) data structures passed by reference. The function effectively returns the two points where the circles intersect. It is presumed that the length of line PQ is less than the sum of the radius of the circles, so they actually do intersect (recall the Triangle Inequality Theorem).

```
/* calculate intersection points of two circles with center points P Q */
void circpts(double trnP, double trnQ, double lenPQ,
        double xP, double yP, double xQ, double yQ,
        struct RETpoint* nxy, struct RETpoint* vxy)  {
/* distance along line PQ equal to the radius of P */
  double lenPH = ((SQ(trnP)) - (SQ(trnQ)) + (SQ(lenPQ))) / (2*lenPQ);
/* length of a line to an intersection point perpendicular to line PQ */
  double lenHN = sqrt((SQ(trnP)) - (SQ(lenPH)));
/* vertical and horizontal distances between the circle center points */
  double difxPQ = xQ - xP;
  double difyPQ = yQ - yP;
/* point where the perpendicular line HN meets line PQ (xH,yH) */
  double xH = xP + (difxPQ * lenPH/lenPQ);
  double yH = yP + (difyPQ * lenPH/lenPQ);
```

```
/* offsets of the intersection points from (xH,yH) */
  double xVN = -difyPQ * (lenHN/lenPQ);
  double yVN =  difxPQ * (lenHN/lenPQ);
/* the actual intersection points */
  nxy->xcoord = xH + xVN;
  nxy->ycoord = yH + yVN;
  vxy->xcoord = xH - xVN;
  vxy->ycoord = yH - yVN;   }
```

2.2.6 Additional Formulae: Points on a Line

These useful functions take an argument of a point (x,y) data structure passed by reference; they could just as easily return this data structure.

```
/* point J on line I--J--K using lenIJ */
void ptada(double xI, double yI, double xK, double yK,
        double lenIJ, double lenIK, struct RETpoint* retxy)  {
  retxy->xcoord = xI + ( (lenIJ / lenIK) * (xK - xI) );
  retxy->ycoord = yI + ( (lenIJ / lenIK) * (yK - yI) );   }
/* point J on line I--J--K using lenKJ */
void ptaba(double xI, double yI, double xK, double yK,
        double lenIJ, double lenIK, struct RETpoint* retxy)  {
  retxy->xcoord = xI + ( (lenIJ / lenIK) * (xI - xK) );
  retxy->ycoord = yI + ( (lenIJ / lenIK) * (yI - yK) );   }
/* point K on line I--J--K using lenIK */
void ptbdab(double xI, double yI, double xJ, double yJ,
  double lenIK, struct RETpoint* retxy)  {
  double ikx = xJ - xI;
  double iky = yJ - yI;
  double bb = sqrt( (SQ(lenIK)) / ( (SQ(ikx)) + (SQ(iky)) ) );
  retxy->xcoord = xI + (ikx * bb);
  retxy->ycoord = yI + (iky * bb);    }
```

3. WSN Node Geolocation

On the Earth the excess of an equilateral spherical triangle with sides 21.3km (and area 393km2) is approximately 1 arc second (1/3600th of a degree). Taking account of both the convergence of the meridians and the curvature of the parallels, if the distance between points is around 20 km the planar distance formula will result in a maximum error of 30 meters (0.0015%) at 70 degrees latitude, 20 meters at 50 degrees latitude, 9 meters at 30 degrees latitude, and be precise at the equator [16]. From another perspective, at a height of two meters the clear line of sight is around 5 km due to the curvature of the earth, so the planar and spherical calculations would return the same result at any latitude. These are microdistances relative to the earth radius, so the choice of using spherical or planar triangle calculations is open, as long as the absolute necessity of using radians for functions that require then is kept firmly in mind.

Like the spherical earth model, an acceptable simplification for the initial calculations is to show the transmission radius of the wireless sensor network node as a circle. In actuality the transmission radius is irregular as it is subject to various types of interference and dependent on antenna characteristics, but these variables can be left for refinement suitable to specific deployments.

The scenarios presented here are base cases, working with minimal information. The goal is for a node to establish an initial estimate of its location, with a finite set of alternate points that could be its actual location. As more nodes join the network and go through this process more information becomes available, and the nodes can narrow their set of possible actual locations through a process of iterative refinement (discussed below). Ultimately the nodes in the network will be able to converge on a stable network map within a quantifiable margin of error for each node.

The base cases take advantage of the fact that wireless networks are inherently broadcast networks, so every node within range of a given node can hear all transmissions. This leads to the concept of “audible” and “inaudible” neighbors: nodes that can send to and receive from each other are audible neighbors, while a node that can hear its neighbor send to another node but cannot hear the response (i.e., eavesdrop on one side of the conversation) has an inaudible neighbor.

The base cases are also predicated upon the ability of a node to transmit its actual or presumed location, and its nominal transmission radius. Optimally the method will provide a way for a node to communicate the location and transmission radius of its audible and inaudible neighbors as well. Exactly how this is achieved (through beaconing, addressing, or some type of protocol for example) is not important for the calculations. The calculations are done with locations expressed as decimal GPS coordinates and distances in kilometers; other coordinate systems and distance measurements could be used.

All of the radios in the scenarios have an equal transmission radius, to avoid a situation where a radio has a transmission radius that can be completely contained within another – this situation has too high a degree of ambiguity to consider here. The diagrams are all drawn in a manner that would make it easy to superimpose a xy axis for easier comprehension; two-dimensional rotation would only affect the absolute values of the coordinates.

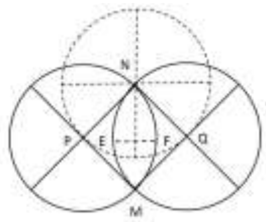
Journal of Applied Technology and Innovation
vol. 1, no. 1, (2017), pp. 10-27

3.1 Base Case: Two Audible Neighbors

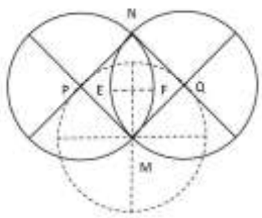
In this scenario, radio “dot” can communicate with (send to and receive from) radios P and Q, but P and Q cannot communicate with each other. In other words, P and Q are audible neighbors of “dot” and “dot” is their audible neighbor, while P and Q are inaudible neighbors of each other. Figure 4 shows three variations.

Radio “dot” can interpolate its location from the location coordinates of P and Q and their transmission radius. If “dot” positions itself at an intersection point of the two circles (N or M), it could not move farther away without losing contact but it could move closer, within the area of intersection of the two circles.

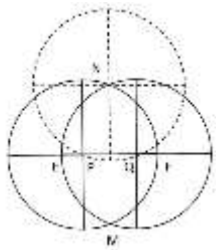
Points N and M are returned by the `circpts()` function, which in fact calculates these points using the height of a triangle with a base side length equal to the distance between P and Q, and the other two sides equal to their transmission radius. The area defined by the spherical triangle NEF or MEF defines the set of possible alternative locations for radio “dot”. However, without more information, “dot” cannot know which of N or M it should choose as its location. Nonetheless, a finite set of possibilities has been defined and an arbitrary choice between N and M (Fig. 4 (a) or (b)) must be made until further information is available.



(a)



(b)



(c)

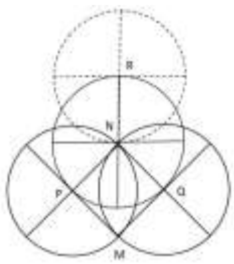
Fig. 4

The set of possible locations is inversely proportional to the distance between P and Q: the shorter the distance between them, the greater the area of the triangle becomes. As illustrated in Fig. 4 (c), the calculations are the same when P and Q are audible neighbors, the set of alternative locations just gets larger.

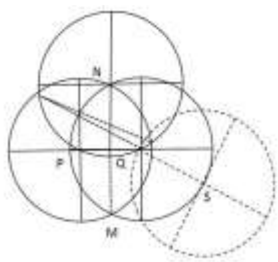
3.2 Base Case: One Audible Neighbor with Two Neighbors

This case is built on the previous one, after radio “dot” has arbitrarily chosen its position as N. In the first variant, Fig. 5 (a), radio N is the audible neighbor of the new radio “dot”, which positions itself at point R; in the other variant (Fig. 5 (b)) radio Q is the audible neighbor of the new radio “dot”, which positions itself at point S.

The new radio “dot” uses the intersections of the inaudible neighbor



(a)



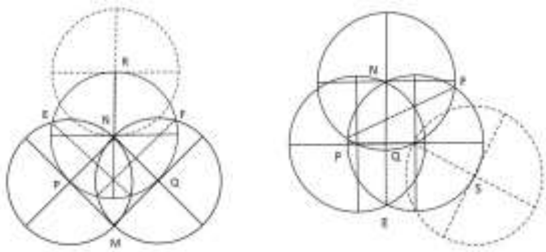
(b)

Fig. 5

circles (P and Q in the first case, P and N in the second) to obtain two points, and extends the line from one of these points through the coordinates to a point that is its transmission radius away from the audible neighbor.

The variants are only distinguished by the location of the point returned by the `circpts()` function that is closest to the audible neighbor: as illustrated in Fig. 5 (a), in the NR variant this point is exactly N, in the QS variant (Fig. 5 (b)) it is not quite exactly Q. Thus it is important to recognize that for this scenario only one of the points returned is useful – the point farther away from the audible neighbor.

In both variants the set of possible alternative points is calculated in the same manner: calculate the overlap of the circle of the audible neighbor with each inaudible neighbor (PN and QN in the first case, QP and QN in the second) and use the points that are farthest apart from each other. In the first case this yields a set of possible actual locations for R as the sum of the areas of triangles RNE and RNF as shown in Fig. 6 (a), in the second it is the sum of the areas of SQE and SQF as shown in Fig. 6 (b). The area is relatively large, but finite for all practical purposes.



(a) **(b)**

Fig. 6

3.3 Base Case: One Audible Neighbor with One Audible Neighbor

This case, illustrated in Fig. 7, is likely to arise for edge nodes in particular. The new radio "dot" positions itself at point S by simply extending line PQ by its transmission radius.

In this case triangle QEF defines the inverse of the set of points for the alternative locations: the actual location of S is any point at distance less than or equal to the transmission radius of S from Q, and outside triangle EFQ.

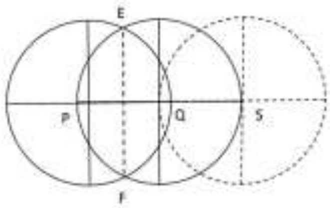


Fig. 7

4. Refining the Estimate

Using the triangles that define the set of alternative locations to refine the initial location estimate is where this exercise gets interesting. In principle, the labeled points in Fig. 8 represent five iterations of the initial calculations using different pairs of audible neighbors, but the diagram is made to illustrate some key ideas rather than represent the outcome of a realistic application of the base case.

It is essential to keep in mind that the proximity of the calculated points offers no insight: any point inside the associated triangle is an equally valid (and equally probable) location for the node, since the calculated points simply represent limits on how far away the nodes can be from each other. Refining the estimate involves examining the overlapping areas of the triangles; ideally they would all overlap and yield a very small set of possible alternative locations, but an outcome like the one illustrated in Fig. 8 where no single point satisfies all of the constraints is theoretically possible.

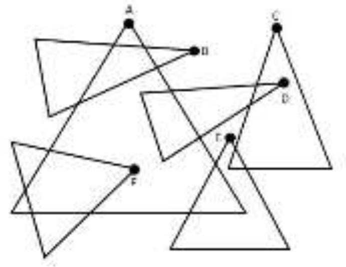


Fig. 8

In any case, creating this mapping requires choosing a pair of triangles and either (a) checking to see if any of the points of one lie inside the area of the other, or (b) checking if the sides of one intersect sides of the other. The essential calculations are quite similar, and rely on checking the sign of the vector cross product. Put very simply, the cross product of two vectors is another vector that is at right angles to both.

```
/* Vector Cross Product */
double vcp(double xP, double yP,
       double xF, double yF, double xG, double yG) {
  return ((xP - xG)*(yF - yG) - (xF - xG)*(yP - yG));  }
/* point P is inside a triangle if it is on the same side of each line */
int ispint(double xP,double yP, double xA,double yA,
       double xB,double yB, double xC,double yC) {
  int vs = 0;
  vs += ( vcp(xP,yP, xA,yA, xB,yB) < 0 ? 1:0 );
  vs += ( vcp(xP,yP, xB,yB, xC,yC) < 0 ? 1:0 );
  vs += ( vcp(xP,yP, xC,yC, xA,yA) < 0 ? 1:0 );
  return ( ((vs == 3) || (vs == 0)) ? 1:0 );   }
/* segment PQ intersects AB when P and Q are on opposite sides of AB
  bonus: if not, which side is PQ on, or is this effectively zero */
int segint(double xP,double yP, double xQ,double yQ,
       double xA,double yA, double xB,double yB) {
  double p = vcp(xP,yP, xA,yA, xB,yB);
  double q = vcp(xQ,yQ, xA,yA, xB,yB);
  if ( (p > 0) && (q > 0) ) {
    if ( (EZERO(p)) && (EZERO(q)) )  return  2;
```

```
  else                    return  1;
  }
if ( (p < 0) && (q < 0) ) {
  if ((EZERO(-p)) && (EZERO(-q)))  return -2;
  else                    return -1;
  }
return 0;  }
```

The `segint()` function uses what is commonly known as an "epsilon test" rather than testing if the value is zero because of the accuracy of the calculations. Calculations with a variable type of float give from 6 to 9 significant decimal digits while double gives 15–17 digit precision, because of the way binary translates to decimal. In this case seven decimal digit are adequate, so a function or a preprocessor macro such as `#define EZERO(v) ((v)<(0.00000005))` is called for.

Looking closely at Fig. 8 it is apparent that we may not necessarily need to check every point or line. Triangle D has two points inside triangle A, so the first one we find is sufficient to know they overlap. Similarly, all three sides of F intersect sides of A, so a single intersection test would be sufficient. B and E have no points inside A, so only the intersection test will offer insight, but C has no intersections with A so the point test would quickly confirm if it is completely inside or completely outside. There is no way to tell in advance if the point or intersection test will be more efficient, although with the intersection test it might be possible to increase the odds of not choosing the wrong line by checking closer line segments first. Depending on the power, processor and memory resources available, it could be possible to put the triangles or the line segments into a spatial tree structure of some type - a grid, quad-tree or kd-tree would allow testing multiple triangles or multiple line segments simultaneously.

5. Results and Discussion

As noted above, the goal is for a node to establish an initial estimate of its location and a finite set of alternate points that could be its actual location, given minimal information. Given the coordinates and transmission radius of two or three neighbors, this can be done by passively listening to transmissions and doing some efficient calculations to determine the farthest a node can be from its neighbors and still receive transmissions. As more nodes join the network and go through this process more information becomes available, and the nodes can narrow their set of possible actual locations through a process of iterative refinement. Ultimately the nodes in the network will be able to converge on a stable network map within a quantifiable margin of error for each node.

The overall accuracy of this network map will depend upon several factors. First, the mathematical process of iterative refinement will require decision rules for choosing amongst several possible locations, and each choice will have a "ripple effect" as the data is used by other nodes. For there to be any relationship to reality at all, some nodes must know their coordinates with complete confidence (ground-truth) to seed the network map, e.g., having a GPS receiver or preprogrammed with accurate GPS coordinates. A node with more than one ground-truth neighbor should be able to interpolate its location with a high degree of confidence (possibly even accuracy), while nodes with no ground-truth neighbors will necessarily have lower confidence in their position estimates. However, as Table 1 shows, there would be an upper limit on accuracy of 8-10 meters using a typical GPS unit with no interference.

The additional information of whether an audible or inaudible neighbor is a ground-truth node or has a ground-truth node as a neighbor would be useful for decisions about where to move within the set of alternative locations. These nodes are the best starting point for the process of determining the overlap of the triangles that define sets of alternative locations. Staying within the union of these sets will yield an estimate that can be considered more accurate than any potential location calculated with data from neighbors that do not have direct or indirect knowledge of ground-truth. A simple move to the middle strategy might be enough to provide acceptable accuracy when these circumstances apply.

The transmission radius of a node is essential information required for determining relative location, but this is also a source of uncertainty. This information can be configured, but will always be a best guess because of the numerous factors that affect path loss. The terrain over which signals travel, the level of moisture in the air, the shape of obstacles and their location relative to the two antennas can all affect signal reception, individually or in concert. In practice, models based on empirical measurements over a given distance in a given frequency range for a particular geographical area or building are used to describe signal propagation, recognizing that these represent an average and are less accurate in a more general environment. Further, empirical measurements have shown that the difference between the average and the actual path loss is random and log-normal distributed, which means that any receiver in range of the transmitter has a nonzero probability of receiving a signal that is too weak to use, and some nodes beyond the average range will receive a usable signal [6].

The practical implication is that the value chosen for the transmission radius when configuring the network nodes must be considered nominal. If the case depicted in Fig. 8 arises, where no single point satisfies all of the constraints, the overall accuracy of the network map should be improved by selecting a point that takes "reasonable" variation into account. It might also be useful to extend the capabilities of the nodes to communicate an indicator of the average quality of the signal received from each audible neighbor, so the decision rules could be tuned to prefer to move closer to stronger signals and not move farther away from weaker sources.

6. Conclusion

It is possible to create a system where a WSN node can calculate an initial estimate of its location and a finite set of alternate points that could be its actual location, given the coordinates and transmission radius of two or three neighbors. The necessary information can be acquired by passively listening to transmissions, assuming a nodes in the network can transmit their actual or presumed location and nominal transmission radius; optimally the nodes would also be able to communicate the location and transmission radius of its audible and inaudible neighbors, and the average quality of the signal received from each audible neighbor. Exactly how this is achieved (e.g., through beaconing, addressing, or some type of protocol) is not important for the calculations.

As with all WSN localization techniques, a primary goal is to derive a satisfactory degree of accuracy from inconsistent radio communication while minimizing power consumption. The set of functions presented here provide efficient calculations to determine the farthest a node can be from its neighbors and still receive transmissions. Three base cases (two audible neighbors, one audible and two other neighbors, one audible neighbor that has only one audible neighbor) are sufficient for the initial calculations. The process of narrowing the set of possible actual locations through iterative refinement as more nodes join the network is where the decision rules will have to be tuned for specific characteristics of the deployment, in order to determine a quantifiable margin of error for each node. The result is a unique low-cost way to address limitations on determining directionality in broadcast networks.

Acknowledgments

Development of the functions presented here has benefitted immensely from innumerable discussions and suggestions posted in various StackExchange fora (http://stackexchange.com/).

References

[1] V. Gungor, and G. Hancke, "Industrial Wireless Sensor Networks: Challenges, Design Principles and Technical Approaches", *IEEE Trans. Ind. Electron.*, vol. 56, no.10, 2009.

[2] L. Cheng et al., "A Survey of Localization in Wireless Sensor Networks", *Int. J. Distributed Sensor Networks*, vol. 2012, Article ID 962523, Dec. 2012.

[3] M. Srbinovska, et al, "Localization Techniques in Wireless Sensor Networks using Measurement of Received Signal Strength Indicator". Electronics, vol. 15 no. 1, 2011.

[4] H. Maheshwari, "Optimizing Range Aware Localization in Wireless Sensor Networks", PhD Dissertation, Fac. Eng., Univ. of Leeds, UK, 2011.

[5] Sandor Szilvasi, et al, "Interferometry in Wireless Sensor Networks", in Interferometry - Research and Applications in Science and Technology, Rijeka, Croatia: InTech, 2012, ch.21, pp. 437-62.

[6] M. Tsai, (2011, Oct.20) "Path-loss and Shadowing (Large-scale Fading)." National Taiwan University [Online]. Available: https://www.csie.ntu.edu.tw/~hsinmu/courses/_media/wn_11fall/path_loss_and_shadowing.pdf.

[7] CK-12 Foundation, (2017), *2.1 Radian Measure* [Online]. Available: http://www.ck12.org/book/CK-12-Trigonometry-Concepts/section/2.1/?noindex=None

[8] Florida Center for Instructional Technology, (2017), *Radians in Complete Circle*, [Online]. Available: http://etc.usf.edu/clipart/36600/36685/radians_36685.htm

[9] P. Osborn, *The Mercator Projections*, Zenodo, 2013. doi:10.5281/zenodo.35392

[10] D. Williams, (2016, Dec. 23) *Earth Fact Sheet* [Online]. Available: https://nssdc.gsfc.nasa.gov/planetary/factsheet/earthfact.html

[11] T. O'Daniel et al., "Localization using GPS Coordinates in IPv6 Addresses of Wireless Sensor Network Nodes", *Indian J. Sci. & Technology*, vol. 10, no. 8, Feb. 2017.

[12] W. Huber, (2011, Apr. 18) "How to measure the accuracy of latitude and longitude?" [Online]. Available: http://gis.stackexchange.com/questions/8650/how-to-measure-the-accuracy-of-latitude-and-longitude/8674#8674

[13] E. Williams, (2011, Apr.24) *Aviation Formulary v1.46* [Online]. Available: http://edwilliams.org/avform.htm

[14] C. Veness, (2015) *Latitude/longitude spherical geodesy formulae & scripts* [Online]. Available: http://www.movable-type.co.uk/scripts/latlong.html

[15] P. Bourke, (1997, Apr) "Intersection of Two Circles", *Circles and Spheres* [Online]. Available: http://paulbourke.net/geometry/circlesphere/

[16] R.Chamberlin, (1996, Oct.) "Q5.1: What is the best way to calculate the great circle distance (which deliberately ignores elevation differences) between 2 points?", *Geographic Information Systems FAQ* [Online]. Available: http://www.faqs.org/faqs/geography/infosystems-faq/