A Computational Geometry Method for DTOA Triangulation

Xiaochun Xu
Computer and Information Science and Engineering Department
University of Florida
Gainesville, FL 32611
Email: xxu@cise.ufl.edu

Nageswara S. V. Rao
Computer Science and Mathematics Division
Oak Ridge National Laboratory
Oak Ridge, TN 37831
Email: raons@ornl.gov

Sartaj Sahni
Computer and Information Science and Engineering Department
University of Florida
Gainesville, FL 32611
Email: sahni@cise.ufl.edu

***Abstract*— We present a computational geometry method for the problem of triangulation in the plane using measurements of distance-differences. Compared to existing solutions to this well-studied problem, this method is: (a) computationally more efficient and adaptive in that its precision can be controlled as a function of the number of computational operations, making it suitable to low power devices, and (b) robust with respect to measurement and computational errors, and is not susceptible to numerical instabilities typical of existing linear algebraic or quadratic methods. This method employs a binary search on a distance-difference curve in the plane using a second distance-difference as the objective function. We establish the unimodality of the directional derivative of the objective function within each of a small number of suitably decomposed regions of the plane to support the binary search. The computational complexity of this method is $O(\log(1/\gamma))$, where the computed solution is guaranteed to be within a distance of γ to the actual solution. We present simulation results to compare this method with existing DTOA triangulation methods.**

Keywords: Triangulation, difference in time of arrival, computational geometry, computational complexity.

I. Introduction

The problem of computing the location of an object from measurements of distance-differences from three known locations is well-studied (for at least three decades) under the title of Difference of Time-of-Arrival (DTOA) localization. This problem arises in a number of established areas such as tracking in aerospace systems [1], [2]. Recently, it has received renewed attention due to the increasing proliferation of wireless sensor networks [3], [4] and embedded networked systems [5]. In several of these applications, the wireless nodes are limited in power and yet the localization computations may have to be repeated quite frequently. Consequently, it has become important to trade-off the number and type of computations needed for localization to save power by gracefully degrading the quality of solution. In addition, the computational precision of arithmetic operations may be limited in some sensor nodes, but its impact on the precision of localization is not well understood. These factors motivate a closer examination of the computational aspects of DTOA triangulation methods; however, our results could be of more general interest as well.

There are two basic formulations of the DTOA localization problem: (a) distance-differences to an object, such as origin of a plume, are measured from known locations, and the problem is to estimate the location of the object; and (b) a device, such as a sensor node, receives distance-differences from beacons with known locations, and the problem is to estimate the location of the sensor node, that is self-localization. The classical source localization problem using DTOA measurements has been solved using two general approaches: (i) linear algebraic solution which typically involves matrix inversion and solving a quadratic equation [2], [6], and (ii) intersection of hyperbolic curves [7]. A recent overview of network-based localization methods may be found in [3]–[5], [8]. In general, the quality of the location estimate is a complex function of the precision with which the underlying numerical operations are implemented, and consequently, there is no apparent and simple way of relating the computations to the "quality" of location estimate. In particular, it is unclear if devoting more computational operations would increase the accuracy of these methods, or conversely if it is possible to reduce the computations without drastically affecting the quality of location estimate. In addition, sensor errors can have drastic

effects on DTOA localization methods. For example, as will be shown in Section VI under simple random noise conditions, the quadratic equation of [2], [6] may have imaginary roots thereby rendering the method incomplete. Also, numerical instabilities may arise in the computations implemented with low precision operations wherein matrix inversions needed for linear algebraic methods may become ill-conditioned resulting in large estimation errors.

The underlying geometric nature of this problem has been well-known [1] although we are unaware of methods that exploit it to fine tune computations as done in several computational geometry methods [9], [10]. We present a computational geometric method for DTOA localization based on a binary search on an algebraic curve defined by a distance-difference function. We exploit the monotonicity of the directional derivative of the other distance-difference on it to support the binary search. The computational complexity of this method is $O(\log(1/\gamma))$, where the computed solution is guaranteed to be within a distance of γ to the actual solution. Alternatively, by fixing the number of operations to k, one can achieve the precision $\gamma = O\left(2^{-k}\right)$. This method is robust with respect to distance measurement errors: (i) γ is of the same order of magnitude as errors in distance measurements; in methods that involve division operations such guarantees cannot be made; and (ii) it is complete in that it will always return an answer, even under random measurement errors. This method is a generalization of the DTOA localization method in [11] proposed as a part of plume identification when the source is inside the acute triangle formed by sensors. In our case, the object can be located anywhere in the monitoring region. In addition, we also provide a detailed analysis of the underlying computation and the proof of the required monotonicity property of the underlying directional derivative.

This paper is organized as follows. In Section II, we examine the relationship between proximity in Euclidean space and proximity in DTOA space. Although all previous localization methods have focused on using proximity in DTOA space, our analysis of Section II shows that this does not guarantee proximity in Euclidean space. We describe our geometric DTOA triangulation method in Section III. Our geometric method guarantees proximity in both Euclidean and DTOA spaces. We prove the correctness of the method in Sections IV and V. We present simulation results in Section VI. A conclusion is drawn in Section VII.

II. Euclidean And DTOA Spaces

Let $S_i = (x_i, y_i)$, $1 \leq i \leq k$, be the locations of k sensors in Euclidean space R^2. For any point $P = (x, y)$ in R^2, the distance, $d(P, S_i)$, between P and S_i is $\sqrt{(x - x_i)^2 + (y - y_i)^2}$. A signal that originates at P at time 0 arrives at S_i at time proportional to $d(P, S_i)$. For simplicity, we assume that the arrival time is $d(P, S_i)$. The difference, Δ_{ij}, in the time of arrival (DTOA) at S_i and S_j is given by

$$\Delta_{ij}(P) = d(P, S_i) - d(P, S_j).$$

Let $\overline{S_iS_j}$ be the line through the points S_i and S_j. As we move P from S_i to S_j along the line $\overline{S_iS_j}$, $\Delta_{ij}(P)$ varies monotonically and linearly from $-d(S_i, S_j)$ to $d(S_i, S_j)$, and equals 0 at the bisector point. From this observation and the triangle inequality, it follows that $|\Delta_{ij}(P)| \leq d(S_i, S_j)$. Furthermore, the locus, $L_{12}(\delta)$, of points defined by

$$L_{ij}(\delta) = \{P | \Delta_{ij}(P) = \delta\}$$

is a hyperbola.

The DTOA space of all $(k-1)$-tuples [$\Delta_{12}(P)$, $\Delta_{13}(P)$,..., $\Delta_{1k}(P)$] forms a $(k-1)$-dimensional vector space denoted by δ^{k-1}. Each point $P = (x, y)$ in R^2 has a unique dual point $P' = (p'_1, p'_2, ..., p'_{k-1})$ in δ^{k-1}, where $p'_j = \Delta_{1(j+1)}(P)$, $j = 1, 2, ..., (k-1)$. However, each point $P' = (p'_1, p'_2, ..., p'_{k-1})$ in δ^{k-1} may have zero or more dual points in R^2. In fact, the dual points of P' are those points in R^2 that are common to (i.e., the common intersections) the $k-1$ hyperbolas $L_{1,j+1}(P)$, $1 \leq j < k$.

In this paper, we consider the *DTOA localization* problem of estimating the location of a source S from the measurements of $\Delta_{1j}(S)$, $2 \leq j \leq k$, S' = [δ_{12}, ..., δ_{1k}] in δ^{k-1}. When there is the possibility of errors in the measurement of the Δ_{1j} values, existing DTOA localization algorithms [14]–[17], estimate the source location by minimizing the sum of least squares error in δ^{k-1} space. We show, in this section, that an estimate that is close in δ^{k-1} space may not be close in Euclidean space R^2. However, an estimate that is close to the source in Euclidean space R^2, is necessarily close to the source in δ^{k-1} space (Lemmas 1 and 2, respectively).

Lemma 1: Two points that are close to one another in δ^{k-1} space may be arbitrarily far apart in R^2.

Proof: Consider the three sensors $S_1(1, 0)$, $S_2(1, -1)$, and $S_3(-1, 0)$ ((1,0) is the location in R^2 of sensor S_1). Let δ_{12} = -$d(S_1, S_2)$ = -1 and δ_{13} = -2e, where e is a small positive. The hyperbolic equation for $L_{12}(\delta_{12})$ is x=1 and $y >= 0$. In other words, $L_{12}(\delta_{12})$ degenerates from a hyperbola to a ray from S_1 vertically up to infinity. The hyperbolic equation for $L_{13}(\delta_{13})$ is $x^2/(e^2) - y^2/(1 - e^2) = 1$ where $x > 0$. The intersection, P, of $L_{12}(\delta_{12})$ and $L_{13}(\delta_{13})$ is $(1, (1 - e^2)/e)$.

Now, suppose we change the value of δ_{13} to -4e. The hyperbolic equation for the new $L_{13}(\delta_{13})$ is $x^2/(4e^2) - y^2/(1 - 4e^2) = 1$ where $x > 0$ and the new intersection, Q, between L_{12} and L_{13} is $(1, (1 - 4e^2)/(2e))$. The distance between P and Q in δ^2 space is $2e$. However, the distance, $d(P, Q)$, between P and Q in Euclidean space R^2 is $|(1 - 4e^2)/(2e) - (1 - e^2)/e| = e + 1/(2e)$. As can be seen, $d(P, Q)$ becomes large as the distance in δ^2 space approaches zero. ■

Lemma 2: Given two points $P(x_p, y_p)$ and $Q(x_q, y_q)$ in Euclidean space R^2 and their respective dual points $P'(x'_p, y'_p)$ and $Q'(x'_q, y'_q)$ in δ^2 space, then we have $d(P', Q') \leq 2\sqrt{2} * d(P, Q)$

Proof: From the definition, we have $x'_p = \Delta_{12}(P) = d(P, S_1) - d(P, S_2)$ and $x'_q = \Delta_{12}(Q) = d(Q, S_1) - d(Q, S_2)$.

$$\begin{aligned} |x'_p - x'_q| &= |(d(P, S_1) - d(P, S_2)) - (d(Q, S_1) - d(Q, S_2))| \\ &= |(d(P, S_1) - d(Q, S_1)) - (d(P, S_2) - d(Q, S_2))| \\ &\leq |d(P, Q) - (-d(P, Q))| \\ &= 2 * d(P, Q) \end{aligned}$$

Similarly, we have $|y'_p - y'_q| \leq 2 * d(P, Q)$ as well. So,

$$d(P', Q') = \sqrt{(x'_p - x'_q)^2 + (y'_p - y'_q)^2} \leq 2\sqrt{2} * d(P, Q)$$

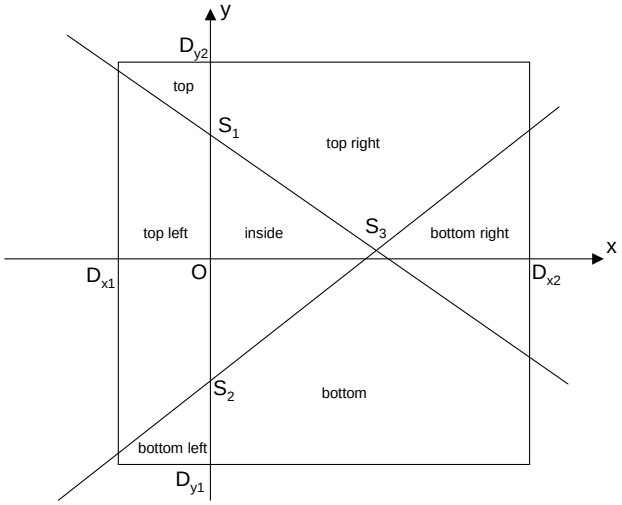

Fig. 1. Canonical placement of 3 sensors and partitioning of monitoring region

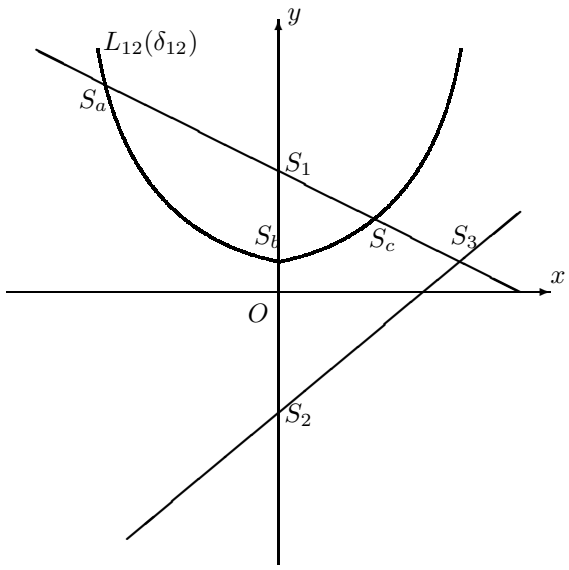

Fig. 2. Canonical placement of 3 sensors and $L_{12}(\delta_{12})$

■

Corollary 1: Given two points P and Q in Euclidean space R^2 and their respective dual points P' and Q' in DTOA space δ^k, $d(P', Q') \leq 2\sqrt{k} * d(P, Q)$.

III. Geometric DTOA Method

In the remainder of this paper, we consider only the case when we have $k = 3$ sensors, S_1, S_2, and S_3. Without loss of generality (w.l.o.g.), we choose our coordinate system so that the line $\overline{S_1S_2}$ falls on the y-axis and so that the midpoint of this line is the origin O as shown in Figure 1. The $[D_{X1}, D_{X2}] \times [D_{Y1}, D_{Y2}]$ box shown in Figure 1 is the monitoring region within which the source S is to be localized. The lines $\overline{S_1S_2}$, $\overline{S_2S_3}$, and $\overline{S_1S_3}$ partition the monitoring region as shown in Figure 1. Although this figure has all sensors within the monitoring region, our development of the geometric localization method does not require this. In fact, the method works even when some or all of the sensors are outside the monitoring region.

Figure 2 shows our three sensors together with the locus $L_{12}(\delta_{12})$. This locus may be partitioned into segments that lie wholly within a region of the partitioning of Figure 1. The segment end points are designated S_j, where j is a lowercase letter. So, $\overline{S_aS_b}$ and $\overline{S_bS_c}$ are two of the segments that L_{12} is partitioned into in Figure 2. Notice that because of our choice of coordinate system, as we move a point P along any segment of $L_{12}(\delta_{12})$, the $x-$ and $y-$coordinates of the point vary monotonically. This is a consequence of the vertical orientation of L_{12}, which, in turn, is assured by the chosen coordinate system.

Let (x_i, y_i) and (x_j, y_j), $x_i \leq x_j$ be the end points of an L_{12} segment and let $P = (x, y)$ be any point on this segment. From Lemma 3 (Section IV), it follows that $x_i \leq x \leq x_j$ and $\min\{y_i, y_j\} \leq y \leq \max\{y_i, y_j\}$. Also, as we move P along a segment of $L_{12}(\delta_{12})$, $\Delta_{13}(P)$ varies monotonically (Section V). In particular, it monotonically decreases with x for the segments in the top, bottom left, and bottom right regions and monotonically increases for the remaining segments. Based on these key observations, our overall strategy to estimate the source S is to utilize the monotonicity of $\Delta_{13}(P)$ to perform a binary search within each segment of L_{12} to determine a set, U, of points such that U has at least one point within a specified accuracy γ of each intersection between $L_{12}(\delta_{12})$ and $L_{13}(\delta_{13})$ that is in the monitoring region. Further, the number of points in U is at most equal to the number of such intersections. It follows that the true source location is within a distance γ (in R^2) of one of the points in U. The details are presented in algorithm geometric_DTOA(δ_{12}, δ_{13}).

algorithm geometric_DTOA(δ_{12}, δ_{13});
begin
  $(x_{12}, y_{12}) \leftarrow$ intersection point of L_{12} with $\overline{S_1S_2}$;
  $I_{X1} \leftarrow$ set of x-coordinates of intersections of L_{12} with $\overline{S_1S_3}$;
  $I_{X2} \leftarrow$ set of x-coordinates of intersections of L_{12} with $\overline{S_2S_3}$;
  $I_X \leftarrow \{D_{X1}, x_{12}, D_{X2}\} \bigcup I_{X1} \bigcup I_{X2}$;
  $I_X \leftarrow I_X - \{x | x \in I_X \;\&\&\; (x < D_{X1} \;||\; x > D_{X2})\}$;
  $I_{sort} \leftarrow$ sort (I_X);
  let $I_{sort} = \{x_{(1)}, x_{(2)}, \ldots, x_{|I_{sort}|}\}$
  let $\{y_{(1)}, y_{(2)}, \ldots, y_{|I_{sort}|}\}$ be the corresponding y-coordinates;
  $U \leftarrow \emptyset$;
  for $i = 1, \ldots |I_{sort}| - 1$ **do**
    $U \leftarrow U \bigcup \{$ locate_L_{13} $(x_{(i)}, y_{(i)}, x_{(i+1)}, y_{(i+1)})\}$;
  return U;
end

Algorithm geometric_DTOA first determines the segments of L_{12}. The end points of these segments are just the intersections of the curve $L_{12}(\delta_{12})$ with each of the three lines $\overline{S_1S_2}$, $\overline{S_1S_3}$, and $\overline{S_2S_3}$. Although a line and a hyperbola may intersect twice (except in the degenerate case when the hyperbola is a vertical ray), our choice of coordinate system ensures that L_{12} intersects $\overline{S_1S_2}$ exactly once, except when L_{12} is a ray. We ignore this case when L_{12} is a vertical ray for now. So, the number of intersections is at most 5 and, in the worst case, we need to consider 6 segments of the hyperbola $L_{12}(\delta_{12})$. The computation of (x_{12}, y_{12}), I_{X1} and I_{X2} may be carried out either by binary searches on the lines $\overline{S_1S_2}$, $\overline{S_1S_3}$, and $\overline{S_2S_3}$ with L_{12} as objective function or by a

method similar to that used in *algorithm* locate_$L_{12}(x, y_L, y_R)$. Note that intersections outside the monitoring region may be ignored.

Next, a binary search is performed within each segment, as shown in *algorithm* locate_$L_{13}(x_L, y_L, x_R, y_R)$. If δ_{13} is not in the range $[\Delta_{min}, \Delta_{max}]$, the algorithm concludes, from the monotonicity property, that there is no point P with $\Delta_{13}(P) = \delta_{13}$ on the segment currently being searched. Otherwise, the continuity of the directional derivative of Δ_{13} implies that there is a point P on the segment for which $\Delta_{13}(P) = \delta_{13}$ and a binary search, as described in the **do-until** loop of *algorithm* locate_$L_{13}(x_L, y_L, x_R, y_R)$, to locate a point on L_{12} that is within γ of P in R^2. In each iteration, either the x- or y-range to be considered is halved by appropriately updating P_1 or P_2. As proved in Theorem 1, our algorithm guarantees to return a point that is within a distance γ, in R^2 space, of the true source location.

algorithm locate_$L_{13}(x_L, y_L, x_R, y_R)$;
begin
  $x_1 \leftarrow x_L$;
  $y_1 \leftarrow y_L$;
  $P_1 = (x_L, y_L)$;
  $x_2 \leftarrow x_R$;
  $y_2 \leftarrow y_R$;
  $P_2 = (x_R, y_R)$;
  $\Delta_{min} = min\{\Delta_{13}(P_1), \Delta_{13}(P_2)\}$;
  $\Delta_{max} = max\{\Delta_{13}(P_1), \Delta_{13}(P_2)\}$;
  if $(\Delta_{min} > \delta_{13})$ or $(\Delta_{max} < \delta_{13})$ **then**
    return($null$);
  do{
    if $|x_1 - x_2| \geq |y_1 - y_2|$ **then**
      $x \leftarrow (x_1 + x_2)/2$;
      $y \leftarrow$ locate_$L_{12}(x, y_L, y_R)$;
    else
      $y \leftarrow (y_1 + y_2)/2$;
      $x \leftarrow$ locate_$L_{12}(y, x_L, x_R)$;
    $\hat{P} = (x, y)$;
    if $(\Delta_{13}(P_1) - \delta_{13}) * (\Delta_{13}(\hat{P}) - \delta_{13}) > 0$ **then**
      $P_1 = \hat{P}$;
    else
      $P_2 = \hat{P}$;
  } **until** $dist(P_1, P_2) \leq \gamma$
  return($\hat{P}$);
end

When L_{12} is a vertical ray, the source lies on the line $\overline{S_1S_2}$ but outside the segment $\overline{S_1S_2}$, whose end points are S_1 and S_2. In this case, we may do a binary search on the relevant segment of the y-axis that is contained in the monitoring region and excludes either the segment from S_1 to $-\infty$ or the segment from S_2 to ∞. As shown in Section V, L_{13} is monotone on both these vertical segments.

IV. Correctness and Complexity of the Method

In this section, we establish the correctness of our geometric DTOA method subject to the monotonicity of L_{13} on each segment of L_{12}. This monotonicity property is established in Section V. The following assumes that L_{12} is not a vertical ray. The correctness proof for the case when L_{12} is a vertical ray (note that this case, which is not included in the statement of algorithm geometric_DTOA, is handled by a binary search on a segment of the y-axis) is similar and simpler.

algorithm locate_$L_{12}(x, y_L, y_R)$;
begin
  substitute x into the hyperbolic equation for $L_{12}(\delta_{12})$;
  solve the quadratic equation for y;
  return the solution that is in the range $[min\{y_L, y_R\}, max\{y_L, y_R\}]$.
end

Lemma 3: As you move along each segment of $L_{12}(\delta_{12})$, the x-coordinate (y-coordinate) monotonically increases or decreases.

Proof: Follows from the definition of a segment and our choice of coordinate system. ∎

Lemma 4: For any point P on a segment $\overline{S_iS_j}$ of $L_{12}(\delta_{12})$, $\max\{d(S_i, P), d(S_j, P)\} \leq d(S_i, S_j)$.

Proof: Let $S_i = (x_i, y_i)$, $S_j = (x_j, y_j)$, and $P = (x, y)$. W.l.o.g., we may assume that the segment is oriented so that $x_i \leq x_j$. From Lemma 3, we have $x_i \leq x \leq x_j$ and $y_i \leq y \leq y_j$ (or $y_j \leq y \leq y_i$). So, $\max\{|x_i - x|, |x - x_j|\} \leq |x_i - x_j|$ and $\max\{|y_i - y|, |y - y_j|\} \leq |y_i - y_j|$. Hence, $\max\{d(S_i, P), d(S_j, P)\} \leq \sqrt{\max\{|x_i - x|, |x - x_j|\}^2 + \max\{|y_i - y|, |y - y_j|\}^2} \leq d(S_i, S_j)$. ∎

Lemma 5: Let $P = (x, y)$ be a point on a segment $\overline{S_iS_j}$ of $L_{12}(\delta_{12})$ such that $\Delta_{13}(P) = \delta_{13}$. The search of this segment using algorithm locate_L_{13} returns a point $\hat{P}$ on $L_{12}(\delta_{12})$ such that $d(\hat{P}, P) \leq \gamma$, where γ is the desired accuracy.

Proof: (x_L, y_L) and (x_R, y_R) are the end points of the segment $\overline{S_i, S_j}$. Since Δ_{13} is monotone on this segment and P is on the segment, δ_{13} is in the range [Δ_{min},Δ_{max}]. So, the binary search described in the algorithm is performed. The original search rectangle is determined by point $P_1 = (x_L, y_L)$ and $P_2 = (x_R, y_R)$. In each iteration, we chop the x- or y-range, whichever is larger, of the search rectangle into half and choose the half that contains P as the new search rectangle by updating P_1 or P_2 accordingly. This basic step is repeated until the Euclidean distance between P_1 and P_2 is no more than γ. From this and Lemmas 3 and 4, it follows that $d(P_1, P) \leq \gamma$ and $d(P_2, P) \leq \gamma$. The lemma now follows from the observation that the point $\hat{P}$ returned by the algorithm is either P_1 or P_2. ∎

Theorem 1: The set of points U returned by algorithm geometric_DTOA contains at least one point that is within γ of each intersection between $L_{12}(\delta_{12})$ and $L_{13}(\delta_{13})$ that is in the monitoring region and the number of points in U is at most equal to the number of such intersections in the monitoring region. Hence, at least one point of U is within γ of the true source location provided this location is in the monitoring region.

Proof: The theorem follows from Lemma 5 and the observations (a) every segment (or segment portion) of $L_{12}(\delta_{12})$ in the monitoring region is searched, (b) every intersection within the monitoring region is on exactly one of the segments, of L_{12}, and (c) algorithm locate_L_{13} returns at most one point per intersection. ∎

Note that the points in the set U returned by algorithm geometric_DTOA are on the locus $L_{12}(\delta_{12})$. So, for each point $P \in U$, $\Delta_{12}(P) = \delta_{12}$. Since each returned point $P \in U$ is within γ, in R^2 space, of an intersection of $L_{12}(\delta_{12})$ and $L_{13}(\delta_{13})$, it follows that $\Delta_{13}(P) \leq 2\sqrt{2}\gamma$ (Lemma 2). By

changing the condition on the binary search loop of algorithm locate_L_{13}, we can ensure that the returned points are within a specified tolerance of intersection points in δ^2 space or within specified tolerances in both R^2 and δ^2 spaces.

The set I_{sort} may be computed in $O(1)$ time. Let $l = \max\{D_{X2} - D_{X1}, D_{Y2} - D_{Y1}\}$. In computing U, there are altogether up to 6 calls to locate_$L_{13}(x_L, y_L, x_R, y_R)$, and each makes $O(\log(l/\gamma))$ calls to locate_$L_{12}(x, y_L, y_R)$, which in turn can be done in $O(1)$ time. Thus the complexity of algorithm geometric_DTOA is $O(\log(l/\gamma))$, which can be adapted by suitably specifying γ. If the number of basic computational operations is fixed at c, then we have $\gamma < O(l * 2^{-c})$. We note that the inclusion of the case when L_{12} is a vertical ray does not change the asymptotic complexity of our algorithm.

V. Monotonicity of Directional Derivative

In this section, we establish the monotonicity of the directional derivative of Δ_{13} on each segment of $L_{12}(\delta_{12})$. We do this first for the case when L_{12} is not a vertical ray. For this case, we consider explicitly each of the seven regions: (a) top left, (b) inside, (c) bottom right, (d) top, (e) bottom left, (f) bottom, and (g) top right as shown in Figure 1. We show that the directional derivative of $\Delta_{13}(.)$ along the curve $L_{12}(.)$ is monotone in each of these regions: it is positive in regions (a), (b), (f), and (g) and is negative in regions (c), (d), and (e).

We have for $i = 1, 2, 3$,

$$\frac{\partial d(P, S_i)}{\partial x} = \frac{(x - x_i)}{d(P, S_i)} \quad \text{and} \quad \frac{\partial d(P, S_i)}{\partial y} = \frac{(y - y_i)}{d(P, S_i)}.$$

Also, the tangent vector to $L_{12}(\delta_{12})$ at $P = (x, y)$ is given by

$$\begin{bmatrix} -\frac{\partial \Delta_{12}(P)}{\partial y} \\ \frac{\partial \Delta_{12}(P)}{\partial x} \end{bmatrix}$$

So, the directional derivative of $\Delta_{13}(P)$ at P on the locus $L_{12}(\delta_{12}) = \{P | \Delta_{12}(P) = \delta_{12}\}$, for any δ_{12}, is given by

$$\begin{bmatrix} \frac{\partial \Delta_{13}(P)}{\partial x} \\ \frac{\partial \Delta_{13}(P)}{\partial y} \end{bmatrix}^T \circ \begin{bmatrix} -\frac{\partial \Delta_{12}(P)}{\partial y} \\ \frac{\partial \Delta_{12}(P)}{\partial x} \end{bmatrix}$$

$$= \begin{bmatrix} \frac{x-x_1}{d(P,S_1)} - \frac{x-x_3}{d(P,S_3)} \\ \frac{y-y_1}{d(P,S_1)} - \frac{y-y_3}{d(P,S_3)} \end{bmatrix}^T \circ \begin{bmatrix} -\frac{y-y_1}{d(P,S_1)} + \frac{y-y_2}{d(P,S_2)} \\ \frac{x-x_1}{d(P,S_1)} - \frac{x-x_2}{d(P,S_2)} \end{bmatrix}$$

We note that some authors define the directional derivative by doing an inner product with a unit tangent vector rather than with any tangent vector. If we wish to conform to this definition, we must divide the directional derivative as given by the above expression by the quantity $\sqrt{\left(\frac{\partial \Delta_{12}(P)}{\partial x}\right)^2 + \left(\frac{\partial \Delta_{12}(P)}{\partial y}\right)^2}$. Since we are interested only in the sign of the directional derivative, it doesn't matter which of the two definitions we use. We continue with the simpler definition that does not require the use of a unit tangent vector.

We use the following three basic identities extensively in our derivations:

$$\sin\alpha - \sin\beta = 2\sin\left(\frac{\alpha - \beta}{2}\right)\cos\left(\frac{\alpha + \beta}{2}\right)$$

$$\sin\alpha + \sin\beta = 2\sin\left(\frac{\alpha + \beta}{2}\right)\cos\left(\frac{\alpha - \beta}{2}\right).$$

$$\cos\alpha + \cos\beta = 2\cos\left(\frac{\alpha + \beta}{2}\right)\cos\left(\frac{\alpha - \beta}{2}\right).$$

A. Top Left Region

In this case, we have $0 < \gamma_1 + \gamma_2 < \pi$, $0 < \gamma_1 + \gamma_3 < \pi$, and $\gamma_3 > \gamma_2$ as shown in Figure 3. The directional derivative is given by

$$\begin{bmatrix} \frac{x-x_1}{d(P,S_1)} - \frac{x-x_3}{d(P,S_3)} \\ \frac{y-y_1}{d(P,S_1)} - \frac{y-y_3}{d(P,S_3)} \end{bmatrix}^T \circ \begin{bmatrix} -\frac{y-y_1}{d(P,S_1)} + \frac{y-y_2}{d(P,S_2)} \\ \frac{x-x_1}{d(P,S_1)} - \frac{x-x_2}{d(P,S_2)} \end{bmatrix}$$

$$\begin{aligned} &= (-\sin\gamma_1 + \sin\gamma_3)(\cos\gamma_1 + \cos\gamma_2) \\ &\quad + (\cos\gamma_1 + \cos\gamma_3)(\sin\gamma_1 - \sin\gamma_2) \\ &= -\sin(\gamma_1 + \gamma_2) + \sin(\gamma_1 + \gamma_3) + \sin(\gamma_3 - \gamma_2) \\ &= 2\sin\left(\frac{\gamma_1 - \gamma_2 + 2\gamma_3}{2}\right)\cos\left(\frac{\gamma_1 + \gamma_2}{2}\right) - \sin(\gamma_1 + \gamma_2) \\ &= 2\cos\left(\frac{\gamma_1 + \gamma_2}{2}\right) \end{aligned}$$

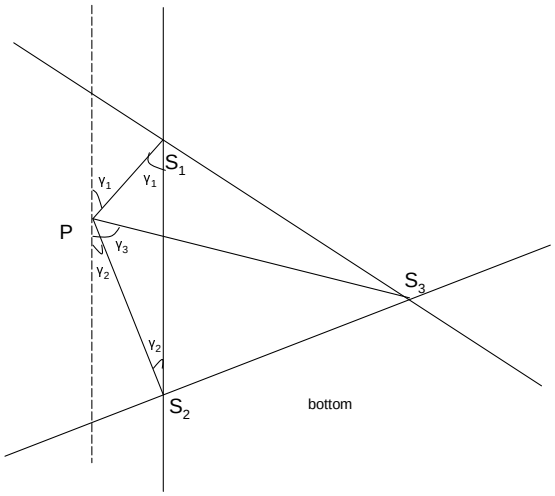

Fig. 3. $P = (x, y)$ is located in the top left region.

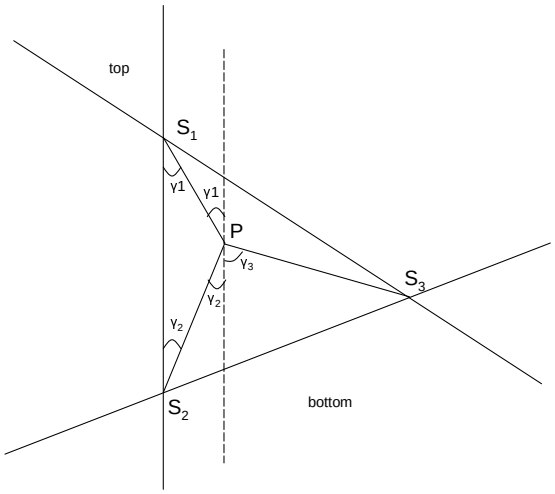

Fig. 4. $P = (x, y)$ is located inside the triangle.

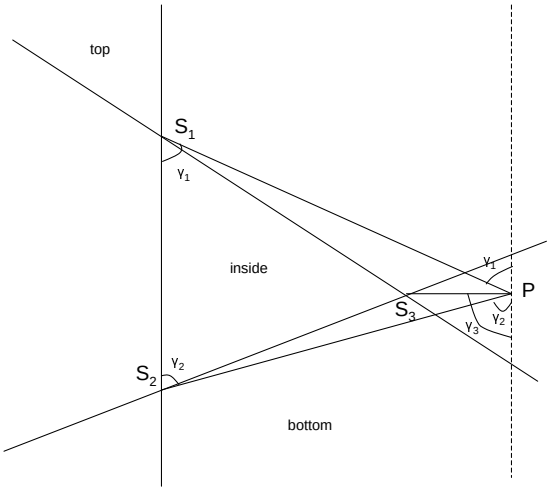

Fig. 5. $P = (x, y)$ is located in the bottom right region.

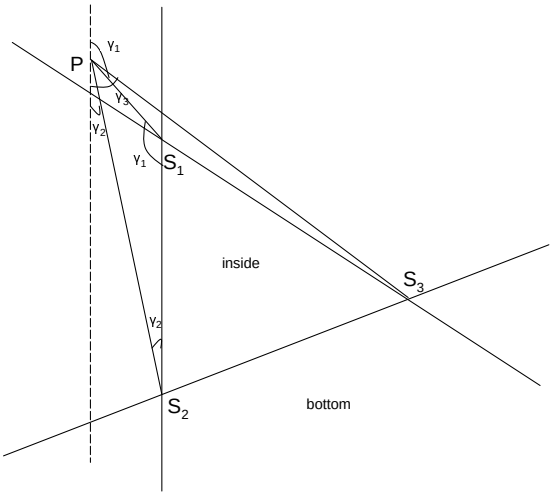

Fig. 6. $P = (x, y)$ is located in the top region.

$$\left[\sin\left(\frac{\gamma_1-\gamma_2+2\gamma_3}{2}\right)-\sin\left(\frac{\gamma_1+\gamma_2}{2}\right)\right]$$
$$= \quad 4\cos\left(\frac{\gamma_1+\gamma_2}{2}\right)\sin\left(\frac{\gamma_3-\gamma_2}{2}\right)\cos\left(\frac{\gamma_1+\gamma_3}{2}\right)$$

We have $0 < \gamma_1 + \gamma_2 < \pi$ which makes the first $\cos$ term positive. We have $\gamma_3 > \gamma_2$ and $0 < \gamma_3 < \pi$. Thus $0 < \frac{\gamma_3-\gamma_2}{2} < \pi/2$, which makes the second $\sin$ term positive. We have $0 < \gamma_1+\gamma_3 < \pi$, which makes the third $\cos$ term positive. Hence the directional derivative is positive.

B. Inside Region

In this case, we have $0 < \gamma_2 + \gamma_3 < \pi$, and $\gamma_3 > \gamma_1$ as shown in Figure 4. The directional derivative of $\Delta(S_1, S_3)$ on the locus $\{(x, y)|\Delta(S_1, S_2) = \delta_{12}\}$, for any δ_{12}, is given by

$$\begin{bmatrix} \frac{x-x_1}{d(P,S_1)}-\frac{x-x_3}{d(P,S_3)} \\ \frac{y-y_1}{d(P,S_1)}-\frac{y-y_3}{d(P,S_3)} \end{bmatrix}^T \circ \begin{bmatrix} -\frac{y-y_1}{d(P,S_1)}+\frac{y-y_2}{d(P,S_2)} \\ \frac{x-x_1}{d(P,S_1)}-\frac{x-x_2}{d(P,S_2)} \end{bmatrix}$$
$$= \quad (\sin\gamma_1+\sin\gamma_3)(\cos\gamma_1+\cos\gamma_2)$$
$$\quad +(-\cos\gamma_1-\cos\gamma_3)(\sin\gamma_1-\sin\gamma_2)$$
$$= \quad \sin(\gamma_1+\gamma_2)+\sin(\gamma_3-\gamma_1)+\sin(\gamma_2+\gamma_3)$$
$$= \quad 2\sin\left(\frac{\gamma_2+\gamma_3}{2}\right)\left[\cos\left(\frac{2\gamma_1+\gamma_2-\gamma_3}{2}\right)+\cos\left(\frac{\gamma_1+\gamma_2}{2}\right)\right]$$
$$= \quad 4\sin\left(\frac{\gamma_2+\gamma_3}{2}\right)\cos\left(\frac{\gamma_1+\gamma_2}{2}\right)\cos\left(\frac{\gamma_1-\gamma_3}{2}\right)$$

We have $0 < \gamma_2 + \gamma_3 < \pi$, which makes the first $\sin$ term positive. Since $\gamma_3 > \gamma_1$, we have $0 < \gamma_1 + \gamma_2 < \pi$, which makes the second $\cos$ term positive. Since $\gamma_3 > \gamma_1$ and $0 < \gamma_3 < \pi$, we have $-\pi/2 < \frac{\gamma_1-\gamma_2}{2} < 0$, which makes the third $\cos$ term positive. Hence the directional derivative is positive.

C. Bottom Right Region

In this case, we have $0 < \gamma_1 + \gamma_3 < \pi$ and $\gamma_3 > \gamma_2$ as shown in Figure 5. The directional derivative of $\Delta(S_1, S_3)$ on the locus $\{(x, y)|\Delta(S_1, S_2) = \delta_{12}\}$, for any δ_{12}, is given by

$$\begin{bmatrix} \frac{x-x_1}{d(P,S_1)}-\frac{x-x_3}{d(P,S_3)} \\ \frac{y-y_1}{d(P,S_1)}-\frac{y-y_3}{d(P,S_3)} \end{bmatrix}^T \circ \begin{bmatrix} -\frac{y-y_1}{d(P,S_1)}+\frac{y-y_2}{d(P,S_2)} \\ \frac{x-x_1}{d(P,S_1)}-\frac{x-x_2}{d(P,S_2)} \end{bmatrix}$$
$$= \quad (\sin\gamma_1-\sin\gamma_3)(\cos\gamma_1+\cos\gamma_2)$$
$$\quad +(-\cos\gamma_1-\cos\gamma_3)(\sin\gamma_1-\sin\gamma_2)$$
$$= \quad \sin(\gamma_1+\gamma_2)-\sin(\gamma_1+\gamma_3)+\sin(\gamma_2-\gamma_3)$$
$$= \quad 2\sin\left(\frac{\gamma_2-\gamma_3}{2}\right)\left[\cos\left(\frac{2\gamma_1+\gamma_2+\gamma_3}{2}\right)+\cos\left(\frac{\gamma_2-\gamma_3}{2}\right)\right]$$
$$= \quad 4\sin\left(\frac{\gamma_2-\gamma_3}{2}\right)\cos\left(\frac{\gamma_1+\gamma_2}{2}\right)\cos\left(\frac{\gamma_1+\gamma_3}{2}\right)$$

Since $\gamma_3 > \gamma_2$ and $0 < \gamma_3 < \pi$, we have $-\pi/2 < \frac{\gamma_2-\gamma_3}{2} < 0$, which makes the $\sin$ term negative. We have $0 < \gamma_1+\gamma_2 < \pi$ and $0 < \gamma_1 + \gamma_3 < \pi$, which makes the last two $\cos$ terms positive. Hence the directional derivative is negative.

D. Top, Bottom Left, Bottom, and Top Right Regions

(d) **Top:** The case of top is identical to the top left region except that $\pi < \gamma_1 + \gamma_3 < 2\pi$ as shown in Figure 6, which makes the third $\cos$ term negative, and hence the directional derivative is negative.

(e) **Bottom Left:** The case of bottom left is identical to the top left region except that $\gamma_2 > \gamma_3$ as shown in Figure 7, which makes the $\sin$ term negative, and hence the directional derivative is negative.

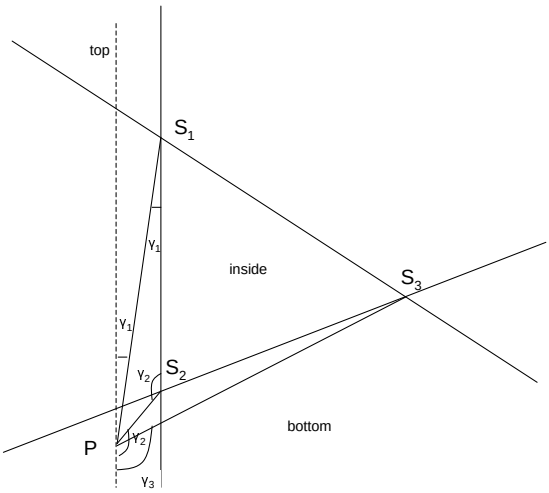

Fig. 7. $P = (x, y)$ is located in the bottom left region.

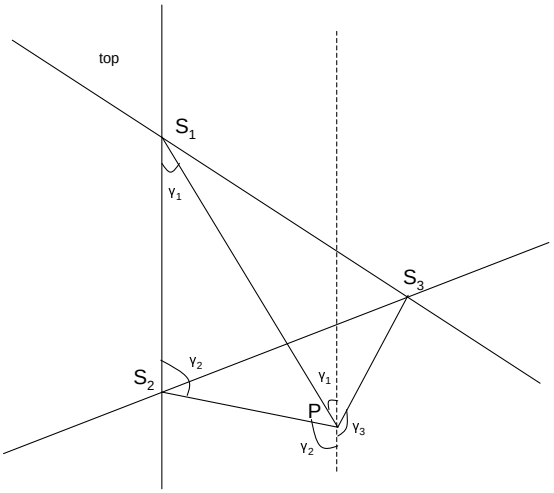

Fig. 8. $P = (x, y)$ is located in the bottom region.

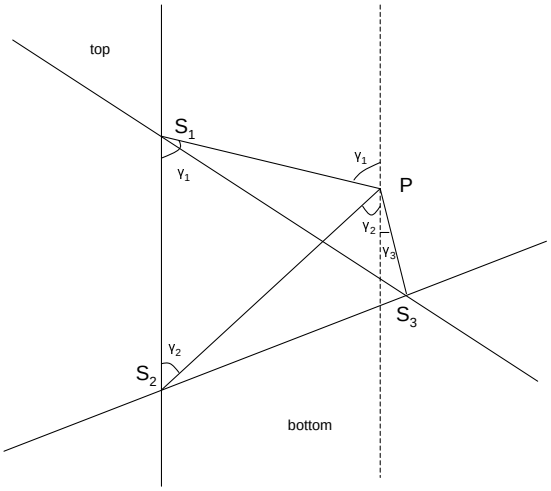

Fig. 9. $P = (x, y)$ is located in the top right region.

(f) **Bottom Region:** For bottom region, the derivation is identical to the case of inside region except that $\pi < \gamma_2 + \gamma_3 < 2\pi$ as shown in Figure 8, which keeps the first $\sin$ term still positive, and hence the directional derivative is positive.

(g) **Top Right:** The case of top right region, as shown in Figure 9, is identical to inside region except that $\gamma_3 < \gamma_1$. Thus we have $0 < \frac{\gamma_1 - \gamma_3}{2} < \pi/2$, which makes the third $\cos$ term still positive, and hence the directional derivative is positive.

Computational results indicating the signs of the directional derivative of randomly generated sources are shown in Figure 10.

When L_{12} is a vertical ray, we need to consider the portion of the segments (a) from S_1 to ∞ and (b) from S_2 to $-\infty$ that lie within the monitoring region. We consider only (a). The proof for (b) is similar. Let P_1 and P_2 be two points on the segment (a). W.l.o.g., assume that P_1 is closer to S_1 than is P_2 (see Figure 11). We see that

$$\begin{aligned}\Delta_{13}(P_1) - \Delta_{13}(P_2) &= (d(P_1, S_1) - d(P_1, S_3)) \\ &\quad -(d(P_2, S_1) - d(P_2, S_3)) \\ &= (d(P_1, S_1) - d(P_2, S_1)) \\ &\quad -(d(P_1, S_3) - d(P_2, S_3)) \\ &= -d(P_1, P_2) - d(P_1, S_3) + d(P_2, S_3) \\ &< 0 \text{ (from the triangle inequality)}\end{aligned}$$

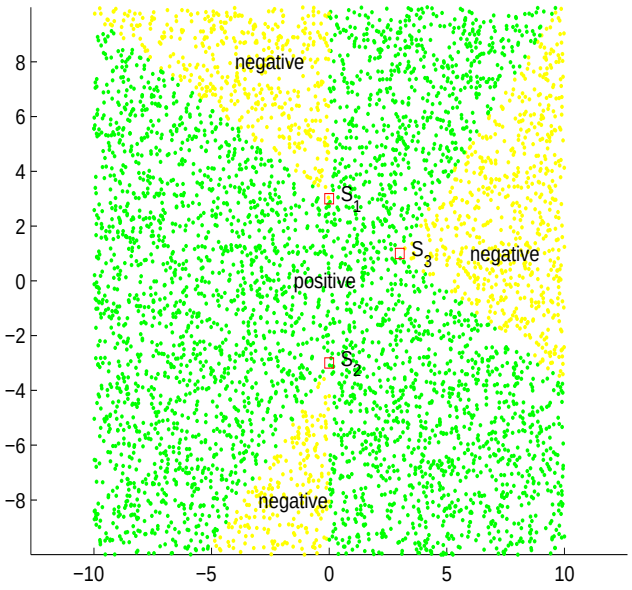

Fig. 10. Source $S = (x, y)$ is randomly selected, and the sign of the directional derivative is computed.

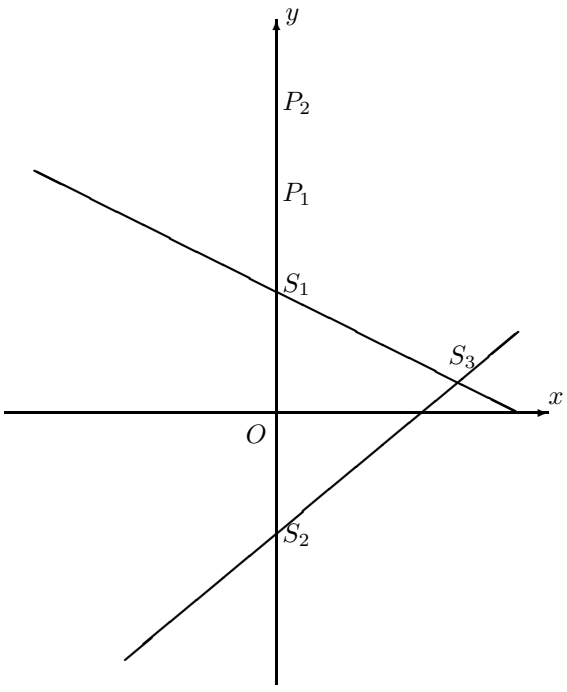

Fig. 11. The degenerate case when $L_{12}(\delta_{12})$ is a vertical ray

Hence, the directional derivative of L_{13} on segment (a) is monotone.

VI. Simulation Results

We compared the performance of our binary search algorithm of Section III versus the linear algebra method of [2], [6], which requires a solution to a quadratic equation. Both algorithms were implemented in Matlab on a Dell Dimension PC with a 2.13 GHz dual-core processor and 2 GB memory. The typical execution times of both methods are only several milliseconds.

Each sensor measurement corresponds to $(1 + f)r$ where r is the actual distance from sensor to source, and f is uniformly randomly generated in the interval $[0, F]$ for a fixed *multiplicative factor* F. While f values are generated independently, sensor error magnitude is proportional to the distance from the sensor to plume origin. Also, the sensor errors are correlated due to the spatial relationships between the sensor locations. a source close to one sensor generates a small error there and larger errors at other sensors, which are located farther away. From these measurements, we computed distance-differences and tested DTOA localization methods. In our experiments, We considered two different scenarios: (i) sensor errors are zero (i.e., $F = 0$), and (ii) sensor errors are greater than zero (i.e., $F > 0$). On a related note, the method of [12] accounts for random errors that are independent Gaussian, and hence is not directly applicable to this case.

Our simulation was conducted in a network of three sensors on a $[0, 100000] \times [0, 100000]$ grid, where location of sources are randomly generated based on the uniform distribution.

A. $F = 0$

We compare the performance of both methods in case that all sensor measurements are accurate. When three sensors form a good triangle, the method of [2], [6] may accurately estimate the source location as shown in [13]. By good triangle, we mean its smallest (largest) angle is not close to 0 (180) degree. However, when three sensors lie in an almost collinear manner, the method of [2] may fail to find a solution as the quadratic equation have imaginary roots for a certain percentage of sources as shown in Figure 12 - 13. For our experiments, each test case may be described by a tuple of $[S1, S2, S3, F, N]$, where $S1$, $S2$, and $S3$ are coordinates of three respective sensors, F is the sensor error, and N is the number of randomly generated sources. Note that we always keep $S1$ closest to the source. Figure 12 gives the number of sources such that [2] returns imaginary roots as well as our method fails to find a solution where $S1 = (0, 0)$, $S2 = (0, 50000)$, $S3 = (0.001, 100000)$, and $N = 12635$. The ratio of the number of such sources against the total number of sources is given. For each test case, we consider various γ_1 and γ_2, where γ_1 and γ_2 are the desired errors acceptable in δ space and R^2 space, respectively. For each $\gamma_1(\gamma_2)$, Figure 12 gives the number of sources whose estimate (excluding the imaginary roots) returned by [2] as well as by our method is within the desired error $\gamma_1(\gamma_2)$ of the actual source. The ratio of the number of such sources to the total number of sources is also given. Figure 13 gives this data for the case where $S1 = (0, 0)$, $S2 = (0, 50000)$, $S3 = (0.0000000000000001, 100000)$, and $N = 12345$.

We note that using our binary search based method versus that of [2] had a great impact on the number of sources that could be estimated. For example, for two test cases shown in Figure 12 and 13 the percentage of sources that leads to imaginary roots in the method of [2] is 1.27% and 27.75%, respectively, whereas our method never fails to find a real solution. Note that almost 28% of sources can't be estimated by the method of [2] in the second test case. On the other hand, the estimate given by our method also shows much better accuracy in both δ space and R^2 space than that of the method of [2]. As shown in Figure 12, to get the ratio of successful estimates to be more than 98% in δ space, the method of [2] needs to set γ_1 to be almost 5000, whereas our method always gives the successful estimate when γ_1 is as small as 0.000001! The similar phenomena is observed in R^2 space as well. When γ_2 is set 10000, the successful ratio of the method of [2] is still slightly less than 97%, whereas our method shows a 100% successful estimate even if we reduce γ_2 by as much as about 10^{10} times! This improvement is even more impressive for the second test case shown in Figure 13, where the estimating quality is improved by more than 10^{11} times!

Another observation is that the estimate accurate in δ space does not necessarily mean it is also accurate in R^2 space. For example, in Figure 12, given γ_1 and γ_2 both to be 1000, the ratio of successful estimate in δ space and R^2 space is 60.93% and 31.69%, respectively, which implies that at least more than 29% of sources that are close to the source in δ space are far away from the source in R^2 space.

B. $F > 0$

When sensor measurements may be inaccurate, a finalization step is added to the end of the original description of our method of Section III. When U is empty, in other words, algorithm locate_L_{13} returns $null$ each time it is invoked, this finalization step chooses as the source estimate the point P of I_{sort}, for which $|\Delta_{13}(P) - \delta_{13}|$ is minimized. This modification is referred to as *binary search with finalization*.

For our experiments, we used 9 test cases, each described by the tuple $[S1, S2, S3, F, N]$. We choose F from {10/100, 5/100, 1/100}. For each test case, we sought various γ_1s and γ_2s. Figures 14 - 22 give the similar data shown in Figure 12. Specifically, the value listed as $\#imaginary$ of our method is for the number of sources such that the original version of our method without finalization fails to find a solution.

We note that when $F > 0$ our method may not find a solution for some sources. However, our method still outperforms that of [2] in terms of the number of sources that could be estimated. In all 9 test cases, the number of such sources by our method without finalization is as many as or less than that of the method of [2]; the reductions were as high as more than 22%.

Also, for all different γ_1s or γ_2s in each test case, the estimate given by our method consistently shows as good as or much better accuracy in both δ space and R^2 space than that of

$S_1=(0,0)$, $S_2=(0,50000)$, $S_3=(0.001,100000)$, F=0, and N=12635								
Method	#imaginary	ratio	δ space			R^2 space		
			γ_1	$\#count$	ratio	γ_2	$\#count$	ratio
Mellen	161	0.0127	100	1328	0.1051	100	722	0.0571
			500	4683	0.3706	500	2384	0.1887
			1000	7698	0.6093	1000	4004	0.3169
			2500	11503	0.9104	2500	7487	0.5926
			5000	12420	0.9830	5000	10512	0.8320
			10000	12473	0.9872	10000	12249	0.9694
Ours	0	0.0	0.00000001	2619	0.2073	0.00000001	246	0.0195
			0.0000001	12011	0.9506	0.0000001	2273	0.1799
			0.000001	12635	1.0	0.000001	12635	1.0

Fig. 12. Data for $S_1=(0,0)$, $S_2=(0,50000)$, $S_3=(0.001,100000)$, and F=0

$S_1=(0,0)$, $S_2=(0,50000)$, $S_3=(0.0000000000000001,100000)$, F=0, and N=12345								
Method	#imaginary	ratio	δ space			R^2 space		
			γ_1	$\#count$	ratio	γ_2	$\#count$	ratio
Mellen	3426	0.2775	100	60	0.0049	100	3	0.00024301
			500	134	0.0109	500	16	0.0013
			1000	210	0.0170	1000	30	0.0024
			2500	378	0.0306	2500	88	0.0071
			5000	572	0.0463	5000	200	0.0162
			10000	935	0.0757	10000	421	0.0341
			50000	6652	0.5388	50000	2070	0.1677
			100000	8919	0.7225	100000	4037	0.3299
Ours	0	0.0	0.00000001	1437	0.1164	0.00000001	225	0.0182
			0.0000001	10546	0.8543	0.0000001	2185	0.1770
			0.000001	12345	1.0	0.000001	12345	1.0

Fig. 13. Data for $S_1=(0,0)$, $S_2=(0,50000)$, $S_3=(0.0000000000000001,100000)$, and F=0

$S_1=(0,0)$, $S_2=(0,50000)$, $S_3=(0.0000000000000001,100000)$, F=10/100, and N=12598								
Method	#imaginary	ratio	δ space			R^2 space		
			γ_1	$\#count$	ratio	γ_2	$\#count$	ratio
Mellen	3944	0.3131	10000	907	0.0720	10000	276	0.0219
			25000	2455	0.1949	25000	861	0.0683
			50000	8034	0.6377	50000	1922	0.1526
			100000	11873	0.9425	100000	3992	0.3169
Ours	2149	0.1706	100	25	0.0020	100	0	0.0
			1000	1183	0.0939	1000	258	0.0205
			2500	4495	0.3568	2500	1067	0.0847
			5000	9273	0.7361	5000	2483	0.1971
			10000	12481	0.9907	10000	5097	0.4046

Fig. 14. Data for $S_1=(0,0)$, $S_2=(0,50000)$, $S_3=(0.0000000000000001,100000)$, and F=10/100

the method of [2]. In particular, when three sensors are almost collinear, the improvement made by our method is significant. For example, when γ_2 is 10000 as shown in Figure 14 -16, the increment of the ratio of successful estimate by our method versus the method of [2] is more than 38%, 62%, and 96%, respectively.

VII. Conclusions

We presented a computational geometric method for the problem of triangulation in plane using measurements of distance-differences. This problem has been extensively studied in the past and several solutions have been deployed, and our re-examination is motivated in part by the requirements of low power sensor nodes. Our method is computationally efficient and adaptive as well as robust with respect to measurement and computational errors. This method is particularly suited for deployment in nodes that adapt their computations in response to power budgets. This method can also be applied when distance measurements are available, and can offer similar advantages over the linear algebraic methods that are often

$S_1=(0,0)$, $S_2=(0,50000)$, $S_3=(0.0000000000000001,100000)$, F=5/100, and N=12341								
Method	#imaginary	ratio	δ space			R^2 space		
			γ_1	$\#count$	ratio	γ_2	$\#count$	ratio
Mellen	3708	0.3005	10000	901	0.0730	10000	304	0.0246
			25000	2230	0.1807	25000	877	0.0711
			50000	6382	0.5171	50000	1881	0.1524
			100000	8721	0.7067	100000	3978	0.3223
Ours	1484	0.1202	100	78	0.0063	100	9	0.00072928
			1000	3048	0.2470	1000	672	0.0545
			2500	8945	0.7248	2500	2129	0.1725
			5000	12236	0.9915	5000	4544	0.3682
			10000	12341	1.0	10000	8029	0.6506

Fig. 15. Data for $S_1=(0,0)$, $S_2=(0,50000)$, $S_3=(0.0000000000000001,100000)$, and F=5/100

$S_1=(0,0)$, $S_2=(0,50000)$, $S_3=(0.0000000000000001,100000)$, F=1/100, and N=12599								
Method	#imaginary	ratio	δ space			R^2 space		
			γ_1	$\#count$	ratio	γ_2	$\#count$	ratio
Mellen	3513	0.2788	10000	951	0.0755	10000	409	0.0325
			25000	2185	0.1734	25000	1063	0.0844
			50000	6829	0.5420	50000	2092	0.1660
			100000	9109	0.7230	100000	4140	0.3286
Ours	650	0.0516	100	840	0.0667	100	156	0.0124
			250	3693	0.2931	250	760	0.0603
			500	8594	0.6821	500	1911	0.1517
			1000	12480	0.9906	1000	4164	0.3305
			2500	12599	1.0	2500	8846	0.7021
			5000	12599	1.0	5000	11691	0.9279
			10000	12599	1.0	10000	12577	0.9983

Fig. 16. Data for $S_1=(0,0)$, $S_2=(0,50000)$, $S_3=(0.0000000000000001,100000)$, and F=1/100

$S_1=(0,0)$, $S_2=(0,50000)$, $S_3=(5000,100000)$, F=10/100, and N=12473								
Method	#imaginary	ratio	δ space			R^2 space		
			γ_1	$\#count$	ratio	γ_2	$\#count$	ratio
Mellen	1649	0.1322	250	52	0.0042	250	19	0.0015
			500	189	0.0152	500	58	0.0047
			1000	725	0.0581	1000	215	0.0172
			2500	3404	0.2729	2500	932	0.0747
			5000	7604	0.6096	5000	2258	0.1810
			10000	10628	0.8521	10000	4804	0.3852
Ours	1647	0.1320	250	78	0.0063	250	19	0.0015
			500	287	0.0230	500	58	0.0047
			1000	1026	0.0823	1000	215	0.0172
			2500	4237	0.3397	2500	932	0.0747
			5000	9093	0.7290	5000	2364	0.1895
			10000	12389	0.9933	10000	5378	0.4312

Fig. 17. Data for $S_1=(0,0)$, $S_2=(0,50000)$, $S_3=(5000,100000)$, and F=10/100

$S_1=(0,0)$, $S_2=(0,50000)$, $S_3=(5000,100000)$, F=5/100, and N=12591								
Method	#imaginary	ratio	δ space			R^2 space		
			γ_1	$\#count$	ratio	γ_2	$\#count$	ratio
Mellen	1007	0.0800	250	228	0.0181	250	51	0.0041
			500	798	0.0634	500	238	0.0189
			1000	2536	0.2014	1000	726	0.0577
			2500	8183	0.6499	2500	2468	0.1960
			5000	11502	0.9135	5000	5009	0.3978
			10000	11579	0.9196	10000	8295	0.6588
Ours	1007	0.0800	250	258	0.0205	250	51	0.0041
			500	921	0.0731	500	238	0.0189
			1000	2886	0.2292	1000	726	0.0577
			2500	9045	0.7184	2500	2468	0.1960
			5000	12514	0.9939	5000	5116	0.4063
			10000	12591	1.0	10000	8776	0.6970

Fig. 18. Data for $S_1=(0,0)$, $S_2=(0,50000)$, $S_3=(5000,100000)$, and F=5/100

$S_1=(0,0)$, $S_2=(0,50000)$, $S_3=(5000,100000)$, F=1/100, and N=12683								
Method	#imaginary	ratio	δ space			R^2 space		
			γ_1	$\#count$	ratio	γ_2	$\#count$	ratio
Mellen	267	0.0211	100	753	0.0594	100	219	0.0173
			250	3582	0.2824	250	965	0.0761
			500	8459	0.6670	500	2374	0.1872
			1000	12337	0.9727	1000	5030	0.3966
			2500	12409	0.9784	2500	9636	0.7598
			5000	12409	0.9784	5000	11926	0.9403
Ours	267	0.0211	100	760	0.0599	100	219	0.0173
			250	3652	0.2879	250	965	0.0761
			500	8672	0.6837	500	2374	0.1872
			1000	12611	0.9943	1000	5030	0.3966
			2500	12683	1.0	2500	9636	0.7598

Fig. 19. Data for $S_1=(0,0)$, $S_2=(0,50000)$, $S_3=(5000,100000)$, and F=1/100

$S_1=(0,0)$, $S_2=(0,100000)$, $S_3=(100000,0)$, F=10/100, and N=12518								
Method	#imaginary	ratio	δ space			R^2 space		
			γ_1	$\#count$	ratio	γ_2	$\#count$	ratio
Mellen	20	0.0016	500	12	0.00095862	500	435	0.0347
			1000	39	0.0031	1000	1520	0.1214
			2500	182	0.0145	2500	6270	0.5009
			5000	624	0.0498	5000	10909	0.8715
			10000	1875	0.1498	10000	12449	0.9945
Ours	0	0.0	250	119	0.0095	250	119	0.0095
			500	193	0.0154	500	435	0.0347
			1000	725	0.0579	1000	1520	0.1214
			2500	3460	0.2764	2500	6272	0.5010
			5000	8504	0.6793	5000	10917	0.8721
			10000	12511	0.9994	10000	12465	0.9958

Fig. 20. Data for $S_1=(0,0)$, $S_2=(0,100000)$, $S_3=(100000,0)$, and F=10/100

$S_1=(0,0)$, $S_2=(0,100000)$, $S_3=(100000,0)$, F=5/100, and N=12483								
Method	#imaginary	ratio	δ space			R^2 space		
			γ_1	$\#count$	ratio	γ_2	$\#count$	ratio
Mellen	4	0.00032	500	44	0.0035	500	1516	0.1214
			1000	112	0.0090	1000	4729	0.3788
			2500	542	0.0434	2500	11042	0.8846
			5000	1542	0.1235	5000	12459	0.9981
Ours	0	0.0	250	183	0.0147	250	415	0.0332
			500	680	0.0545	500	1516	0.1214
			1000	2392	0.1916	1000	4729	0.3788
			2500	8494	0.6804	2500	11043	0.8846
			5000	12478	0.9996	5000	12463	0.9984

Fig. 21. Data for $S_1=(0,0)$, $S_2=(0,100000)$, $S_3=(100000,0)$, and F=5/100

$S_1=(0,0)$, $S_2=(0,100000)$, $S_3=(100000,0)$, F=1/100, and N=12398								
Method	#imaginary	ratio	δ space			R^2 space		
			γ_1	$\#count$	ratio	γ_2	$\#count$	ratio
Mellen	0	0.0	250	149	0.0120	250	6337	0.5111
			500	500	0.0403	500	11029	0.8896
			1000	1382	0.1115	1000	12394	0.9997
			2500	1393	0.1124	2500	12398	1.0
Ours	0	0.0	100	708	0.0571	100	1542	0.1244
			250	3432	0.2768	250	6337	0.5111
			500	8436	0.6804	500	11029	0.8896
			1000	12394	0.9997	1000	12394	0.9997
			2500	12398	1.0	2500	12398	1.0

Fig. 22. Data for $S_1=(0,0)$, $S_2=(0,100000)$, $S_3=(100000,0)$, and F=1/100

used for triangulation based on distances. Furthermore, by computing distance-differences from distance measurements, this method would be less susceptible to one-sided bias errors in distance measurements. This is particularly useful in certain self-localization tasks, where a single sensor is employed to measure distances to reference beacons.

This paper is only a step towards utilizing computational geometric methods for solving localization problems. It would be of future interest to consider extensions of this method for cases where more than three sensors are deployed and multiple measurement sets are provided. It would also be interesting to see if the proposed method can be extended under random noise models. For the special case when S_1, S_2 and S_3 form an acute triangle, a training method was proposed in [11] wherein the localization method can be trained in-situ to account for sensor correlations. The current method can be similarly employed but the training procedure is likely to be more involved. It would be of future interest to explore the "tracking" ability of this method by repeatedly executing it on a stream of distance-difference measurements corresponding to a moving object.

ACKNOWLEDGMENTS

This work is funded by the SensorNet program at Oak Ridge National Laboratory (ORNL) through Office of Naval Research. ORNL is managed by UT-Battelle, LLC for U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

REFERENCES

[1] R. Schmidt, "A new approach to geometry of range difference location," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 8, no. 3, 1972.
[2] G. Mellen, M. Pachter, and J. Raquet, "Closed-form solution for determining emitter location using time difference of arrival measurements," *IEEE Trans. on Aerospace and Electronic Systems*, vol. 39, no. 3, pp. 1056–1058, 2003.
[3] F. Zhao and L. Guibas, *Wireless Sensor Networks*. Elsevier, 2004.
[4] B. Krishnamachari, Ed., *Networking Wireless Sensors*. Cambridge University Press, 2005.
[5] G. Pottie and W. Kaiser, *Principles of Embedded Networked System Design*. Cambridge University Press, 2005.
[6] H. C. Schau and A. Z. Robinson, "Passive source localization employing intersecting spherical surfaces from time-of-arrival differences," *IEEE Trans. on Acoustics, Speech, and Signal Processing*, vol. 35, no. 8, pp. 1223–1225, 1987.
[7] B. T. Fang, "Simple solutions for hyperbolic and related position fixes," *IEEE Transactions on Systems, Man and Cybernetics-B*, vol. 26, no. 9, pp. 748–753, 2005.
[8] A. H. Sayed, A. Tarighat, and N. Khajehnouri, "Network-based wireless location," *IEEE Signal Processing Magazine*, pp. 24–40, July 2005.
[9] F. P. Preparata and I. A. Shamos, *Computational Geometry: An Introduction*. New York: Springer-Verlag, 1985.
[10] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*. Springer-Verlag, 1987.
[11] N. S. V. Rao, "Identification of simple product-form plumes using networks of sensors with random errors," in *International Conference on Information Fusion*, 2006.
[12] Y. T. Chan and K. C. Ho, "A simple and efficient estimator for hyperbolic location," *IEEE Trans. on Image Processing*, vol. 42, no. 8, pp. 1905–1915, 1994.

[13] N. S. V. Rao, X. Xu, and S. Sahni, “A Computational Geometry Method for DTOA Triangulation,” submitted to *International Conference on Information Fusion*, 2007.
[14] M.A. Spirito and A.G. Mattioli, “On the hyperbolic positioning of GSM mobile stations,” in *Proc. International Symposium on Signals, Systems and Electronics*, Sept 1998.
[15] M.A. Spirito, “Further results on GSM mobile station location”, *IEE Electronics Letters*, vol. 35, no. 22, 1999.
[16] S. Fischer, H. Koorapaty, E. Larsson, and A. Kangas, “System performance evaluation of mobile positioning methods,” in *Proc. IEEE Vehicular Technology Conference*, Houston, TX, USA, May 1999.
[17] P-J. Nordlund, F. Gunnarsson, and F. Gustafsson, “Particle filters for positioning in wireless networks,”, in *Invited to EUSIPCO*, Toulouse, France, Sep 2002.