# Design of a distributed localization algorithm to process angle-of-arrival measurements

Jason H. Rife
Tufts University

>> Accepted Article <<

CITATION:
J. Rife, "Design of a distributed localization algorithm to process angle-of-arrival measurements," 2015 *IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, Woburn, MA, 2015, pp. 1-6.
doi: 10.1109/TePRA.2015.7219661

CORRESPONDING AUTHOR:
Jason Rife
jason.rife@tufts.edu



FINANCIAL SUPPORT:
Tufts University sabbatical program

# Design of a Distributed Localization Algorithm to Process Angle-of-Arrival Measurements

Jason Rife
Dept. Mechanical Engineering
Tufts University
Medford, MA
jason.rife@tufts.edu

***Abstract*—This paper presents ANIM, a novel algorithm that uses angle-of-arrival (or *bearing*) measurements for relative positioning of networked, collaborating robots. The algorithm targets shortcomings of existing sensors (e.g., vulnerability of GPS to jamming) by providing a cheap, low-power alternative that can exploit existing, readily available communication equipment. The method is decentralized and iterative, with subgroups of three robots alternately estimating (i) orientation of the plane containing the robots and (ii) the direction of each edge between robots. Simulations demonstrate that ANIM converges reliably and provides accuracy sufficient for practical applications involving coordinated flying robots.**

***Keywords—networks, localization, bearings-only sensing***

## I. Introduction

This paper develops a relative positioning algorithm for networks of flying robots, satellites, or other network-enabled, autonomous systems distributed in 3D space.

Most positioning systems used in robotic and aerospace applications rely on ranging measurements, angular measurements, or both. GPS, for example, allows receivers to locate themselves by measuring range to satellites in orbit [1]. In aviation, many navigation aids exist that use either range measurements (e.g., radar altimeters and DME) or angular measurements (e.g., ILS and VOR) [2]. In the robotic domain, angular sensors (e.g., monocular cameras) and mixed range-and-angle sensors (e.g., lidar) have been used extensively.

The increasingly low cost of robotic platforms is opening new opportunities for network-enabled, multi-robot systems. Demonstrations of collaborative robotic systems (or *swarms*) typically rely on highly specialized sensing equipment to localize robots relative to one other. For instance, fixed-camera systems are often used in controlled indoor environments [3]. Outdoors, specialized GPS systems that process differentially corrected carrier measurements can be used for centimeter-level positioning [4]. Unfortunately, fixed-camera systems are not practical for unstructured environments, nor is GPS always available (particularly indoors, in deep space, and in areas subject to interference or jamming [5]).

Given the limitations of relative positioning systems based on fixed-camera and GPS technologies, new alternatives are starting to emerge. Some proposed alternatives exploit range-based measurements [6],[7] and others, angular measurements [8]. Range-based systems tend to provide higher relative-positioning accuracy, but angular systems have an advantage in that they can be designed to reuse hardware already onboard a robot, thereby potentially saving on size, weight and power. For instance, nearly any existing communication system with multiple antennas can be dual-purposed to infer angle from time-difference-of-arrival for two signals, without requiring custom waveforms or synchronized clocks. This consideration makes the approach very attractive. What is missing is a practical, distributed algorithm that can fully exploit angular measurements within a large network of collaborating robots, such as in a swarm of unmanned aerial vehicles (UAVs). It is the aim of this paper to introduce such an algorithm.

## II. Triangulation (and Trilateration)

This section begins by reviewing the concepts of triangulation and trilateration. Both approaches aim to solve the localization problem, which is to determine the relative positions and/or orientations of a set of robots (or *nodes*). The straight line segments between nodes are called *edges*. Whereas triangulation leverages angles between edges to solve the localization problem, trilateration uses range information along those edges.

As it turns out, neither triangulation nor trilateration, alone, is sufficient to solve the localization problem fully. Angle measurements and distance measurements are each "blind" in a particular way. Triangulation can be used to resolve the relative orientations of objects at each node; however, relative positions obtained in triangulation are known only to an arbitrary scale factor. This is to say that defining a triangle with three angles determines the shape of the triangle (e.g., the length ratio for any pair of edges) but not the size of the triangle (e.g., the absolute length of any one edge). By contrast, trilateration resolves the relative positions of each node but not their relative orientations. In other words, if one vertex of a triangle is fixed, the triangle may be rotated about that vertex to any orientation without altering its edge lengths [9].

In practical terms, both angular and distance information must be combined to obtain a full solution to the localization problem. Sometimes this is done by introducing special nodes (or *anchors*), which have known locations and/or orientations. For instance, GPS uses satellites at known locations as anchors; GPS receivers use these anchors to determine absolute

position in Earth-fixed coordinates. As another example, stereo-vision systems use a pair of cameras as anchors. Knowing the offset and rotation between the cameras enables localization of point features observed by both cameras. Due to space limitations, this paper considers pure triangulation without anchors, and thus resolving scale factor is left as a topic for future work.

### A. Measurements

To define a triangulation (or trilateration) methodology, the first step is to model the sensor measurements. Each measurement captures information about the relative locations of two nodes: $i$ and $j$. If $\mathbf{p}_{ij}$ is the vector between these two nodes, then the nodes are separated by a distance $d_{ij}$, where

$$d_{ij} = \left\| \mathbf{p}_{ij} \right\|. \tag{1}$$

A unit pointing vector $\mathbf{u}_{ij}$ can be used to define the angular relationship from node $i$ to node $j$.

$$\mathbf{u}_{ij} = \mathbf{p}_{ij} \,/\, \left\| \mathbf{p}_{ij} \right\| \tag{2}$$

Relative position is a product of distance and angular data.

$$\mathbf{p}_{ij} = d_{ij} \mathbf{u}_{ij} \tag{3}$$

This nonlinear relationship between distance and angular measurements makes solving the localization problem difficult.

As an aside, it is worth noting two subtleties about the unit pointing vector (2). First, the pointing vector can be expressed in many coordinate systems. As such, each robot might collect angular measurements in a different (e.g., body fixed) coordinate system! For simplicity, we assume here that it is possible for all nodes to convert angular measurements into a common coordinate system (e.g., ENU, or east-north-up). Second, it is significant to note that the unit pointing vector has three coordinates, of which only two are independent. These two independent coordinates are sometimes expressed in other forms, for instance as azimuth and elevation angles, $\theta$ and $\phi$. By convention elevation $\phi$ is the upward component of the pointing vector, and azimuth $\theta$ is a rotation about the gravity axis (e.g., compass heading).

$$\mathbf{u}_{ij} = \begin{bmatrix} \cos\theta\cos\phi & \sin\theta\cos\phi & \sin\phi \end{bmatrix}^T \tag{4}$$

Other representations of angular data are possible, including many defined for rigid body dynamics (e.g. quaternions) [10] and others for projective geometry (e.g. pixel location in an image). This paper will exclusively use unit pointing vectors to describe angular measurements, since all angular measurements can be converted to this form.

### B. Constraints

In addition to measurements, a second type of information that contributes to solving the localization problem is the geometric constraint. Geometric constraints can be used to resolve discrepancies among noisy measurements $\bar{\mathbf{p}}_{ij}$. (The overbar notation distinguishes the noisy measurement from the truth $\mathbf{p}_{ij}$.) For example, in Fig. 1 it is evident that the true edge vectors $\mathbf{p}_{ij}$ must lie in a plane, but that the noisy measurements $\bar{\mathbf{p}}_{ij}$ generally do not obey this constraint.

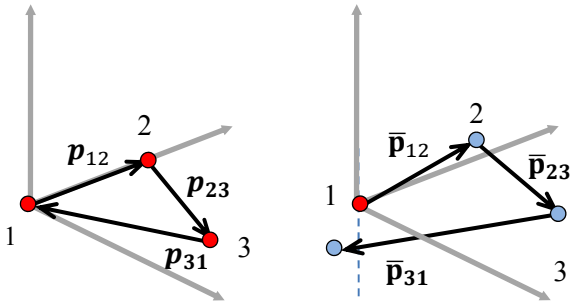


Fig. 1. Any three points define a plane (left), but noisy measurements to those points are not typically coplanar (right).

Geometric constraints relating triangle edges can be defined in many ways, including the useful formulations listed below.

#### 1) Vector Summation

Perhaps the simplest constraint on a triangle's edge vectors is that they should sum to zero. Assuming the nodes in a triangle are labeled {1,2,3}, this constraint is expressed

$$\mathbf{p}_{12} + \mathbf{p}_{23} + \mathbf{p}_{31} = \mathbf{0}. \tag{5}$$

This sum is cyclical around the triangle perimeter. Related forms are obtained by flipping edge direction and sign, noting

$$\mathbf{u}_{ij} = -\mathbf{u}_{ji}. \tag{6}$$

Constraint (5) appears to be linear, but it is in fact a nonlinear combination of range and angle observables, according to (3). It is possible to decouple range and angle information by taking the vector norm of (5) to give:

$$d_{23} = \left\| \mathbf{p}_{23} \right\| = \left\| \mathbf{p}_{13} - \mathbf{p}_{12} \right\|. \tag{7}$$

This equation is the basis for GPS processing, where the known position of a satellite (say, $\mathbf{p}_{12}$) is related to an unknown user position (say, $\mathbf{p}_{13}$) via a measured distance $d_{23}$.

#### 2) Cross-Product Comparison

Geometric constraints using cross-products are common in triangulation. The cross-product of two vectors is orthogonal to those vectors. Hence the cross product makes it convenient to compute the normal to a plane by crossing any two independent edge vectors. For the plane containing nodes {1,2,3}, the unit normal $\mathbf{n}_{123}$ can be computed to be

$$\mathbf{n}_{123} = \frac{\mathbf{u}_{12} \times \mathbf{u}_{23}}{\left\| \mathbf{u}_{12} \times \mathbf{u}_{23} \right\|} = \frac{\mathbf{u}_{23} \times \mathbf{u}_{31}}{\left\| \mathbf{u}_{23} \times \mathbf{u}_{31} \right\|} = \frac{\mathbf{u}_{31} \times \mathbf{u}_{12}}{\left\| \mathbf{u}_{31} \times \mathbf{u}_{12} \right\|}. \tag{8}$$

In the above equation, the denominators ensure unit length; however they also introduce a severe nonlinearity that makes it inconvenient to evaluate (8) in a numerical method. Instead, a related cross-product constraint is more often used:

$$\left(\mathbf{u}_{12}\times\mathbf{u}_{23}\right)\cdot\mathbf{u}_{31}=0\,. \qquad (9)$$

In computer vision this expression is called the *epipolar* constraint [11] and expressed in matrix form,

$$\mathbf{u}_{12}^{T}\mathbf{E}_{23}\mathbf{u}_{31}=0\,, \qquad (10)$$

where $\mathbf{E}_{23}$, a skew-symmetric matrix representing the cross-product with $\mathbf{u}_{23}$, is sometimes labeled the *essential matrix*. In either form, the epipolar constraint is a nonlinear product of three terms (said to be *trilinear*).

*3) Null Space Vector*

Another way to find a normal is to identify the null space basis for vectors lying in a plane. For instance, the null space of the edges connecting nodes {1,2,3}, includes the normal $\mathbf{n}_{123}$:

$$\begin{bmatrix}\mathbf{u}_{12}^{T}\\ \mathbf{u}_{23}^{T}\\ \mathbf{u}_{31}^{T}\end{bmatrix}\mathbf{n}_{123}=\mathbf{0}\,. \qquad (11)$$

The matrix in constraint (11) is not invertible and so an analytic formula cannot easily be written to relate $\mathbf{n}_{123}$ to the unit vectors $\mathbf{u}_{ij}$. As such constraint (11) is rarely used in localization, as using it requires the introduction of extra unknowns (i.e., the components of $\mathbf{n}_{123}$). Constraint (11) is nonetheless quite useful, and it is the backbone of the new algorithm developed later in this paper.

*4) Gravity-Defined Rotation*

Another constraint that invokes orthogonality is the following. Consider a vector $\mathbf{r}_{ij}$ obtained by rotating the elevation $\phi$ of an edge vector $\mathbf{p}_{ij}$ by 90°. Then $\mathbf{r}_{ij}$ and $\mathbf{p}_{ij}$ are orthogonal. Applying this relationship to nodes {1,2,3} gives

$$\begin{bmatrix}\mathbf{r}_{23}^{T}\mathbf{p}_{23}\\ \mathbf{r}_{13}^{T}\mathbf{p}_{13}\end{bmatrix}=\mathbf{0}\,. \qquad (12)$$

Applying the vector summation of (5), this equation becomes

$$\begin{bmatrix}\mathbf{r}_{23}^{T}\\ \mathbf{r}_{13}^{T}\end{bmatrix}\mathbf{p}_{23}=\begin{bmatrix}0\\ -\mathbf{r}_{13}^{T}\mathbf{p}_{12}\end{bmatrix}. \qquad (13)$$

The latter formula is the basis of bearing-only tracking algorithms called *orthogonal vector estimators*. Equation (13) resembles (11), except in that it is inhomogeneous (nonzero right-hand side) and, generally speaking, invertible. For bearings-only tracking applications in 2D, including some radar applications, this formulation creates a set of equation that can be solved uniquely. Augmenting the system with a fourth point (e.g., allowing for three ground stations and one target) allows for target localization in 3D. The algorithm functions by inverting a matrix of $\mathbf{r}_{ij}$ vectors (constructed from measurements) to obtain a state vector relating one ground station to the target (e.g., $\mathbf{p}_{23}$) knowing the relative displacements between ground stations (e.g., $\mathbf{p}_{12}$). Given a gravity vector $\mathbf{g}$, the $\mathbf{r}_{ij}$ are constructed by

$$\mathbf{r}_{ij}=\left(\mathbf{g}\times\mathbf{u}_{ij}\right)\times\mathbf{u}_{ij}\,. \qquad (14)$$

*5) Law of Cosines*

The law of cosines is yet another algebraic relationship that encodes the fundamental structure of a triangle. Noting that the dot product and cosine are related, the law can be written:

$$\left\|\mathbf{p}_{12}\right\|^{2}+\left\|\mathbf{p}_{23}\right\|^{2}-\underbrace{2\mathbf{p}_{23}\cdot\mathbf{p}_{12}}_{\text{cosine term}}-\left\|\mathbf{p}_{31}\right\|^{2}=0\,. \qquad (15)$$

The law of cosines is inherently quadratic in the $\mathbf{p}_{ij}$ variables, a feature which makes (15) well suited for convex optimization employing second-order cone programming [8].

*6) Sum of Angles*

Perhaps the most well-known geometric constraint is that the interior angles of a triangle sum to 180° (or to π radians). Despite its intuitive simplicity, this relationship is highly nonlinear in a 3D space.

$$\begin{aligned}&\operatorname{acos}\left(\mathbf{u}_{12}\cdot\mathbf{u}_{13}\right)+\operatorname{acos}\left(\mathbf{u}_{21}\cdot\mathbf{u}_{23}\right)+\cdots\\ &\operatorname{acos}\left(\mathbf{u}_{32}\cdot\mathbf{u}_{31}\right)-\pi=0\end{aligned} \qquad (16)$$

Multiple solutions and severe nonlinearity hinder convergence and make it difficult to apply this formula to localization.

### C. Optimization-based Estimators

To combine measurements and geometry, triangulation methods are most often framed as optimization programs that minimize a scalar cost function, subject to constraints. In many cases, the cost function is related to a residual difference $\delta\mathbf{u}_{ij}$ between measurements $\bar{\mathbf{u}}_{ij}$ and estimates $\hat{\mathbf{u}}_{ij}$:

$$\delta\mathbf{u}_{ij}=\hat{\mathbf{u}}_{ij}-\bar{\mathbf{u}}_{ij}\,. \qquad (17)$$

For instance, *maximum likelihood estimators* [12] obtain $\hat{\mathbf{u}}_{ij}$ as the argument minimizing the sum of the squared residuals.

$$\hat{\mathbf{u}}_{ij}=\operatorname{argmin}\left(\sum_{(i,j)\in\Omega}\left\|\delta\mathbf{u}_{ij}\right\|^{2}\right) \qquad (18)$$

Here Ω is the set of all measurements between node pairs.

The accuracy of this type of optimal estimate can typically be improved by providing more information in the form of constraints. For instance, one might augment (18) by applying constraint (9) to all measurement triangles.

$$\hat{\mathbf{u}}_{ij} = \operatorname{argmin}\left( \sum_{(i,j)\in\Omega} \left\| \delta\mathbf{u}_{ij} \right\|^2 \right)$$
$$s.t.\ \ \left( \hat{\mathbf{u}}_{ij} \times \hat{\mathbf{u}}_{jk} \right) \cdot \hat{\mathbf{u}}_{ki} = 0 \ \ \forall\ \ (i,j,k) \in \Gamma_{triangle} \tag{19}$$

Here, the set of all node triplets connected by at least two measurements is labeled $\Gamma_{triangle}$. Enforcing this constraint ensures all estimated edges are coplanar, even if the raw measurements are not (as in Fig. 1). Unfortunately, the solution to (19) is inherently centralized, meaning it scales poorly with number of robots; furthermore its non-convexity means that convergence is difficult to guarantee.

To ensure reliable convergence, convex optimization researchers have proposed an alternative formulation of the optimization problem, one based on the law of cosines [8].

$$\hat{\mathbf{u}}_{ij} = \operatorname{argmin}\left( \sum_{(i,j,k)\in\Gamma_{triangle}} \left\| \delta d_{ijk} \right\|^2 \right)$$
$$s.t.\ \ \delta d_{ijk} = \left\| \mathbf{p}_{ij} \right\|^2 + \left\| \mathbf{p}_{jk} \right\|^2 - \left\| \mathbf{p}_{ki} \right\| - 2\left\| \mathbf{p}_{ij} \right\| \left\| \mathbf{p}_{jk} \right\| \hat{\mathbf{u}}_{ij} \cdot \hat{\mathbf{u}}_{jk} \ \ \forall\ \ (i,j,k) \in \Gamma_{triangle} \tag{20}$$

Problem (20) is nonconvex, but a convex approximation exists and can be solved with efficient interior point methods [13]. This approximation is limited in that it is not only centralized, but also inconsistent, such that a second approximation (projection) is required to obtain a usable solution.

To address these limitations, it might be possible to adapt existing quasi-linear triangulation methods, since quasi-linear algorithms have the potential to provide exact, decentralized solutions. Examples of such algorithms include the *orthogonal vector estimator* [12] and the *eight-point algorithm* [11]. The orthogonal vector estimator solves a linear system created from the gravity-defined rotation constraint (13). Unfortunately this method requires knowledge of displacements between fixed anchor nodes, a feature which is problematic for applications involving mobile robot swarms. As for the eight-point algorithm, this quasi-linear algorithm uses the epipolar constraint (10) to estimate the elements of the essential matrix $\mathbf{E}_{23}$ (describing the offset of two stereo cameras) when measurements from both cameras to many points in the environment ( $\mathbf{u}_{2k}$ and $\mathbf{u}_{3k}$ ) are available. The generalization of the eight-point algorithm is nonlinear and nonconvex, essentially equivalent to (19), above.

Given the difficulty of adapting existing triangulation methods to obtain an exact, decentralized solution, a new algorithm is sought.

## III. Triangulation with Alternating Normals

This section introduces a novel localization algorithm that processes angular measurements. The new localization algorithm is dubbed ANIM (for alternating-normals iterative method). It is a *snapshot* method, meaning that the algorithm processes measurements for a given time step without regards to earlier or later measurements. Furthermore, the method is *decentralized*, in the sense that no one node needs to see all measurements acquired at a given time step across the full network. Rather, information is spread through the network via iterative broadcasts until the solution converges.

To implement ANIM, each node is assumed to broadcast its own bearing measurements, or in subsequent iterations, refined estimates of those measurements. Broadcast bearing data are formatted as unit vectors converted into ENU coordinates. Broadcasts do not explicitly relay estimates received from other collaborators; rather, information is distributed solely through the process of each node refining and rebroadcasting its own measurements. Convergence of the iterative process is guaranteed (but the convergence proof is left to a future paper).

Once estimates have sufficiently converged, the iterative stage of the algorithm ends. A second, post-processing stage follows, in which distances between nodes are computed and a scale-free relative position solution is obtained. The two stages of ANIM are described in detail below.

### A. First Stage: Iterating for Geometric Consistency

The first stage of ANIM alternately solves for surface normal and edge vectors. The process begins by setting the initial values for the state estimates equal to the measurements: $\hat{\mathbf{u}}_{ij} = \bar{\mathbf{u}}_{ij}$ . Next, normals are obtained for every triangle with at least two independent edge measurements. To do this, all available edge measurements relating nodes {i,j,k} are compiled in the matrix $\mathbf{A}_{ijk}$. Up to six measurements may be available in all, as in:

$$\mathbf{A}_{ijk} = \begin{bmatrix} \hat{\mathbf{u}}_{ij} & \hat{\mathbf{u}}_{ji} & \hat{\mathbf{u}}_{jk} & \hat{\mathbf{u}}_{kj} & \hat{\mathbf{u}}_{ki} & \hat{\mathbf{u}}_{ik} \end{bmatrix}^T . \tag{21}$$

A multiplicative residual vector $\mathbf{q}_{ijk}$ can be defined such that

$$\mathbf{q}_{ijk} = \mathbf{A}_{ijk}\hat{\mathbf{n}}_{ijk} . \tag{22}$$

Absent noise, the edge vectors are orthogonal to the surface normal $\mathbf{n}$, and so, generalizing (11), the residual $\mathbf{q}_{ijk}$ should be zero. In the presence of noise, the multiplicative residual can be minimized in a least-squares sense.

$$\hat{\mathbf{n}}_{ijk}^T = \operatorname*{argmin}_{\mathbf{n}} \left( \mathbf{n}^T \mathbf{A}_{ijk}^T \mathbf{A}_{ijk} \mathbf{n} \right) \tag{23}$$

An efficient minimization method employs the singular value decomposition, which factors the matrix $\mathbf{A}_{ijk}$ into a diagonal matrix $\mathbf{\Sigma}$ and two unitary matrices $\mathbf{U}$ and $\mathbf{V}$.

$$\mathbf{A}_{ijk} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T \tag{24}$$

It is easy to show (23) is minimized by $\mathbf{v}_{\min}$, the column of $\mathbf{V} \in \mathbb{R}^{3\times 3}$ matched to the smallest singular value in $\mathbf{\Sigma}$.

$$\hat{\mathbf{n}}_{ijk} = \mathbf{v}_{min} \tag{25}$$

After computing the singular value decomposition for every triangle {i,j,k}, the normal for each is obtained from (25).

Now, all edges adjacent to at least two normal vectors can be updated. Consider any edge estimate $\hat{\mathbf{u}}_{ij}$. If that edge is adjacent to $K_{ij}$ normal-vector estimates, those normals can be compiled into a matrix $\mathbf{B}_{ij} \in \mathbb{R}^{3\times K_{ij}}$ where

$$\mathbf{B}_{ij} = \begin{bmatrix} \hat{\mathbf{n}}_{ijk_0} & \hat{\mathbf{n}}_{ijk_1} & \hat{\mathbf{n}}_{ijk_2} & \cdots & \hat{\mathbf{n}}_{ijk_{K_{ij}}} \end{bmatrix}^T. \tag{26}$$

It is useful to identify a corresponding residual $\boldsymbol{\rho}_{ij}$, where

$$\boldsymbol{\rho}_{ij} = \mathbf{B}_{ij}\hat{\mathbf{u}}_{ij}, \tag{27}$$

This residual must be zero for the normal and edge vectors to be geometrically consistent. Due to noise, consistency is not initially achieved, and so the least-squares residual can be minimized to obtain an optimal updated estimate for the edge vector $\hat{\mathbf{u}}_{ij}$. By analogy to (25), the updated optimal value is

$$\hat{\mathbf{u}}_{ij} = \mathbf{x}_{min}. \tag{28}$$

Here $\mathbf{x}_{\min}$ is a column of $\mathbf{X}$ in the singular value decomposition

$$\mathbf{B}_{ij} = \mathbf{W}\mathbf{S}\mathbf{X}^T, \tag{29}$$

with $\mathbf{W}$ and $\mathbf{X}$ being unitary matrices and $\mathbf{S}$ being diagonal. More specifically, $\mathbf{x}_{\min}$ is the column of $\mathbf{X}$ corresponding to the smallest singular value in $\mathbf{S}$.

After updating $\hat{\mathbf{u}}_{ij}$ for every edge with at least two associated normals (e.g. every edge with $K_{ij} \geq 2$), the first stage repeats, alternating the estimation of normal vectors via (25) and pointing vector updates via (28). Iteration continues to convergence.

### *B. Second Stage: Computing Internodal Distances*

When unit edge vector estimates $\hat{\mathbf{u}}_{ij}$ have sufficiently converged and iterations are terminated, the second stage of ANIM obtains edge-length estimates $\hat{d}_{ij}$. These lengths can be computed for any triangle by substituting (3) into (5) to give:

$$\begin{bmatrix} \mathbf{u}_{12} & \mathbf{u}_{23} & \mathbf{u}_{31} \end{bmatrix} \begin{bmatrix} d_{12} \\ d_{23} \\ d_{31} \end{bmatrix} = \mathbf{0}. \tag{30}$$

This equation is homogeneous, and so the solution is not unique. In other words, the solution can only be obtained to an unknown scale factor. However, if at least one edge length in the connected network is known, then the scale factor can be found to provide an unambiguous localization solution.

## IV. Simulation

To assess its capability to support coordinated robotic flight, ANIM was simulated for a representative network of six robots, distributed in three-dimensions. Each robot was modeled as a node in the network, able to measure and communicate its unit pointing vectors to all other robots. Robot measurements were simulated at only a single time step, for the locations listed in TABLE I. All locations given in the table are listed in units of meters of displacement relative to the first robot (Node 1), designated to be at the origin.

A total of 30 noisy measurements were simulated across the network, accounting for five measurements acquired by each of six robots. Measurement noise was simulated by adding a three-dimensional Gaussian random vector to the true measurement and scaling the vector sum back to unit length. The Gaussian random noise vector was modeled to be spherically symmetric. Its standard deviation in any direction was set to one of two values: either $\sigma = 0.02$ (2% of unit vector) or $\sigma = 0.10$ (10% of unit vector).

ANIM converges quickly, with residual errors dropping by two orders of magnitude within 10 first-stage iterations. The results of the solution are shown in Fig. 2 ($\sigma = 0.02$) and Fig. 3 ($\sigma = 0.10$). True locations are represented as blue circles; ANIM estimates for 20 Monte Carlo trials are shown as red dots. The scale ambiguity was not explicitly resolved; however, for purposes of visualization, estimated edge lengths were scaled for best consistency with the truth. Estimates are plotted from the point of view of Robot 1, showing the inferred location of all other robots. Errors tend to align with the pointing vector to each robot *j* (e.g. in the illustration, errors tend to align with the true pointing vectors $\mathbf{u}_{1j}$ from node 1 to each other node *j*). As such, a standard deviation over the sample population was calculated for errors projected in the pointing-vector direction. Standard deviations are listed on the plots in two forms, as a percentage of range (reflecting the scale-factor ambiguity) and, for intuitive clarity, in units of meters (given consistency-based scaling).

Not surprisingly, absolute errors (in meters) are largest for robots on the periphery, as the bearing measurements toward these robots are all acquired from roughly the same direction. Interestingly, percentage errors are larger nearest the reference node (Node 1) since, apparently, dividing by a short baseline magnifies the relative error.

TABLE I. Node Locations Relative to Origin (Node 1)

| | Node 2 | Node 3 | Node 4 | Node 5 | Node 6 |
|---|---|---|---|---|---|
| X (m) | 0 | 300 | -400 | -200 | 400 |
| Y (m) | 0 | -100 | 400 | -200 | 100 |
| Z (m) | 100 | 0 | 100 | 200 | 600 |

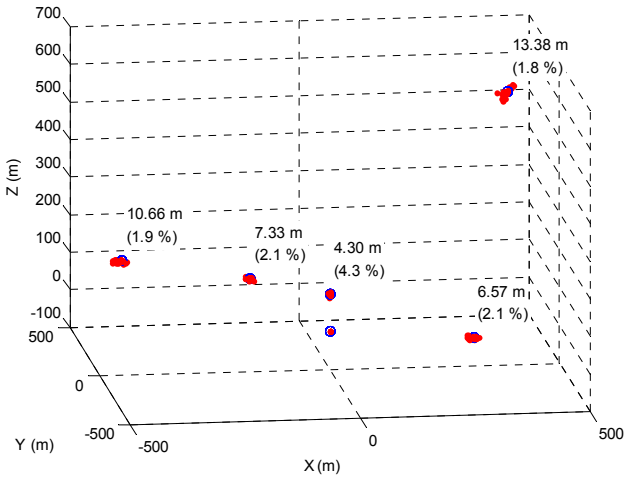


Fig. 2. ANIM results for 20 Monte Carlo trials ($\sigma = 0.02$)

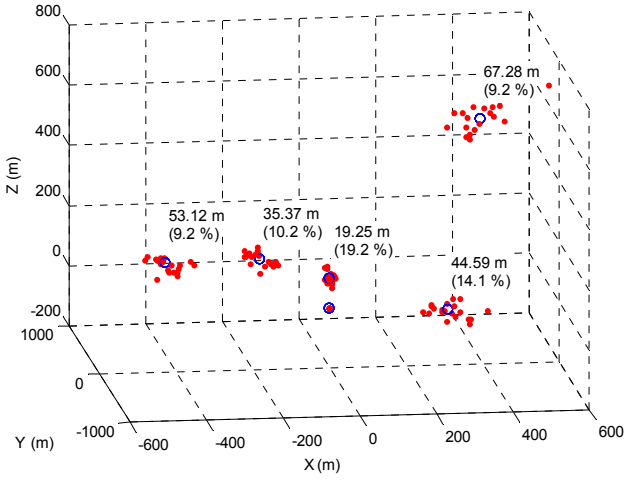


Fig. 3. ANIM results for 20 Monte Carlo trials ($\sigma = 0.10$)

## V. Discussion

The simulation provides preliminary guidance for design of a collaborative navigation system. Simulations suggest that relative position estimates with accuracy of about 2% of range (approximately 10 m on the scale of the simulation) are expected for the lower magnitude of measurement error ($\sigma = 0.02$), which corresponds to an angular measurement error of about 1.5° accuracy. For the larger level of angular measurement error ($\sigma = 0.10$, or about 5.7°), position accuracy is insufficient for most robotic flight applications.

Significant improvements to ANIM positioning accuracy are anticipated when the number of collaborating robots is increased. Testing this hypothesis will be an important topic for future work. Another important topic will be resolving the scale factor ambiguity by adding anchors or additional measurements.

A key result was that ANIM functioned successfully as an iterative algorithm, converging in 100% of trials tested. This preliminary result provides confidence that the algorithm is likely to perform robustly when implemented in hardware in the future. Note that only the iterative part of the algorithm (Stage 1) was evaluated in a distributed manner. For the simulations presented, the distance estimates $d_{ij}$ were computed in a centralized fashion. Future work is needed to de-centralize this processing stage, applying well understood principles for distributed solution of systems of linear equations [14].

## VI. Summary

This paper presented ANIM, a novel localization method for processing angular measurements to obtain the relative 3D positions of a network of collaborating robots. The localization solution is subject to a scale-factor ambiguity, which can be resolved if at least one distance measurement relating node pairs is available. The proposed method is intended to leverage pre-existing robot equipment (e.g. communication systems), so that no additional weight or volume is required to install the navigation sensor. The algorithm inherently leverages distributed computation, which helps minimize communication and processing requirements for individual robots. Preliminary simulations suggest that ANIM consistently converges and provides reasonable accuracy, with localization errors about 2% of separation distance for 1.5° measurement errors.

## References


[1] P. Misra and P. Enge, *Global Positioning System: Signals, Measurements and Performance, Revised 2nd Ed.*, Ganga-Jamuna Press, Lincoln, MA, 2011.

[2] G. Siouris. *Aerospace Avionics Systems: A Modern Synthesis*, San Diego, CA: Academic Press, Inc 1993.

[3] Q. Lindsey, D. Mellinger, and V. Kumar, "Construction with quadrotor teams," *Autonomous Robots*, Vol. 33, No. 3, pp. 323-336, 2012.

[4] Williamson, W., Rios, T., and Speyer, J., "Carrier Phase Differential GPS/INS Positioning for Formation Flight," *American Control Conference (ACC)*, San Diego, CA, 1999.

[5] John A. Volpe National Transportation Systems Center, "Vulnerability Assessment of the Transportation Infrastructure Relying on the Global Positioning System," 29 Aug. 2001.

[6] Nishino, H., Tateishi, K., Ikegami, T., "Inter-Satellite Location Measurement of the Formation Flight Satellite System using UWB signal," *AIAA International Communications Satellite Systems Conference*, AIAA, Seoul, South Korea, 2007.

[7] Chen, T., and Xu, S., "Approach Guidance with Double-Line-of-Sight-Measuring Navigation Constraint for Autonomous Rendezvous," *Journal of Guidance, Control, and Dynamics*, Vol. 34, No. 3, 2011, pp. 678-687.

[8] P. Biswas, H. Aghajan, and Y. Ye, "Semidefinite programming for sensor network localization using angle information," *Proc. IEEE 39th Asilomar Conf. Signals Systems and Computers*, 2005.

[9] J. Rife, "Collaborative positioning for formation flight," *AIAA J. Guidance, Control, and Dynamics*, vol. 46, no. 1, pp. 304-307, 2013.

[10] P. Mitiguy, *Advanced Dynamics and Motion Simulation*, MotionGenesis, 2013.

[11] E. Trucco and A. Verri, *Introductory Techniques for 3-D Computer Vision*, Prentice Hall, 1998.

[12] K. Doğançay and G. Ibal, "Instrumental variable estimator for 3D bearings-only emitter location," *IEEE Int. Conf. Intelligent Sensors, Sensor Networks and Information Processing*, 2005.

[13] P. Biswas and Y. Ye, "Semidefinite programming for ad hoc wireless networking localization," *Proc. Int. Symp. Information Processing in Sensor Networks*, 2004.

[14] U. Khan and J. Moura, "Distributing the Kalman Filter for large-scale systems," IEEE Trans. Signal Processing, vol. 56, no. 10, pp. 4919-4935, 2008.