Network localization from relative bearing measurements

Ryan Kennedy and Camillo J. Taylor

***Abstract*— We present an approach for 2D sensor network localization when only bearing measurements are available and no global coordinate frame is known. Our work builds off of the linear constraint given in *Kennedy et al. (2012)* for sets of nodes that form triangles. We extend that constraint to general networks and present methods for locally optimizing the resulting cost function. We also show how these methods can be used for 3D network localization when the vertical axis is known. The algorithms are evaluated on both synthetic and real datasets, and we also show how they can be applied to the "structure from motion" problem in the field of computer vision.**

I. INTRODUCTION

Sensor network localization is a problem which arises in many circumstances when sensors are deployed in an environment. We focus specifically on the case when sensors are able to take bearing measurements and cannot measure distances. This problem occurs, for example, in robotic networks [1]: if each robot is equipped with a set of antennae, it may be able to tell which direction another robot is located with respect to itself but not how far away it is. This problem is also seen in camera networks since standard cameras can only determine the relative bearings between points, but not depth.

Most of the related work on the topic of network localization deals with distance constraints [2], [3], [4], [5], where nodes can measure relative distances to a set of other nodes. The case when only bearing measurements are used has been studied much less often. In [6], the conditions under which a network that uses only bearing measurements is rigidly-constrained are explored, but an efficient algorithm for actually localizing such a network is not given. A similar situation to the one addressed in this paper is studied in [7], where a quadratic constraint is given for relative angle measurements by introducing additional variables, and an approximate solution is computed using semidefinite programming. In [8], a probabilistic approach is taken.

In *Brand et al. (2004)* [9], a globally-optimal spectral solution is given for the case when a global coordinate frame was known. This was partially extended to networks lacking a global coordinate frame in [10], where it was shown that if three nodes all measure the relative bearings to each other, then this can be written as a linear constraint. However, such triangular constraints are not always common.

A related problem arises in computer vision as the "structure from motion" problem, where camera measurements of points in the world are used to reconstruct the network consisting of both points and cameras [11], [12]. We apply our algorithm to this setup in Section VII-B.

Authors are with the GRASP Lab, Department of Computer and Information Science, University of Pennsylvania. kenry@cis.upenn.edu

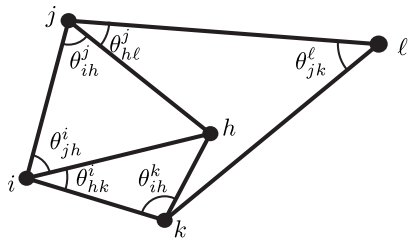


Fig. 1: A network where each node is only able to measure relative angles between others. In this paper we present a method for estimating the layout of such a network.

In this paper, we extend the constraint derived in [10] to arbitrary networks using bearing constraints. Our contributions are as follows:

1) We re-derive the linear constraint of [10] in a much simpler way using complex numbers.
2) We extend the constraint from triangles to general angular constraints by adding additional parameters.
3) We propose two methods for locally optimizing the resulting cost function.
4) We show how our methods can be used for 3D networks when the vertical axis is known by introducing a subsequent linear system.
5) We evaluate our methods on several datasets, including a "structure from motion" dataset from the computer vision community.

II. PROBLEM DESCRIPTION

Given a network of n nodes, we consider the setup where each node is capable of measuring its relative bearing to other nodes within the network but where no global coordinate frame is known. Equivalently, each node is able to measure the angle between other nodes relative to its own position. This setup is depicted in Figure 1. The goal is to estimate the overall layout of the network, up to a similarity transformation, based on this bearing information.

For now, we assume that the network exists on a 2-dimensional plane. An extension of our methods to higher dimensions is non-trivial, although we present an approach that can be used in certain situations in Section VI. The 2D layout of the network of n nodes is then represented by a vector $x = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^T \in \mathbb{C}^n$. The Cartesian

coordinates of a node $x_i \in \mathbb{C}$ are given by the real and imaginary parts of the number x_i, respectively. In [10], [13], a quadratic constraint is derived for this situation. We begin by reviewing this constraint. By using complex numbers rather than pairs of real values, the notation becomes simpler.

Let i be a node which measures the angle between two other nodes j and k, denoted as θ^i_{jk}. We desire an embedding such that the angle between $(x_j - x_i)$ and $(x_k - x_i)$ is θ^i_{jk}. Equivalently, if the vector $(x_j - x_i)$ is rotated by an angle θ^i_{jk} by multiplying it by $e^{i\theta}$, then it should be parallel with $(x_k - x_i)$. If it is subsequently rotated by $\pi/2$, the two vectors should be orthogonal. In other words, $e^{i(\theta^i_{jk}+\pi/2)}(x_j - x_i)$ and $(x_k - x_i)$ should be orthogonal, and thus their inner product should vanish:

$$\text{Re}\left\{\overline{(x_k - x_i)}e^{i(\theta+\pi/2)}(x_j - x_i)\right\} = 0\ . \tag{1}$$

Here, $\overline{\cdot}$ is the complex conjugate operation. Expanding this into a quadratic form gives the equivalent matrix constraint

$$\begin{bmatrix}\bar{x}_i & \bar{x}_j & \bar{x}_k\end{bmatrix}\begin{bmatrix}c+\bar{c} & -c & -\bar{c} \\ -\bar{c} & 0 & \bar{c} \\ -c & c & 0\end{bmatrix}\begin{bmatrix}x_i \\ x_j \\ x_k\end{bmatrix} = 0 \tag{2}$$

for nodes x_i, x_j and x_k where $c = e^{i(\theta^i_{jk}+\pi/2)}$. In this way, each angle constraint can be written as a quadratic constraint on the variables x, and any solution x should satisfy all constraints (in the absence of noise).

Unfortunately, these constraints are not positive semidefinite, as was noted in [10]. Because of this, the resulting optimization problem is quite difficult. In [10], it was found that if three nodes all observe each other, resulting in three angle measurements, θ^i_{jk}, θ^j_{ik} and θ^k_{ji} that sum to π, then there is a linear combination of the three associated quadratic constraints which results in *a positive semidefinite quadratic constraint which is equivalent to the three original constraints*. Furthermore, because a positive semidefinite matrix M can be factored as $M = L^T L$, the constraint is in fact linear.

In other words, [10] shows how angle measurements which form triangles can be written as linear constraints; any triangle which has the same internal angles will satisfy this constraint. However, this constraint is not obvious. The first contribution of this paper is to show how the linear constraint of [10] can be derived in a much simpler manner. We proceed by separately deriving a linear constraint for triangles and subsequently showing that this is the same as the one given in [10].

III. LINEAR CONSTRAINT

Let $t \in \mathbb{C}^3$ be a triangle, where each entry $t_i \in \mathbb{C}$ denotes a single vertex on the plane:

$$t = \begin{bmatrix}t_1 \\ t_2 \\ t_3\end{bmatrix}\ . \tag{3}$$

Our linear constraint is based on the following theorem:

Theorem 1: Let $t = \begin{bmatrix}t_1 \\ t_2 \\ t_3\end{bmatrix} \in \mathbb{C}^3$ be a triangle parameterized by three points in the complex plane. Then, the set of all triangles which are related to t by a similarity transformation (i.e., they have the same internal angles) is given by span $\{t, \mathbb{1}\}$ where $\mathbb{1}$ is the constant vector of 1's.

Proof: First, observe that any rotation and scaling of t in the complex plane can be accomplished by multiplying t by some value $\alpha \in \mathbb{C}$. In particular, if α is written as $\alpha = re^{i\theta}$, then αt is a new triangle produced by scaling t by r and rotating it by the angle θ. Similarly, any translation of t can be written as $t + \beta\mathbb{1}$ for $\beta \in \mathbb{C}$. Thus, any similarity transformation of t can be written as $\alpha t + \beta\mathbb{1}$.

The proof is then straightforward. If $x \in \mathbb{C}^3$ is related to t by a similarity transformation, then x can be written as $x = \alpha t + \beta\mathbb{1}$ for some constants $\alpha, \beta \in \mathbb{C}$ and thus $x \in$ span $\{t, \mathbb{1}\}$. In the reverse direction, if $x \in$ span $\{t, \mathbb{1}\}$, then x can be written as $x = \alpha t + \beta\mathbb{1}$ for some constants $\alpha, \beta \in \mathbb{C}$ and x is therefore similar to t. ■

The linear constraint is then derived as follows. For a triangle $t \in \mathbb{C}^3$, Theorem 1 shows that the space of similarity transformations of t in the complex plane is given by the set span $\{t, \mathbb{1}\}$. Alternatively, let $t_\perp \in \mathbb{C}^3$ be the vector which is orthogonal to both t and $\mathbb{1}$ (there is only one such vector up to scale, since the space is 3-dimensional). Then, any vector $x \in$ span $\{t, \mathbb{1}\}$ is orthogonal to $t_\perp$ and therefore any similarity transformation of t satisfies the constraint

$$t_\perp^* x = 0\ . \tag{4}$$

Furthermore, the vector $t_\perp$ can be calculated easily by taking the complex conjugate of the cross product

$$t_\perp \propto \overline{t \times \mathbb{1}}\ , \tag{5}$$

where the cross product here is computed just as it is with real-valued vectors.

This linear constraint is quite simple, but is the same as the constraint derived by [10], as we show in the following proposition.

Proposition 1: The linear constraint $t_\perp^* x = 0$ is the same as the "triangle constraint" given in Section IV.A of [10], up to scale.

Proof: For a triangle t, the constraint given by [10] is of the form

$$x^T M x = 0\ . \tag{6}$$

In [10], it was shown that the matrix M is positive semidefinite with exactly one non-zero eigenvalue [1]. Thus, M is rank-1, and can be written as

$$M = mm^* \tag{7}$$

using the definitions of positive semidefinite and rank. The constraint is then

$$x^*(mm^*)x = \|m^* x\|_2^2 = 0\ , \tag{8}$$

[1] when using complex numbers; if real numbers are used then there are two equal non-zero eigenvalues.

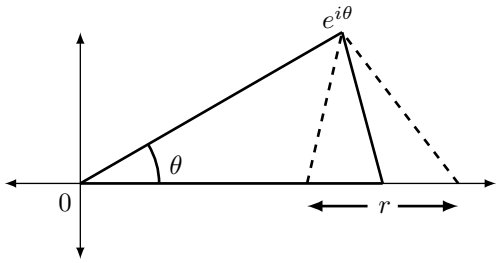


Fig. 2: An angle constraint can be written as a triangle constraint with one unknown parameter. We parameterize this constraint as a triangle with an unknown side length, $r \in \mathbb{R}_+$.

which is equivalent to

$$m^* x = 0 \,. \tag{9}$$

This is of the same form as our constraint in Equation (4), and both equivalently constrain x to be a similarity transformation of t. However, Theorem 1 shows that any such vector must be orthogonal to both t and $\mathbb{1}$. Because the space of triangles is only 3-dimensional, there is only one such vector and they must be the same, up to scale. ■

IV. APPLICATION TO NON-TRIANGULAR PROBLEMS

The linear triangle constraint $t_\perp^* x = 0$, unfortunately, only deals explicitly with triangles. In a general network localization problem, not all constraints will be triangular. If sensors are able to sense others within a fixed radius (for example, due to the sensor's range), then it is relatively likely that triangle constraints do exists due to the properties of Euclidean geometry, as was shown in the experiments in [10]. However, even then some constraints will not form triangles, and other situations may have many fewer or even no triangle constraints at all.

To deal with non-triangular constraints, we consider an angle constraint as a triangle constraint with one unknown angle. In this way, an angle constraint can be written as a triangle constraint *with an unknown parameter*.

To formalize this notion, suppose that a node t_1 measures the angle between t_2 and t_3 as θ. Then, let the triangle $t(r)$ with unknown parameter r be defined as

$$t(r) = \begin{bmatrix} 0 \\ e^{i\theta} \\ r \end{bmatrix} \,, \tag{10}$$

as depicted if Figure 2. Notice that the angle that t_1 measures between t_2 and t_3 is exactly θ for all values of the parameters r. As $r \in (0, \infty)$ is varied, the other two angles change. Thus, the angle constraint can be written as the parameterized linear constraint

$$\exists r \quad \text{s.t.} \quad t(r)_\perp^* x = 0 \,, \tag{11}$$

where

$$t(r)_\perp = \overline{t(r) \times \mathbb{1}} = \begin{bmatrix} e^{-i\theta} - r \\ r \\ -e^{-i\theta} \end{bmatrix} \,. \tag{12}$$

For a network, each linear constraint can be stacked into a matrix $A(r)$, with $r \in \mathbb{R}_+^m$, resulting in a linear system with unknown variables x and unknown parameters r. If done in a naive fashion, however, this may result in a very large number of constraints. For example, consider a network where a node i is able to sense k other nodes. For just node i, we would have k^2 linear constraints by measuring the angle between all pairs. However, only $k-1$ of these constraints are linearly independent. A set of these $k-1$ constraints can be found by choosing one node j as the primary node and measuring only the angle between j and all other $k-1$ nodes that i sees. The choice of which primary node to use is an interesting issue, but we found results to be sufficient by choosing the primary node at random. In any case, the result is a system of m linear equations and n nodes, which we write as $A(r)x = 0$.

V. OPTIMIZATION

The system of constraints under consideration is

$$A(r)x = 0 \,, \tag{13}$$

where both $x \in \mathbb{C}^n$ and $r \in \mathbb{R}_+^m$ are unknown. Solving Equation (13) with respect to both x and r is a difficult problem. In order to impose some tractability on the problem and to deal with noisy systems where no exact solution exists, we instead optimize the cost function

$$\min_{x \in \mathbb{C}^n, r \in \mathbb{R}_+^m} \|A(r)x\|_2^2 = x^* M(r) x \quad \text{s.t.} \quad x^* \mathbb{1} = 0 \,, \tag{14}$$

where $M(r) = A(r)^* A(r)$. The constraint $x^* \mathbb{1} = 0$ is used to avoid the trivial solution of x being a constant vector. Still, this is difficult to optimize globally, and instead we propose two different methods which rely on local optimization. Initialization of these method is discussed in Section V-C.

A. Alternating minimization

Although Equation (14) is difficult to solve with respect to both x and r simultaneously, if either x or r are fixed, the other can be optimally solved for. First, consider r to be fixed. The cost $x^* M(r) x$ is then a positive semidefinite quadratic form, the minimum of which is given by the smallest eigenvector of $M(r)$. However, this vector is simply a constant vector. Because we have constrained $x^* \mathbb{1} = 0$, the optimal solution is the vector which minimizes the cost function and is orthogonal to the constant vector, which is given by the *second* smallest eigenvector of $M(r)$. Because $M(r)$ is very sparse ($A(r)$ has only 3 non-zero entries per row), a sparse eigensolver can be used to calculate the eigenvalues and eigenvectors efficiently, even for large networks.

Now, consider x to be fixed. Each unknown r_i is associated with only row i of $A(r)$. To simplify matters, then, consider just one row of $A(r)$ and the three associated values of x, which for simplicity we denote as x_1, x_2 and x_3. This cost is of the form

$$\left\| \begin{bmatrix} (e^{i\theta} - r) & r & -e^{i\theta} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \right\|_2^2 \,, \tag{15}$$

which can be written as

$$\left\| r(x_2 - x_1) - e^{i\theta}(x_3 - x_1) \right\|_2^2 . \tag{16}$$

By taking the derivative of this expression, setting it to 0 and solving for r (and making use of the fact that r is real-valued, or $\bar{r} = r$), the optimal value of r is found to be

$$r = \text{Re}\left\{ e^{i\theta} \frac{x_3 - x_1}{x_2 - x_1} \right\} . \tag{17}$$

It can also be enforced that $r > 0$ by subsequently setting

$$r = \max\{r, \epsilon\} \tag{18}$$

for small $\epsilon > 0$. In our experiments, we use $\epsilon = 10^{-5}$.

Beginning with an initialization of either x or r, each can be alternately solved for until a local optimum of the cost function is obtained. In Section V-C, we propose a method for initializing r.

B. Eigenvalue minimization

As noted in the previous section, for a fixed value of r the optimal cost is given by the second-smallest eigenvalue of the matrix $M(r)$ with x being the associated eigenvector. Indeed, the cost function can be regarded as a function $f(r) : \mathbb{R}_+^m \to \mathbb{R}$ which maps a vector r to the second-smallest eigenvalue of $M(r)$. A local optimization can be performed using the derivative of this function.

The derivative of an eigenvalue of a matrix with respect to a parameter is relatively straightforward (see Section 17.2 of [14]). Let x be eigenvector corresponding to the second-smallest eigenvalue of $M(r)$. Then, we have

$$\frac{\partial f}{\partial r_i} = x^* \frac{\partial M(r)}{\partial r_i} x \tag{19}$$

$$= x^* \frac{\partial A(r)^*}{\partial r_i} A(r) x + x^* A(r)^* \frac{\partial A(r)}{\partial r_i} x \tag{20}$$

$$= 2\text{Re}\left\{ x^* A(r)^* \frac{\partial A(r)}{\partial r_i} x \right\} . \tag{21}$$

Furthermore, because each r_i is only associated with row i of $A(r)$, the derivative of $A(r)$ with respect to any one variable r_i will result in a matrix of all zeros except for row i. This allows the derivative of f to be computed with respect to all r_i within the same matrix operation as

$$\frac{\partial f(r)}{\partial r} = 2\text{Re}\left\{ \left[\overline{A(r)x} \right] \circ [A'(r)x] \right\} , \tag{22}$$

where $\overline{\cdot}$ is the element-wise complex conjugate, $\circ$ is the Hadamard (element-wise) product, and $A'(r)$ is the matrix formed by taking the derivative of each row i of A with respect to the associated variable r_i.

The algorithm is then a gradient descent optimization: given a vector r, calculate the eigenvector x associated with the second-smallest eigenvalue of $M(r)$, evaluate the derivative $\frac{\partial f(r)}{\partial r}$, take a step in the negative gradient direction, and repeat until convergence. In practice, we use a quasi-Netwon method which estimates the Hessian matrix over time and converges better than standard gradient descent.

C. Initialization

Any local optimization method must be initialized, ideally within the basin of attraction of the global optimum. For the algorithms given in this paper, we begin with an initialization of r (rather than x). The most straightforward way to initialize r is to set $r = \mathbb{1}$. Observe (Figure 2) that setting $r_i = 1$ corresponds to a triangular constraint where both of the unknown angles are given an equal value of $[\pi - \theta_i]/2$. Intuitively, this says that because we don't know the correct values for the other angles, both are given equal weight. More intelligent initializations may be possible depending on the nature of the network being examined, but we found setting $r = \mathbb{1}$ to work well in most cases.

D. Regularization of r

In many real-world scenarios, nodes will tend to sense other nodes which are a similar distance away. This implies that each r_i will be close to 1 and it is unlikely that the optimal values of r will either be very small or very large. This can be formalized by adding a regularization term to the cost function,

$$g(r) = \lambda \|r - 1\|_2^2 , \tag{23}$$

where $\lambda \geq 0$ controls the amount of regularization. For the alternating minimization algorithm given in Section V-A, the optimal value of r can be calculated explicitly as

$$r = \frac{\text{Re}\left\{ e^{i\theta}(x_2 - x_1)^*(x_3 - x_1) \right\} + \lambda}{\|x_2 - x_1\|_2^2 + \lambda} , \tag{24}$$

which reduces to Equation (17) if $\lambda = 0$.

VI. APPLICATION TO $\mathbb{R}^3$ WITH GRAVITY

Unfortunately, the linear triangle constraint given in Equation (4) is not directly generalizable to 3-dimensional space. However, in real-world applications, it may be possible to determine the vertical axis in the world even if the full orientation of the sensors is not known. For example, photographs tend to be taken such that they are aligned vertically or a vertical axis can be determined by lines in the image such as the sides of a building, or in a sensor network each node may be equipped with an accelerometer that can be used to determine the direction of gravity. In any case, if a vertical axis can be determined then a 3D reconstruction can be decomposed into two parts. First, all angles are measured orthogonal to the known vertical direction (i.e., projected onto the 2D ground plane). The 2D layout of the network, as if viewed from above, can be estimated using the methods of this paper and of [10]. Then, given an estimated 2D layout of the network, the vertical positions of the nodes are estimated separately as follows.

Let $x \in \mathbb{C}^n$ be the fixed estimate of the 2D layout of a network, with the vertical height of the nodes, $z \in \mathbb{R}^n$, unknown. Suppose that node i at position x_i observes node j at position x_j to be at a relative angle ϕ_{ij} (Figure 3). This forms a triangle, from which we derive the linear constraint

$$z_j - z_i = \tan(\phi_{ij}) \|x_j - x_i\|_2 . \tag{25}$$

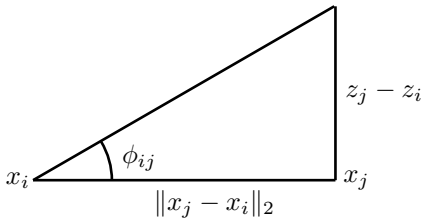


Fig. 3: Visualization of the linear constraint of Equation (25) used for 3D network localization. The 2D locations x of all points are estimated first, and then the vertical angles are used to determine the height of each node.

The full set of observations can in this way be written as a sparse linear system, and a least-squares solution is readily solved for.

VII. EXPERIMENTS

Algorithms We compare the two algorithms presented in this paper, ALTMIN (Section V-A) which performs alternating minimization, and EIGMIN (Section V-B) which locally minimizes the second eigenvalue of $M(r)$. Eigenvalues and eigenvectors were computed using MATLAB's `eigs()` function. For EIGMIN, minimization was done using the implementation of L-BFGS in the `minFunc` package [15].

Error measures Two different error measures are used: the matrix error $\|Ax\|_2^2 = x^*(A^*A)x$, and the 2D root mean squared error (RMSE) after Procrustes alignment of the estimated network to the groundtruth dataset. Note that for sparsely-connected networks, the matrix error may approach zero while the 2D RMSE does not, corresponding to cases where the network is under-constrained and all angle constraints can be satisfied without reaching the true 2D layout of the network.

A. Synthetic dataset

We begin by analyzing performance on synthetically-generated random 2D networks. For each random network layout, a set of 100 points were randomly placed within a square with sides of length $\sqrt{2}/2$. Each node is then able to sense other nodes with a distance R from itself. Note that the square has a diagonal of length 1, and so if $R = 1$ each node is able to see every other.

Without noise Results are shown in Figure 4 for the case of noise-free data. The sensing radius for the nodes is set to values of 0.15, 0.2 and 0.5. This corresponds to each node begin able to see about 11%, 18% and 77% of all other nodes, respectively. As the sensing radius of the nodes increases, the number of constraints grows and fewer iterations are required to reach a low error. In most cases, ALTMIN reaches a lower error much more quickly that EIGMIN. In some cases when $R = 0.15$, the network is under-constrained and the matrix cost $x^*(A^*A)x$ reaches a low error even though the 2D RMSE remains relatively high, especially for EIGMIN. However, when the network is more constrained, ALTMIN nearly always converges more quickly for both error measures.

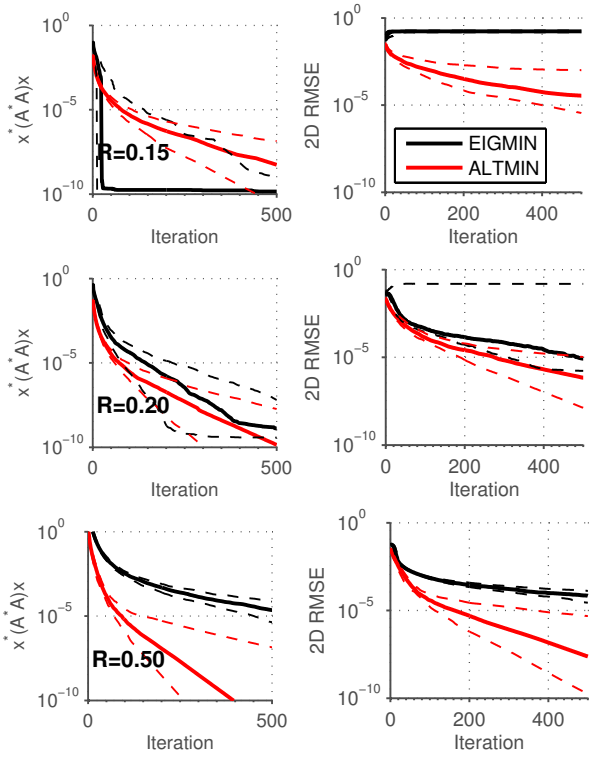


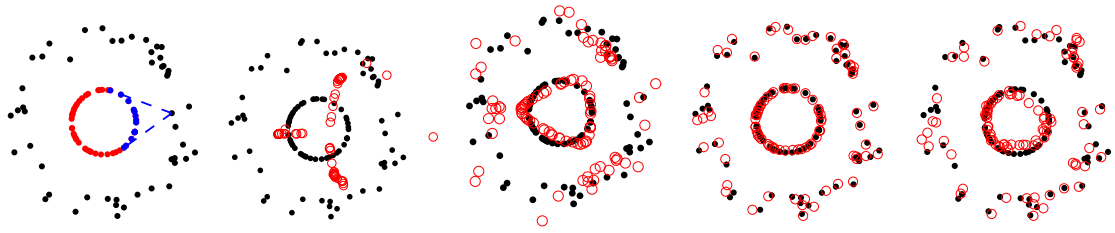
Fig. 4: Comparison of EIGMIN and ALTMIN algorithms on 100 random networks of 100 points where each node has a sensor range of a fixed radius R. We plot the median of all runs, with the 25^{th} and 75^{th} percentiles plotted as dashed lines. **Top row:** $R = 0.15$, **Middle row:** $R = 0.2$, **Bottom row**: $R = 0.5$.

With noise The sensing radius of each node was fixed to $R = 0.2$. For each run of the algorithm, normally-distributed random noise with standard deviation σ was added to each angle measurement. The algorithms were run until the matrix error did not decrease more than 10^{-10}. Results are given in Table I. The algorithm ALTMIN finds solutions that are relatively close to that of the groundtruth, and significantly better than EIGMIN. Although the mean number of iterations that EIGMIN takes to converge is lower, this is due to it occasionally encountering very sub-optimal local minima that take only a few iterations to converge to. ALTMIN does not seem to have this problem, and has much better convergence.

Effect of regularization For random networks, we found that regularization does not improve the solutions. However, this is not true for other types of networks. For example, in Section VII-B, regularization is necessary for finding good solutions.

B. Application to Structure from Motion

Structure from Motion (SFM) is well-studied problem in computer vision where a 3D model of a scene is estimated from a given set of images or a video. The first step in structure from motion estimation is to find points which



(a)

(b)
$\lambda = 0$
3.64×10^{-4}
0.8558

(c)
$\lambda = 10^{-6}$
1.65×10^{-3}
0.3437

(d)
$\lambda = 10^{-4}$
1.96×10^{-3}
0.0714

(e)
$\lambda = 10^{-2}$
6.00×10^{-2}
0.1705

Fig. 5: **Synthetic 2D SFM dataset** **(a)** A circular object with 50 visible points is surrounded by 50 cameras. Each camera is able to see points on the side of the circle within its field of view, as depicted here in blue. **(b)-(e)** Reconstruction using ALTMIN (red circles) using various values of λ for regularization. The numbers shown are the value of λ (top), the matrix cost $x^*(A^*A)x$ (middle), and the 2D RMSE (bottom). The best reconstruction is given for $\lambda \approx 10^{-4}$.

σ		EIGMIN	ALTMIN	Groundtruth
0.1°	$x^*(A^*A)x$	6.40×10^{-3}	6.19×10^{-5}	3.16×10^{-6}
	2D RMSE	6.08×10^{-2}	2.01×10^{-4}	0
	# iterations	304.3	463.5	–
0.2°	$x^*(A^*A)x$	3.65×10^{-3}	9.36×10^{-5}	1.24×10^{-5}
	2D RMSE	4.14×10^{-2}	2.26×10^{-3}	0
	# iterations	340.5	522.3	–
0.5°	$x^*(A^*A)x$	2.99×10^{-3}	2.98×10^{-4}	7.99×10^{-5}
	2D RMSE	4.85×10^{-2}	4.79×10^{-4}	0
	# iterations	310.2	565.4	–
1.0°	$x^*(A^*A)x$	5.17×10^{-3}	1.11×10^{-3}	3.20×10^{-4}
	2D RMSE	6.38×10^{-2}	4.60×10^{-3}	0
	# iterations	290.1	594.4	–
2.0°	$x^*(A^*A)x$	8.89×10^{-3}	4.38×10^{-3}	1.26×10^{-3}
	2D RMSE	6.18×10^{-2}	2.82×10^{-3}	0
	# iterations	312.1	531.4	–
5.0°	$x^*(A^*A)x$	1.64×10^{-2}	2.49×10^{-2}	7.75×10^{-3}
	2D RMSE	9.36×10^{-2}	1.89×10^{-2}	0
	# iterations	346.1	1359.9	–

TABLE I: **Effect of noise** on the algorithms. The noise level σ is varied and the sensing radius is fixed at $R = 0.2$. We show the mean error and number of iterations over 100 trials. Each algorithm was run until the matrix error did not decrease more than 10^{-10}. The last column shows the matrix error of the groundtruth 2D layout (which has zero 2D RMSE). Although EIGMIN takes fewer average iterations, this is mostly caused by occasional local optima that prevent convergence to the groundtruth, as can be seen in the error.

correspond in several frames, which may be done by tracking points in a video sequence or matching features in an unordered set of images. The 3D locations of all cameras and scene points are then estimated using only the 2D (x, y) positions of the points in each image.

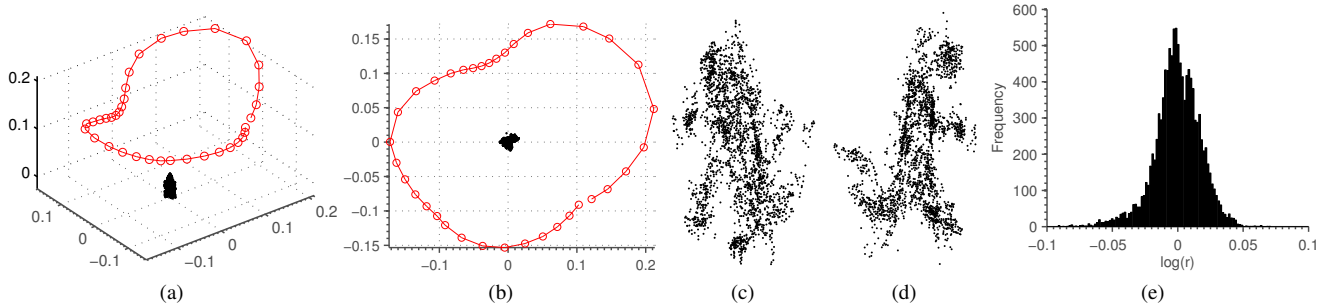
This setup is quite similar to the one studied in this paper. The 3D features in the world are projected along rays to 2D points in an image. In this way, a camera can measure *angles* between scene points, but cannot determine distances. These angles, as measured in the world coordinate system, however, depend on the intrinsic parameters of the camera, which are often unknown and are either estimated or assumed during optimization.

For our purposes, we assume that the intrinsic parameters of all cameras are known. Additionally, we assume that the vertical coordinate axis is known, so that the 2D layout of points can be estimated separately from their vertical heights. These two assumptions are not necessarily that constraining in real-wold situations. In the first case, images can be taken with a pre-calibrated camera so that the camera intrinsics are known. In the second case, images are often taken while the camera is aligned with gravity, or else horizontal and vertical lines on buildings or other objects in the image can be used to estimate the vertical axis. Note that one interesting difference between this setup and a sensor network is that not all points are able to measure angles; only cameras can. This network then has a set of nodes which can make angle measurements (cameras) and a set of nodes which cannot (points). This is a setup that can not be addressed at all by the triangular constraints given by [10] since there are no triangles whatsoever.

One popular method for 3D reconstruction from a set of images involves an algorithm known as "bundle adjustment" [11], [12]. Bundle adjustment is a local optimization method which takes some initial set of positions of all cameras and points and iteratively solves a nonlinear least-squares problem. Typically, this least-squares problem includes all camera parameters. Here, we have factored out these parameters and consider only the 3D layout of all points. Unlike most approaches, our setup treats cameras and scene points identically: they are all just nodes in a network.

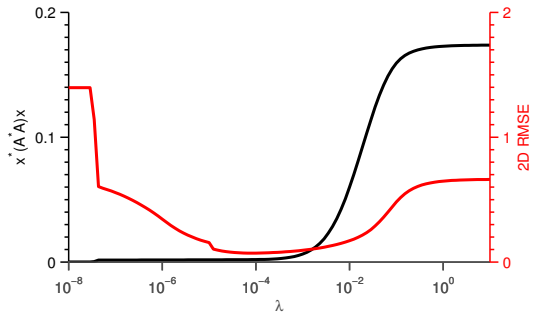
In the following experiments, we focus only on the ALTMIN algorithm because we have found that it converges faster and is less susceptible to local optima.

Synthetic 2D dataset We begin by studying a synthetic 2D dataset. In this dataset, a circular object of radius 1 has 50 visible points and is surrounded by 50 cameras. Each camera can measure angles between points on the circular object which are within its field of view. This setup is depicted in



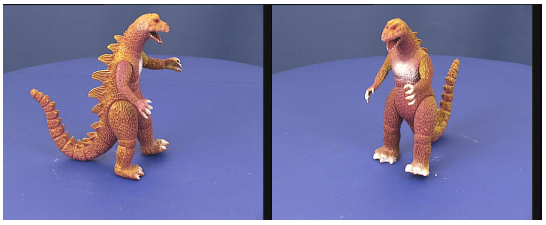
(a) (b) (c) (d) (e)

Fig. 6: **Results on Dinosaur dataset** **(a),(b)** Estimated 3D layout of all cameras (red) and scene points (black). **(c),(d)** Two views of the estimated 3D Dinosaur model. **(e)** Histogram of the log of radius values for the given reconstruction.



(a)

Fig. 7: The matrix cost $x^*(A^*A)x$ and the 2D RMSE for the synthetic SFM dataset shown in Figure 5 as the regularization parameter λ is varied for the algorithm ALTMIN. The best reconstruction is achieved for $\lambda \approx 10^{-4}$.



(a) Frame 5 (b) Frame 10

Fig. 8: Two frames from the Dinosaur dataset used in our experiments.

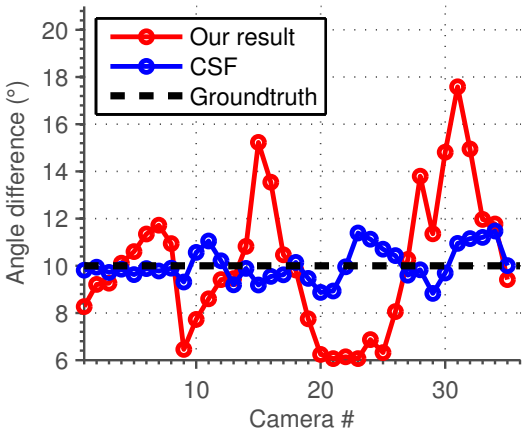


Fig. 9: Variation of camera values in our reconstruction of the dinosaur from an ideal reconstruction. Each camera should rotate 10° around a circle.

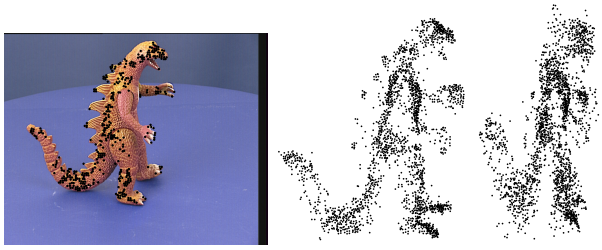
Figure 5a. We also added normally-distributed noise with a standard deviation of $\sigma = 1^\circ$ to each angle measurement.

The reconstruction achieved by ALTMIN with no regularization is shown in Figure 5b, and is quite far from the optimum. Imposing regularization greatly improves the results. The estimated layouts using several different values of λ are shown in Figures 5b-5e. The error for a range of values of λ is given in Figure 7. Even though applying regularization results in a higher value for the matrix error $x^*(A^*A)x$, the resulting 2D RMSE is significantly improved.

Real 3D dataset We use the Dinosaur dataset from [16], depicted in Figure 8, which is a toy dinosaur on a turntable. The sequence consists of 36 frames taken every 10 degrees around the circle. Points in each image were tracked over the video sequence using a KLT tracker [17]. However, by using tracking alone, the first and last frame will not have any constraints in common, and so we tracked points from frame 1 to frame 36 and then back to frame 1. The camera matrices for this dataset are given in [18], which were used to properly measure angles.

Our resulting sequence has 36 frames and 2330 points, and the matrix A is of size 11667×2366. The reconstruction was done using ALTMIN with $\lambda = 10^{-4}$ and subsequently estimating the vertical height of each point. The result is shown in Figure 6. The reconstruction is accurate, although somewhat noisy due to noise in the tracking algorithm. Ideally, the camera centers should lie on the same plane and form a circle around the dinosaur in increments of 10° between cameras. Deviations from this result are shown in Figure 9, where the angles between successive cameras vary from as little as 6° to as much as 17°.

In Figure 10, we compare to another structure-from-motion algorithm, CSF [19]. CSF is based on a result due to Tomasi and Kanade [20] that under certain assumptions on the camera being used, an optimal 3D model can be found



(a) (b) (c)

Fig. 10: **Comparison to CSF [19]** **(a)** Frame 3 with tracked points in black. **(b)** 3D reconstruction using CSF [19]. **(c)** 3D reconstruction using our algorithm.

from a factorization of the measurement matrix. The CSF algorithm also encourages the cameras to move smoothly over time. As Figure 10 shows, both our result and that of CSF are similar, although CSF is somewhat more accurate. This is because our method estimates heights separately from the 2D locations and uses only angles, while CSF directly minimizes reprojection error, leading to a more accurate reconstruction for SFM. In Figure 9, the variation of camera values for CSF is compared to our own algorithm. Again, CSF results in a better reconstruction for this dataset due to it directly optimizing reprojection error.

We should also note that the optimization encountered a local minimum when initialized to $r = \mathbb{1}$, and it had to be re-initialized. This is a familiar problem in the bundle adjustment literature, where an initialization is crucial for accurate results.

This experiment also demonstrates the efficiency of our local optimization. Each iteration of ALTMIN took about 0.05 seconds on a standard laptop computer and used a relatively small amount of memory due to the extreme sparsity of $A(r)$. Our method thus has the potential to be used on very large networks.

VIII. DISCUSSION

The problem of network localization is an important and difficult one. Although relative bearing measurements can be written as quadratic constraints [10], the constraints are difficult to optimize. An improvement was made by [10], where it was shown that constraints can be written for triangles which can be efficiently optimized. Here, we have extended this work to the case of general bearing measurements, even when no triangles are present in the network. In essence, we have traded a set of constraints which are difficult to optimize for a set which have more variables (due to the unknown r values), but which can be locally-optimized quite efficiently. We have shown that this method works well and is very efficient for several different types of networks.

There are several directions for future research. The problem of initialization is a common one in local optimization, and there may be better ways to find an initialization here. For highly-complex problems, an incremental method may be useful as is often done for large-scale SFM problems. It would also be interesting to explore the connection to bundle adjustment in more detail, since this is a very important algorithm in computer vision. Our results provide a new tool which may lead to improvements in this and other domains.

Acknowledgements We thank Haggai Nuchi for helpful discussions regarding the triangle constraints.

REFERENCES

[1] G. Mao, B. Fidan, and B. Anderson, "Wireless sensor network localization techniques," *Computer networks*, vol. 51, no. 10, pp. 2529–2553, 2007.

[2] J. Aspnes, T. Eren, D. Goldenberg, A. Morse, W. Whiteley, Y. Yang, B. Anderson, and P. Belhumeur, "A theory of network localization," *IEEE Transactions on Mobile Computing*, vol. 5, no. 12, pp. 1663–1678, 2006.

[3] D. Moore, J. Leonard, D. Rus, and S. Teller, "Robust distributed network localization with noisy range measurements," in *Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM, 2004, pp. 50–61.

[4] P. Biswas and Y. Ye, "Semidefinite programming for ad hoc wireless sensor network localization," in *Proceedings of the 3rd international symposium on Information processing in sensor networks*. ACM, 2004, pp. 46–54.

[5] P. Biswas, T.-C. Lian, T.-C. Wang, and Y. Ye, "Semidefinite programming based algorithms for sensor network localization," *ACM Transactions on Sensor Networks (TOSN)*, vol. 2, no. 2, pp. 188–220, 2006.

[6] T. Eren, W. Whiteley, and P. Belhumeur, "Using angle of arrival (bearing) information in network localization," in *IEEE Conference on Decision and Control*. IEEE, 2006, pp. 4676–4681.

[7] P. Biswas, H. Aghajan, and Y. Ye, "Integration of angle of arrival information for multimodal sensor network localization using semidefinite programming," in *39th Asilomar Conference on Signals, Systems and Computers*. Citeseer, 2005.

[8] P. Rong and M. L. Sichitiu, "Angle of arrival localization for wireless sensor networks," in *Sensor and Ad Hoc Communications and Networks, 2006. SECON'06. 2006 3rd Annual IEEE Communications Society on*, vol. 1. IEEE, 2006, pp. 374–382.

[9] M. Brand, M. Antone, and S. Teller, "Spectral solution of large-scale extrinsic camera calibration as a graph embedding problem," *European Conference on Computer Vision*, pp. 262–273, 2004.

[10] R. Kennedy, K. Daniilidis, O. Naroditsky, and C. J. Taylor, "Identifying maximal rigid components in bearing-based localization," *IROS*, 2012.

[11] S. Agarwal, N. Snavely, S. M. Seitz, and R. Szeliski, "Bundle adjustment in the large," in *Computer Vision–ECCV 2010*. Springer, 2010, pp. 29–42.

[12] D. Crandall, A. Owens, N. Snavely, and D. Huttenlocher, "Discrete-continuous optimization for large-scale structure from motion," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 3001–3008.

[13] J. Spletzer and C. J. Taylor, "A bounded uncertainty approach to multi-robot localization," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2003, pp. 1258–1264.

[14] J. H. Gallier, *Geometric methods and applications: for computer science and engineering, second edition*. Springer, 2011, vol. 38.

[15] M. Schmidt. (2014) minfunc. [Online]. Available: http://www.di.ens.fr/~schmidt/Software/minFunc.html

[16] A. W. Fitzgibbon and A. Zisserman, "Automatic camera recovery for closed or open image sequences," in *Computer VisionECCV'98*. Springer, 1998, pp. 311–326.

[17] S. Birchfield. (2014) Klt: An implementation of the kanade-lucas-tomasi feature tracker. [Online]. Available: http://www.ces.clemson.edu/~stb/klt

[18] U. of Oxford. (2014) Visual geometry group. [Online]. Available: http://www.robots.ox.ac.uk/ vgg/data/data-mview.html

[19] P. F. Gotardo and A. M. Martinez, "Computing smooth time trajectories for camera and deformable shape in structure from motion with occlusion," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 10, pp. 2051–2065, 2011.

[20] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *International Journal of Computer Vision*, vol. 9, no. 2, pp. 137–154, 1992.