

Linear Distributed Localization of Omni-Directional Camera Networks

William E. Mantzel
willem@rice.edu

Rice University
Houston, TX 77005

Abstract

Localization, estimating the positions and orientations of a set of camera nodes, is a critical first step in camera network applications such as geometric estimation and scene reconstruction. We propose a distributed algorithm for camera network localization between multiple omni-directional cameras with sparse overlapping view structure. This procedure involves a distributed variation of the bundle adjustment algorithm with a linear error metric tailored for omni-directional cameras. The linear structure of the system allows us to show convergence up to a scale factor of the iterative algorithm through an SVD-based approach and provide bounds on convergence rates. Finally, we show simulated and experimental results showing on-par performance with modern centralized techniques.

1 Introduction

Camera networks comprising battery-operated and wirelessly-linked camera nodes in particular show potential to be versatile, inexpensive, and ubiquitous. However, these networks are faced with a constraint. In order to minimize production costs and form factor to enable mass production, the individual camera nodes in this network may be small in size with primitive fixed point processors. This small size constrains the camera node's energy source which in turn limits the amount of node-to-node communication, since these energy costs typically comprise the large majority of total energy consumption on each node.

This constraint is well known to those in the sensor network community who have devoted themselves to successful efforts in energy efficient scalable techniques for statistical estimation, compression, and tracking. However, the bulk of the advancements in this field have been for relatively low bandwidth sensors where the need for distributed algorithms is not as clear. Indeed, scalar field estimation such as temperature or humidity (for example) could alternatively be performed in a centralized fashion with a modest communication requirement (around kilobytes per node per day under rudimentary independent encoding). After reducing the roles of these sensor nodes to wireless data loggers, one could stream this information to some centralized point in a time-sliced communication regime that would enable

these applications to run within an order of magnitude of the shelf life of the batteries themselves.

In contrast, it is clear that sophisticated computer vision applications such as IBR or tele-immersive environments will require aggressive joint compression techniques and distributed algorithms due to their high dimensional data flows. These high-level applications depend on the fundamental tasks of point tracking and correspondence as well as intrinsic and extrinsic calibration. The scope of this paper focuses primarily on extrinsic calibration (localization), while conceding that significant advancements in these related fields are necessary in order to effectively distribute this task across an actual camera network.

We present a localization technique called *Distributed Alternating Localization-Triangulation (DALT)*, a blend of the interleaved bundle adjustment algorithm with Alternating Least Squares, with an error metric suitable for omni-directional cameras [1, 2]. This approach works well for camera networks because it is distributed, widely applicable to a variety of camera network configurations, robust to network outages, and proven to converge to its optimum.

After reviewing the background material and discussing the error metric in Chapter 2, we describe our algorithm in Chapter 3. In Chapter 4, we analyze its theoretical properties, and in Chapter 5, we show its performance through simulation and an experiment.

2 Background

In this section, we will review the necessary background and construct an error criterion useful in localization of an omni-directional camera network.

Each *feature point* will be represented in world coordinates as $X = [X_x \ X_y \ X_z]^T \in \mathbb{R}^3$ and have representation in the camera-centric coordinate system as \bar{X} through the transformation:

$$\bar{X} = \mathbf{R}^T (X - T). \quad (1)$$

for orthogonal rotation matrix $[\mathbf{R}_x \ \mathbf{R}_y \ \mathbf{R}_z] = \mathbf{R} \in \mathbb{R}^{3 \times 3}$.

Under perspective projection, the image coordinates are related by a scale factor of similarity to the point coordinates in the camera's coordinate system. This projection

can be written as:

$$V = \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = \frac{1}{\|\bar{X}\|} \begin{bmatrix} \bar{X}_x \\ \bar{X}_y \\ \bar{X}_z \end{bmatrix}. \quad (2)$$

Note that $V \in \mathbb{S}^2 = \{Y \in \mathbb{R}^3 : \|Y\| = 1\}$, where \mathbb{S}^2 the unit sphere is a natural native projection surface for omnidirectional cameras.

These normalized pinhole image coordinates are related to the actual omni-directional image coordinates by some continuous function. Previous work for conventional planar cameras has led to powerful and practical means for inverting this function with sub-pixel accuracy to obtain the pure homogeneous image coordinates on the hypothetical image surface [3]. Work still remains on calibration of omnidirectional cameras, though some progress in this area has been made as well [4, 5].

After this distortion-inverting function is recovered offline once for each camera, these calibration sets will likely suffice for later deployments, because the propagation properties of light vary only slightly over a wide range of environmental conditions when compared to the measurement precision of modern calibration techniques. Thus, one can rectify or project the image onto any surface they choose (\mathbb{S}^2 in our case).

We now construct an error criterion based on (2) useful in localization. We start with the following three constraints

$$\frac{V_x}{V_y} = \frac{\bar{X}_x}{\bar{X}_y}, \quad (3a)$$

$$\frac{V_y}{V_z} = \frac{\bar{X}_y}{\bar{X}_z}, \quad (3b)$$

$$\frac{V_z}{V_x} = \frac{\bar{X}_z}{\bar{X}_x}, \quad (3c)$$

which can be modified to yield the trio of constraints:

$$V_x \bar{X}_y - V_y \bar{X}_x = 0, \quad (4a)$$

$$V_y \bar{X}_z - V_z \bar{X}_y = 0, \quad (4b)$$

$$V_z \bar{X}_x - V_x \bar{X}_z = 0. \quad (4c)$$

Notice that these constraints are linearly independent and in particular they sum to zero. The nature of this constraint is actually rank 2. This set of constraints can be rewritten as a cross-product operation:

$$V \times \bar{X} = 0. \quad (5)$$

Note that catadioptric cameras (for example) may not be accurately modelled with a single focal center because their constituent rays actually strike several different points on the mirror surface. In this case, one may substitute the metric $V \times (\bar{X} + m(V))$ instead, where $m(V)$ is the shape of

the mirror surface parameterized by the direction vector V . For the purpose of simplicity, this metric will be omitted from the rest of the paper, but remains applicable to future discussion.

In order to address the localization problem for the entire camera network, we will now introduce a new subscript notation for the remainder of this chapter as well as the next chapter. The image coordinates of feature point $p \in \mathbb{P}$ observed by camera $c \in \mathbb{F}$ will be denoted $V_{c,p}$. Note that many such feature points $V_{c,p}$ will not be defined due to visibility constraints and occlusions. We will write $X_p \in \mathbb{R}^3$ for the p th point in the canonical world coordinate system. \mathbf{R}_c and T_c will represent camera c 's orientation and translation. We will denote matrices in boldface and distinguish vectors and scalars by using upper and lowercase respectively. All norms are the ℓ_2 norm.

We now pose the camera network localization problem in terms of the metric (5) given as $\|V \times \bar{X}\|$. More precisely, we are minimizing

$$\begin{aligned} \epsilon &= \sum_{c \in \mathbb{F}} \sum_{p \in \mathbb{P}(c)} \|V_{c,p} \times \bar{X}_{c,p}\|^2 & (6) \\ &= \sum_{c \in \mathbb{F}} \sum_{p \in \mathbb{P}(c)} \sin(\theta_{c,p})^2 \|\bar{X}_{c,p}\|^2 \|V_{c,p}\|^2 \\ &= \sum_{c \in \mathbb{F}} \sum_{p \in \mathbb{P}(c)} \sin(\theta_{c,p})^2 \|\mathbf{R}_c^T (X_p - T_c)\|^2 \\ &= \sum_{c \in \mathbb{F}} \sum_{p \in \mathbb{P}(c)} \sin(\theta_{c,p})^2 \|X_p - T_c\|^2, \end{aligned}$$

where $\theta_{c,p}$ is the angle between the line drawn from camera c to point p , and the observed feature direction vector V for that camera/point pair.

This global linearized omni-directional error may be expressed as the sum of the contributing components for each camera node or each feature point:

$$\epsilon = \sum_{c \in \mathbb{F}} \epsilon_c = \sum_{p \in \mathbb{P}} \epsilon_p.$$

This error metric has the benefit of being intuitive and sensible, while being computationally advantageous because it is linear in both X_p and T_c . The principle drawback to this metric is its dependence on $\|X_p - T_c\|$ so that the minimization will favor solutions where the feature points are closer to the cameras. Fortunately as we will later see, this ‘‘gravitational’’ effect that tries to pull the cameras and points to a single place can be effectively mitigated in most cases.

3 Distributed Localization

Now that we understand the geometrical structure of images, we can discuss the approach towards localizing a camera network. In this section, we will develop the *Distributed Alternating Localization-Triangulation (DALT)* algorithm, a simple but effective linear iterative technique for estimating cameras' positions and orientations.

Algorithm 1 *DALT*

```
1: while localization unknown do
2:   if received localization broadcast of 2 or more linked
     cameras then
3:     use these neighboring localizations to triangulate
       6 or more feature points
4:     initial self-localization from these points
5:   end if
6: end while
7: repeat
8:   broadcast localization estimate of self to linked cam-
     eras
9:   use all available localizations to triangulate as many
     feature points  $X_p$  as possible
10:  localize the camera, keeping orientation fixed
11: until localization estimate converges
```

The pseudocode in Algorithm 1 describes the algorithm that runs on each node at a high level. The algorithm begins when two or more neighboring nodes with initial localization estimates first broadcast their position to other neighboring cameras. (In the following section, we will show that this algorithm converges to the same configuration up to a scale factor regardless of initial configuration, though a better initial estimate will ensure more rapid convergence.) When receiving this broadcast, these other camera nodes then triangulate visible points, use the points' estimates to compute their own initial positions, then broadcast this estimate to their neighbors. Through this algorithm, the triangulation of positions of new feature points lead to the localization of previously unlocalized cameras and vice versa. This algorithm is easily distributable if each camera node localizes itself using triangulated points and then collaborates with nearby camera nodes to triangulate other common points. Now we will give the details behind lines 9, and 10, which form the core of the algorithm, as well as lines 3 and 4, the initialization.

Lines 3, 9: Triangulation. Given estimates for the translations and orientations of at least two cameras that view a common feature point, we estimate that point's 3-D position X by solving a least-squares system.

Accumulating the trio of constraints (5) for each of the $n \geq 2$ cameras gives an over-determined linear system for the $X_p \in \mathbb{R}^3$

$$\begin{bmatrix} [V_{c_1,p}] \times \mathbf{R}_{c_1}^T \\ [V_{c_2,p}] \times \mathbf{R}_{c_2}^T \\ \vdots \\ [V_{c_n,p}] \times \mathbf{R}_{c_n}^T \end{bmatrix}_{3n \times 3} [X_p]_{3 \times 1} = \begin{bmatrix} [V_{c_1,p}] \times \mathbf{R}_{c_1}^T T_{c_1} \\ [V_{c_2,p}] \times \mathbf{R}_{c_2}^T T_{c_2} \\ \vdots \\ [V_{c_n,p}] \times \mathbf{R}_{c_n}^T T_{c_n} \end{bmatrix}_{2n \times 1}, \quad (7)$$

which can be cast as a least squares problem

$$\min \| \mathbf{A} X_p - B \| = \epsilon_p,$$

and solved with a pseudoinverse approach:

$$X_p = \mathbf{A}^\dagger B,$$

where $\mathbf{A}^\dagger \doteq (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T$. This process is illustrated in Figure 1.

Line 10: Localization. With an estimate for the orientation of the camera as well as the locations of two or more feature points, one can localize a camera via a technique that mirrors the preceding operation by interchanging the roles of feature points and camera nodes as follows

$$\begin{bmatrix} [V_{c,p_1}] \times \mathbf{R}_c^T \\ [V_{c,p_2}] \times \mathbf{R}_c^T \\ \vdots \\ [V_{c,p_m}] \times \mathbf{R}_c^T \end{bmatrix}_{3m \times 3} [T_c]_{3 \times 1} = \begin{bmatrix} [V_{c,p_1}] \times \mathbf{R}_c^T X_{p_1} \\ [V_{c,p_2}] \times \mathbf{R}_c^T X_{p_2} \\ \vdots \\ [V_{c,p_m}] \times \mathbf{R}_c^T X_{p_m} \end{bmatrix}_{3m \times 1}. \quad (8)$$

As in triangulation, only two or more points are needed. The duality of these two triangulation operations will be exploited in the following section to prove optimal convergence of this algorithm up to a scale factor.

Line 4: Initial Localization. Given the locations of at least 6 points that are visible to camera c , one can use constraint (5) to estimate the twelve parameters in \mathbf{R} and T , the orientation and translation of the camera.

Because simultaneously estimating \mathbf{R}_c and T_c would involve a bilinear least squares estimation, we temporarily substitute $S \doteq -\mathbf{R}^T T$ for T in order to cast this problem as a linear null space estimation, namely

$$\min_{\|Z\|=1} \|FZ\|^2 = \sum_{p \in \mathbb{P}(c)} \|[V_{c,p}] \times (\mathbf{R}_c^T X_p + S_c)\|^2, \quad (9)$$

where F and Z are defined as:

$$F = \begin{bmatrix} [[V_{c,p_1}] \times \otimes X_{p_1}^T \\ [V_{c,p_2}] \times \otimes X_{p_2}^T \\ \vdots \\ [V_{c,p_m}] \times \otimes X_{p_m}^T] \\ [[V_{c,p_1}] \times \\ [V_{c,p_2}] \times \\ \vdots \\ [V_{c,p_m}] \times] \end{bmatrix}_{3m \times 12} \quad (10)$$

$$Z = \begin{bmatrix} R_c(\cdot) \\ S_c \end{bmatrix} \quad (11)$$

where $R_c(\cdot)$ denotes the vertical concatenation of the columns of \mathbf{R}_c .

This ‘‘camera-from-points’’ technique is a straightforward extension of the Direct Linear Transformation (DLT) to handle omnidirectional images [6]. This equation is a special case of the DLT equation when the linear intrinsic parameters are known.

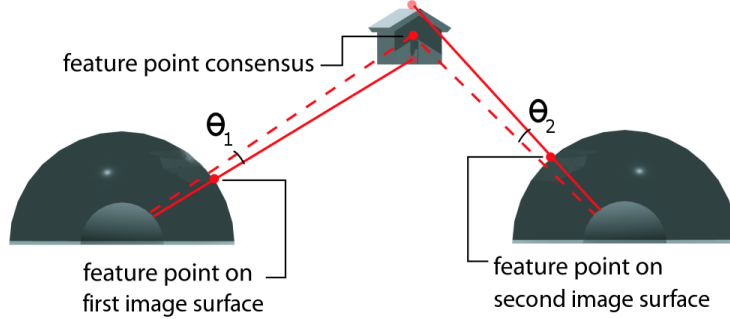


Figure 1: Omni-directional projection model. The feature point shown is triangulated using a least-squares technique that attempts to find the point in 3-D space that is the most consistent with the feature directions from each node that views this point.

The solution to this system is a null space estimate of the $3m \times 12$ matrix F that we obtain by taking the eigenvector corresponding to the smallest eigenvalue of $F^T F$. The sign of the eigenvector is chosen such that the determinant of the orientation matrix is positive. The camera’s orientation and translation \hat{R} and \hat{S} are then determined up to a positive scale factor.

However, the \hat{R} determined in this process may not be orthogonal. We can enforce orthogonality with the singular value decomposition (SVD) of the orientation matrix, given as $U^T \hat{R} V = \text{diag}(\sigma_1, \sigma_2, \sigma_3)$. The closest valid orientation (in Frobenius norm) to the set $\{\alpha \hat{R} : \alpha > 0\}$ is UV^T with corresponding translation $\frac{1}{\sigma} \hat{S}$ for $\sigma = \frac{1}{3}(\sigma_1 + \sigma_2 + \sigma_3)$. Finally, one can substitute back to obtain the localization vector $T = -\frac{1}{\sigma} \hat{R} \hat{S}$.

There are alternative approaches to solving this simple linear system and then forcing the result onto the three-dimensional space of orthogonal matrices with positive determinant. Depending on the available resources, the camera node may be capable of a more sophisticated but perhaps also more demanding computation, or may improve upon this estimate with a gradient descent technique.

Line 10 $\frac{1}{2}$: Optimizing Orientation. While running this iterative algorithm, it is possible to occasionally optimize each camera’s orientation estimate individually while keeping the other parameters fixed using the Levenberg-Marquart algorithm [7]. Using constraint (5) in a least-squares optimization ensures that the new orientation will match those constraints more closely than their previous values, and will not inhibit convergence as discussed below. The Levenberg-Marquart is a robust numerical optimization procedure guaranteed to converge, but the point of convergence may only be a local optimum.

Though this is a nonlinear optimization, it will not present much of a computational challenge to the camera node because only three degrees of freedom are involved (resulting in a 3×3 matrix inverse). Also, each node has

all the information it needs to perform this optimization locally, so no communication is necessary.

4 Convergence Analysis

As we saw in the previous section, by alternating between localization and triangulation on each camera node in an iterative fashion, we are able to compute and propagate across the network a localization estimate that is jointly consistent in some global coordinate system.

In this section we analyze the performance of the *DALT* algorithm. We begin by specifying and analyzing a centralized version of the algorithm and then show that *DALT* is an equivalent distributed algorithm. Subsequently we will analyze the centralized algorithm and through its convergence, prove convergence of the distributed algorithm.

For the purpose of simplicity, we assume that one camera is defined to be the origin. In practice, one may remove this constraint, though the localization will be free to “drift” unless certain anchor points are specified (via GPS for instance).

In this case, the constraints from equations (5) become the linear equation.

$$AT = BX \tag{12}$$

or equivalently,

$$\begin{bmatrix} -A & B \end{bmatrix} \begin{bmatrix} T \\ X \end{bmatrix} = 0 \tag{13}$$

where $X = [X_1^T, X_2^T, \dots, X_P^T]^T$ and $T = [T_1^T, T_2^T, \dots, T_P^T]^T$.

From this point forward, X and T are the vectors containing the vertical concatenation of all of the locations of feature points and cameras. Also, all future subscripts found on these vectors in this section will now refer to the iteration number of the procedure instead of specific camera or point number.

For the sake of brevity, we will make the following statement and its several corollaries without full justification. First, all camera nodes that are reachable with the *DALT* algorithm form a topology of cameras that is localizable to within a single global scale factor of similarity. This in turn implies that the solution to (13) is unique up to a scale factor, or equivalently that the null space of the matrix $[-\mathbf{A} \mid \mathbf{B}]$ is one-dimensional, or equivalently that the intersection $\text{Range}(\mathbf{A}) \cap \text{Range}(\mathbf{B})$ is a one-dimensional subspace. This also implies that both \mathbf{A} and \mathbf{B} have full column rank, or else this solution space would be trivial.

If an initial estimate for the positions of all the cameras T_0 was known, then we could compute a least-squares estimate for X as:

$$X_1 = (\mathbf{B}^T \mathbf{B})^{-1} \mathbf{B}^T \mathbf{A} T_0 = \mathbf{B}^\dagger \mathbf{A} T_0, \quad (14)$$

where \mathbf{B}^\dagger is the pseudoinverse of \mathbf{B} . Note that if $\mathbf{A} T_0 \in \text{Range}(\mathbf{A}) \cap \text{Range}(\mathbf{B})$ initially, then this least-squares solution would produce a solution $[T_0 | X_1]^T = \alpha [T' | X']^T$ of (13) with scalar α where $[T' | X']^T$ is the true solution. If not, we could estimate T_1 from X_1 in a similar manner.

Estimating T from X and X from T iteratively produces a technique for iteratively localizing when the orientations are known. This type of technique is known as *Alternating least-squares* [2]. The n -th iteration of this technique can be formulated as

$$\mathbf{A} T_n = \prod_{i=1}^n (\mathbf{A} \mathbf{A}^\dagger \mathbf{B} \mathbf{B}^\dagger) \mathbf{A} T_0 = \mathbf{C}^n \mathbf{A} T_0 \quad (15)$$

where $\mathbf{C} = \mathbf{A} \mathbf{A}^\dagger \mathbf{B} \mathbf{B}^\dagger$.

Now because each constraint (row) of \mathbf{A} only involves one camera T_c and each constraint of \mathbf{B} only involves one point X_p , they each have a block diagonal structure after appropriate permutation of their constraints (rows) as illustrated here:

$$\begin{bmatrix} \mathbf{A}_{11} \\ \vdots \\ \mathbf{A}_{m1} \\ \\ \vdots \\ \\ \mathbf{A}_{1n} \\ \vdots \\ \mathbf{A}_{mn} \end{bmatrix} T = \begin{bmatrix} \mathbf{B}_{11} & & & \\ & \ddots & & \\ & & \mathbf{B}_{1m} & \\ \mathbf{B}_{21} & & & \\ & \ddots & & \\ & & \mathbf{B}_{2m} & \\ & & & \vdots \\ \mathbf{B}_{n1} & & & \\ & \ddots & & \\ & & & \mathbf{B}_{nm} \end{bmatrix} X.$$

The submatrices are given as:

$$\mathbf{A}_{pc} = \mathbf{B}_{cp} = [V_{c,p}] \times \mathbf{R}_c^T.$$

Because \mathbf{A} is block diagonal, \mathbf{A}^\dagger is merely composed of the pseudoinverse of each of the blocks. Each individual block of \mathbf{A} corresponds to a particular camera and comprises precisely the constraints that each camera node uses to localize itself. Also, each individual block of \mathbf{A}^\dagger corresponds to the distributed localization operation that each one of the cameras performs. Hence the global least-squares localization computation is equivalent to all locally distributed least-squares computations. Similarly, \mathbf{B} is also block diagonal under appropriate permutations of its rows. A similar argument shows that the centralized triangulation operation involving \mathbf{B}^\dagger is equivalent to all distributed least-squares triangulation operations. Hence, the centralized algorithm (*CALT*) is equivalent to the operations performed by the distributed algorithm (*DALT*).

We seek to minimize $\|\mathbf{A} T - \mathbf{B} X\|^2$. However, we must also impose a norm constraint on X and T so that this criterion is not trivially satisfied with X and T at the origin. Requiring $\|\mathbf{A} T\| = \|\mathbf{B} X\| = 1$ is a natural choice since we are minimizing $\|\mathbf{A} T - \mathbf{B} X\|^2$. Also, $\|\mathbf{A} T\|$ and $\|\mathbf{B} X\|$ define norms on X and T since \mathbf{A} and \mathbf{B} have full column rank. Note that we are unable to constrain $\|T\|$ and $\|X\|$ directly since we don't know the relation of these magnitudes, only that $\|\mathbf{A} T\|$ should be roughly $\|\mathbf{B} X\|$. This minimum least-squares can easily be cast as a maximum inner product optimization since

$$\begin{aligned} \epsilon &= \min_{\substack{\|\mathbf{A} T\|=1 \\ \|\mathbf{B} X\|=1}} \|\mathbf{A} T - \mathbf{B} X\|^2 \\ &= \min \left[\sum_{c \in \mathbb{F}} \sum_{p \in \mathbb{P}(c)} \|V_{c,p} \times \bar{X}_{c,p}\|^2 \right] \\ &= \min [\|\mathbf{A} T\|^2 - 2(\mathbf{A} T)^T (\mathbf{B} X) + \|\mathbf{B} X\|^2] \\ &= 2(1 - \max[(\mathbf{A} T)^T (\mathbf{B} X)]). \end{aligned}$$

This maximum inner product is related to a quantity called the first *principle angle* between subspaces. In particular, the cosine of the first principle angle is defined as the maximum inner product between any unit vector from one subspace and any unit vector from another [8]. We leverage the related properties to analyze these minimizers and construct a direct method for solving this system.

To this end, let $\mathbf{A} = \mathbf{Q}_A \mathbf{R}_A$ and $\mathbf{B} = \mathbf{Q}_B \mathbf{R}_B$ be Q-R factorizations of matrices \mathbf{A} and \mathbf{B} such that $\mathbf{Q}_A \in \mathbb{R}^{M \times 3P}$, $\mathbf{Q}_A^T \mathbf{Q}_A = \mathbf{I}$, and \mathbf{R}_A is upper triangular. Likewise, $\mathbf{Q}_B \in \mathbb{R}^{M \times 3F}$, $\mathbf{Q}_B^T \mathbf{Q}_B = \mathbf{I}$ and \mathbf{R}_B is upper triangular. Let $q = \min(3P, 3F)$ and let the SVD of $\mathbf{Q}_A^T \mathbf{Q}_B$ be given as $\mathbf{Y}^T (\mathbf{Q}_A^T \mathbf{Q}_B) \mathbf{Z} = \text{diag}(\sigma_1, \dots, \sigma_q) = \Sigma$.

Theorem 1 (*Centralized ALT algorithm*) *The minimizers of*

$$\min_{\substack{\|\mathbf{A} T\|=1 \\ \|\mathbf{B} X\|=1}} \|\mathbf{A} T - \mathbf{B} X\|^2$$

are given as

$$\begin{aligned}\widehat{T} &= \mathbf{A}^\dagger \mathbf{Q}_A \mathbf{Y}_1 \\ \widehat{X} &= \mathbf{B}^\dagger \mathbf{Q}_B \mathbf{Z}_1.\end{aligned}$$

Proof: This SVD can be characterized as a maximization [8]

$$\begin{aligned}\sigma_1 &= \mathbf{Y}_1^T \mathbf{Q}_A^T \mathbf{Q}_B \mathbf{Z}_1 \\ &= \max_{\|Y\|=1} \max_{\|Z\|=1} Y^T (\mathbf{Q}_A^T \mathbf{Q}_B) Z \\ &= \max_{\|\mathbf{Q}_A Y\|=1} \max_{\|\mathbf{Q}_B Z\|=1} (\mathbf{Q}_A Y)^T (\mathbf{Q}_B Z) \\ &= \max_{\|\mathbf{A}T\|=1} \max_{\|\mathbf{B}X\|=1} (\mathbf{A}T)^T (\mathbf{B}X) \quad (16) \\ &= (\widehat{\mathbf{A}T})^T (\widehat{\mathbf{B}X}) \quad (17)\end{aligned}$$

where equation (16) can be formulated by letting $Y = \mathbf{R}_A T$, $Z = \mathbf{R}_B X$ for invertible \mathbf{R}_A , \mathbf{R}_B (since \mathbf{A} and \mathbf{B} are full rank). Therefore,

$$\begin{aligned}\widehat{\mathbf{A}T} &= \mathbf{Q}_A \mathbf{Y}_1 \\ \widehat{\mathbf{B}X} &= \mathbf{Q}_B \mathbf{Z}_1 \\ \widehat{T} &= \mathbf{A}^\dagger \mathbf{Q}_A \mathbf{Y}_1 \\ \widehat{X} &= \mathbf{B}^\dagger \mathbf{Q}_B \mathbf{Z}_1.\end{aligned}$$

□

Theorem 2 *The CALT algorithm (and hence the equivalent DALT algorithm) converges to its optimal value up to a rate of exponential decay. More explicitly:*

$$\begin{aligned}\frac{1}{\sigma_1^{2n}} T_n &\rightarrow \beta \widehat{T} \\ \frac{1}{\sigma_1^{2n-1}} X_n &\rightarrow \beta \widehat{X}\end{aligned}$$

as $n \rightarrow \infty$.

Proof: It is readily seen that

$$\begin{aligned}\mathbf{A}\mathbf{A}^\dagger &= \mathbf{Q}_A \mathbf{R}_A (\mathbf{R}_A^T \mathbf{Q}_A^T \mathbf{Q}_A \mathbf{R}_A)^{-1} \mathbf{R}_A^T \mathbf{Q}_A^T = \mathbf{Q}_A \mathbf{Q}_A^T \\ \mathbf{B}\mathbf{B}^\dagger &= \mathbf{Q}_B \mathbf{R}_B (\mathbf{R}_B^T \mathbf{Q}_B^T \mathbf{Q}_B \mathbf{R}_B)^{-1} \mathbf{R}_B^T \mathbf{Q}_B^T = \mathbf{Q}_B \mathbf{Q}_B^T.\end{aligned}$$

Using these Q-R decompositions with the SVD $\mathbf{Y}^T (\mathbf{Q}_A^T \mathbf{Q}_B) \mathbf{Z} = \Sigma$ in (15) gives

$$\begin{aligned}\mathbf{A}T_n &= \mathbf{C}^n \mathbf{A}T_0 \quad (18) \\ &= \mathbf{Q}_A ((\mathbf{Q}_A^T \mathbf{Q}_B) (\mathbf{Q}_A^T \mathbf{Q}_B)^T)^n \mathbf{R}_A T_0 \\ &= \mathbf{Q}_A \mathbf{Y} \Sigma^{2n} \mathbf{Y}^T \mathbf{R}_A T_0.\end{aligned}$$

If $\sigma_1 = 1$, then the first principle angle is 0 and there exists an exact solution to $\mathbf{A}T = \mathbf{B}X$. Otherwise, when $1 > \sigma_1 > \sigma_2 > \dots > \sigma_n$, Σ^{2n} (and hence T_n) converges

to zero. However, $\frac{1}{\sigma_1^{2n}} \Sigma^{2n}$ converges to $e_1 e_1^T$ as $n \rightarrow \infty$ so that

$$\begin{aligned}\frac{1}{\sigma_1^{2n}} \mathbf{A}T_n &\rightarrow \mathbf{Q}_A \mathbf{Y}_1 \mathbf{Y}_1^T \mathbf{Q}_A^T \mathbf{Q}_A \mathbf{R}_A T_0 \quad (19) \\ &= \widehat{\mathbf{A}T} [(\widehat{\mathbf{A}T})^T (\mathbf{A}T_0)] \\ &= \beta \widehat{\mathbf{A}T}\end{aligned}$$

where β can be thought of as the result of an inner product that tries to preserve the original estimate of scale inherent in T_0 .

So far, we have shown that under fixed orientations, the estimate of the camera translation vector T converges to its least-squares optimum up to a rate of exponential decay. A similar line of reasoning would show that the feature point position vector X converges to its optimum up to a rate of exponential decay as well. □

To analyze convergence rates, note that the camera position initialization vector T_0 can be decomposed into $\beta \widehat{T} + T_\epsilon$ where $\mathbf{A}T_\epsilon \in (\widehat{\mathbf{A}T})^\perp$. Since $\mathbf{C}^n \widehat{\mathbf{A}T} = \sigma_1^{2n} \widehat{\mathbf{A}T}$ and $\|\mathbf{C}^n \mathbf{A}T_\epsilon\| \leq \sigma_2^{2n} \|\mathbf{A}T_\epsilon\|$, the relative error due to the T_ϵ term dies off as $O\left(\left(\frac{\sigma_2}{\sigma_1}\right)^{2n}\right)$ in the worst case.

Fortunately, in many cases $\sigma_1 \approx 1$ while $\sigma_2 \ll 1$. For example, a 10-camera network viewing a common set of 50 points with feature point error of 5 pixels on a 320×240 display and an error in orientation of 5 degrees, we have $\sigma_1 = 0.9978$ and $\sigma_2 = 0.4997$. In this case, after only 5 DALT iterations the error term would drop to $\sigma_2^{10} \approx 0.001$ of its original value while the scale factor is only attenuated by a factor of $\sigma_1^{10} \approx 0.98$. This reassures us that the decay effect due to the first principle angle will be minor and that we can reduce most of the localization error without significantly affecting the initial scale factor estimate.

In cases when the first principle angle is non-negligible and σ_1 is significantly less than 1, the initial scale factor can be prevented from shrinking to zero by imposing a position constraint on a single camera or point with respect to another camera, point, or origin. These constraints can be included among the other least-squares constraints in the localization process and can be weighted appropriately so that they are not too overbearing on the rest of the DALT algorithm.

5 Results

5.1 DALT Simulation

In order to evaluate DALT through simulation, we use the scale and coordinate system invariant measure of error, the cRMSd metric, which we denote Δ [9]. Note that the DALT algorithm already produces an optimal estimate under the ϵ metric mentioned above in equation (6). We will further

show below that optimizing the latter constraint still performs reasonably well when evaluated using this cRMSd metric.

The simulation was performed as follows. A set of points are randomly chosen from inside the unit sphere to represent the feature points. The n cameras were then placed in a random configuration outside the unit sphere.

For purposes of simulation, we used a simple graph overlay model, imposing artificial visibility constraints on the cameras so that each camera overlaps with at most m of its neighbors. For given cameras, $c, c' \in \{1, 2, \dots, n\}$, these two cameras only share common visible points if $|c - c'| \leq m/2$. Each camera views at least a set of six points specific to that camera called its primary points. In addition, each camera also views the primary points belonging to the other cameras it overlaps with.

The *DALT* algorithm is evaluated by its centralized equivalent as given above in Section 4. Because of the multiple parameters involved that determine the performance, we have shown a few results in Table 5.1 rather graphing them. Each row represents the average result of fifty trials for each algorithm. The results from this simulation seem to suggest that the accuracy in orientation and feature point tracking play a much more critical role when there is a low amount of connectivity in the camera network.

Table 1: *DALT* simulated performance on sparse camera networks. The simulation is run for n cameras whose view overlaps with m other cameras. The error in feature points and initial orientation is σ_u and σ_ω . The resulting performance measured in the Δ error metric against ground truth is listed in column Δ_D . Because of the scale invariant nature of localization, Δ_D is referenced to the baseline of the environment, and is therefore dimensionless.

σ_ω	σ_u	n	m	Δ_D
0.003	0.003	20	12	1.0e-2
0.001	0.003	20	12	8.9e-3
0.003	0.003	20	8	1.3e-2
0.003	0.003	20	6	1.6e-2
0.001	0.001	20	6	3.4e-3
0.010	0.010	20	4	2.5e-1
0.003	0.003	10	4	1.3e-2
0.003	0.003	50	4	1.4e-1

5.2 Experiment on City Scanning Dataset

To evaluate this algorithm on real omni-directional image data, we used images from MIT’s online omni-directional image database. Of these 566 omni-directional nodes, the Green Court dataset consists of 30 nodes over a region of

roughly 80m by 115m [10]. Of these 30, we present results for a set of 26 nodes with such an overlapping view structure to have a unique solution up to a single common scale factor.

Due to the large baselines between images in this dataset, we tracked points manually, rather than relying on a feature-point matching algorithm, in order to demonstrate a simple proof-of-concept. We tracked roughly 250 points across 26 nodes with an average estimated pixel error of 5 pixels (8 milliradians) with the exception of roughly 20 mismatched points. These mismatched points were later detected by their high residual error values $\|V_{c,p} \times \bar{X}_{c,p}\|$ while running the *DALT* algorithm, and subsequently discarded from future iterations of the algorithm. Each node has up to 53 of these points in its visibility set and each pair of linked cameras has at least 10 common points.

Using the available intrinsic calibration and orientation information, we geo-referenced these feature direction vectors into a common coordinate system, then we ran the centralized equivalent of the distributed *DALT* algorithm, immediately giving the asymptotic result.

For this set of nodes, we compared against two sources: the initial GPS readings, and the processed position information using the method discussed in [10]. The reported accuracy of these sources is 2.5 meters and 5cm respectively.

Shown in Figure 2 is the asymptotic result corresponding to the centralized algorithm, and is compared with the GPS data via the cRMSd error metric Δ . The resulting cRMSd distance for this centralized asymptotic estimate to the initial GPS dataset is 2.98m (or 1.87m of error in the 2-D ground plane). All nodes coincided with their GPS measured counterparts to within 7m (or 5m in the 2-D ground plane). For reference, the postprocessed data given in [10] differs from the initial GPS dataset by 4.28m (or 3.63m of error in the 2-D ground plane).

In practice, this routine will not run indefinitely. The actual distributed algorithm converged to within 15 centimeters cRMSd of this asymptotic configuration after 50 iterations (from a random initial configuration in error of roughly 25 meters). After these 50 iterations, the corresponding scale had decreased by approximately 2%.

Apart from the seven degrees of freedom in the Δ error metric used in the rigid-body registration step (which reduces the apparent RMS error slightly), this asymptotic result is obtained independently from the GPS estimates, which are reportedly accurate to within 2.5m. Our conclusion is then that the results are no less accurate than either of the two other sources.

6 Conclusions

In this paper, we have distributed a particular type of least-squares minimization useful in camera network localiza-

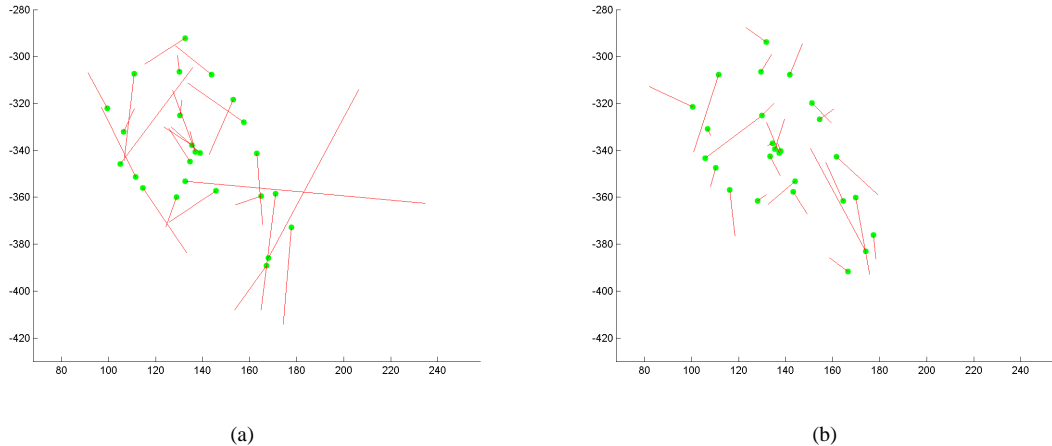


Figure 2: Localization errors (in meters) for omni-directional set. Both the post-processed result from [10], and the result from the *DALT* algorithm are shown in (a) and (b) referenced to the initial GPS data. In both plots, the localization estimates were first registered to the GPS data before comparison. The error lines are drawn in the direction of the GPS reading. Their length represents $10\times$ the discrepancy with GPS.

tion. We have given a proof of convergence and explored graph theoretic conditions under which this point of convergence is unique.

The localization of a camera network enables many exciting applications such as scene representation and geometrical extraction. However, it is not yet clear how well these applications will cope with the challenges of a distributed camera network. The exchange of a few images between nodes will perhaps be acceptable, but there will likely be a significant need for distributed compression, especially on video data. Investigation in this area as well as advancement towards an implementation of an actual wireless ad-hoc camera network will likely prove compelling areas for future research and development.

Acknowledgements

Many thanks to Professors Rich Baraniuk, Mark Embree, and Danny Sorensen for their for their guidance, feedback, and support. Also a special thanks to Ilan Goodman, Ray Wagner, and Santashil PalChaudhuri for their feedback and stimulating discussions.

References

- [1] R. Hartman and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, 2000.
- [2] J. De Leeuw, F. W. Young, and Y. Takane, “Additive structure in qualitative data: An alternating least squares method with optimal scaling features,” *Psychometrika*, vol. 41, no. 4, pp. 471–, 1976.
- [3] J. Bouguet, “Camera calibration toolbox for matlab,” http://www.vision.caltech.edu/bouguetj/calib_doc/.
- [4] N. Goncalves and H. Araujo, “Mirror shape recovery from image curves and intrinsic parameters: Rotationally symmetric and conic mirrors,” in *OMNIVIS*, 2003.
- [5] S. N. Sinha and M. Pollefeys, “Towards calibration a pan-tilt-zoom camera network,” in *OMNIVIS*, 2004.
- [6] Y. Abdel-Aziz and H. Karara, “Direct linear transformation into object space coordinates in close-range photogrammetry,” in *Proc. Symp. on Close-Range Photogram.*, 1971, pp. 1–18.
- [7] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge Univ. Press, 1988.
- [8] G. Golub and C. Van Loan, *Matrix Computations*, John Hopkins University Press, 1996.
- [9] K. S. Arun, T. S. Huang, and S. D. Blostein, “Least-squares fitting of two 3-d point sets,” *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 9, no. 5, pp. 698–700, 1987.
- [10] S. Teller, M. Antone, Z. Bodnar, M. Bosse, S. Coorg, M. Jethwa, and N. Master, “Calibrated, registered images of an extended urban area,” in *CVPR*, 2001.