

Distributed Camera Network Localization

William E. Mantzel, Hyeokho Choi, Richard G. Baraniuk
 Department of Electrical and Computer Engineering
 Rice University, Houston, Texas 77005

ABSTRACT

Localization, estimating the positions and orientations of a set of cameras, is a critical first step in camera-based sensor network applications such as geometric estimation, scene reconstruction, and motion tracking. We propose a new distributed localization algorithm for networks of cameras with sparse overlapping view structure that is energy efficient and copes well with networking dynamics. The distributed nature of the localization computations can result in order-of-magnitude savings in communication energy over centralized approaches.

I. INTRODUCTION

Increasingly powerful microprocessors, along with recent advances in micro-sensor technology, wireless communications, and power efficiency, provide new opportunities to sense, compute, and discover via wireless sensor networks. A *sensor network* consists of a collection of nodes placed or scattered throughout an environment of interest. Each node consists of a power supply, set of sensors, data processor, and a wireless communication system.

In this paper, we focus on networks of digital cameras. As CCD and CMOS image/video cameras become smaller, less expensive, and more power efficient, *camera networks* will become increasingly practical and widespread. Example camera network applications include most of the key tasks from computer vision, including low-dimensional correspondence searches, mosaicking through image rectification, view morphing, image-based navigation, and scene geometry extraction. Due to the high-dimensionality of image and video data, it is paramount in a camera network to distribute as much processing as possible locally within the network.

A critical first step in any camera network deployment is *localization*; the camera nodes must determine their positions and orientations in three-dimensional (3-D) space. Most current sensor network localization techniques use acoustic delays and radio frequency (RF) intensities to create a set of pairwise distances in order to localize [1]. In contrast, little work has been done on distributed algorithms for camera localization via the images acquired at each node.

In this paper, we develop the Distributed Alternating Localization-Triangulation algorithm (DALT), an iterative, distributed algorithm for camera network localization. This

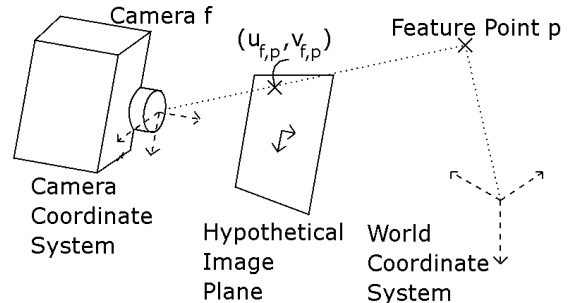


Fig. 1. Coordinate systems and projection of feature point X_p onto the hypothetical focal plane. After pinhole projection, the normalized image plane coordinates are given by $[u_{f,p}, v_{f,p}]$.

iterative piecewise linear localization algorithm performs on par with state-of-the-art centralized localization algorithms where one processor has access to all of the acquired images.

After setting up the localization problem in Section II, we develop the DALT algorithm in Section III and overview a convergence proof. In Section IV, we discuss practical issues such as DALT's robustness and tradeoffs between computation and communication. Section V presents results on actual images. We conclude with a discussion on this algorithm in Section VI.

II. CAMERA NETWORK LOCALIZATION

This section overviews the camera localization problem and explores the conditions required for a unique solution. Concerning notation, scalars are generally written in lower case, while vectors are written in upper case. All matrices appear in boldface. The subscripts x , y , and z refer to the first, second, and third elements of a vector or the first, second, and third rows of a matrix. All other subscripts serve as indices for groups and sequences of vectors and matrices.

A. Camera Projection Model

Consider a *feature point* at the position X in 3-D space; its coordinates in the fixed "world" coordinate system are $X = [X_x, X_y, X_z]^T \in \mathbb{R}^3$. Consider a camera viewing the feature point from position $-RT$ and orientation R in the world coordinate system; we call T the camera's translation vector.¹ The camera has its own local "camera-centric" coordinate

This research was supported by NSF, AFOSR, ONR, and the Texas Instruments Leadership University Program. Email: {willem,choi,richb}@rice.edu. Web: dsp.rice.edu.

¹An orientation matrix is an orthogonal matrix with $\det(R) = +1$. $R_f[i, j]$ is the inner product between the world coordinate system's i th unit axis and the camera coordinate system's j th unit axis.

system in which the coordinates of the feature point become $\bar{X} = \mathbf{R}^T X + T$.

As depicted in Figure 1, the feature point X maps to a 2-D image coordinate $[u, v]^T$ on the camera's hypothetical *focal plane*. Under normalized pinhole (perspective) image formation [2], the 2-D image coordinates of X are $[u, v]^T = [\bar{X}_x, \bar{X}_y]^T / \bar{X}_z$.

These relations lead to the following two fundamental bilinear constraints for any camera/feature point pair:

$$u(R_z^T X + T_z) = (R_x^T X + T_x), \quad (1)$$

$$v(R_z^T X + T_z) = (R_y^T X + T_y) \quad (2)$$

where R_x, R_y, R_z are the first, second and third *columns* of \mathbf{R} , respectively.

The actual image coordinates are related to the normalized pinhole image coordinates by a continuous function $c: \mathbb{R}^2 \rightarrow \mathbb{R}^2$. This function is commonly modelled as either affine or a slightly more involved parameterized function that takes into account terms of radial distortion caused by the lens. Previous work by Tsai, Zhang, Heikkila, Silven, and Bouguet has led to powerful and practical means for computing these parameters [3, 4, 5, 6]. Hence, in practice, c can be effectively inverted with sub-pixel accuracy. Because these *intrinsic parameters* vary on a small scale (less than their measurement precision) over a wide variety of environmental conditions, they can often be estimated off-line before deployment. Hence, for the remainder of the paper, we will assume that we have access to the true normalized image coordinates $[u, v]^T$.

B. Localization Problem

Now suppose that we have several cameras with unknown locations $-\mathbf{R}_f T_f$ and orientations \mathbf{R}_f , $f \in \{1, 2, \dots, F\}$, and suppose that each camera views a subset of a set of labelled *feature points* X_p , $p \in \{1, 2, \dots, P\}$ whose 3-D positions are also unknown. The *camera localization problem* is to estimate the $\{T_f\}$, $\{\mathbf{R}_f\}$. The closely related problem of *scene geometry extraction* is to estimate the $\{X_p\}$. Camera localization is also known as *extrinsic camera calibration* in the computer vision literature and bears a similarity to the *structure from motion* problem.

By moving all the terms from constraints (1), (2) to one side of the equation, we can formulate the localization problem as a bilinear least-squares optimization:

$$\min_{\mathbf{R}_f, T_f, X_p} \sum_{f=1}^F \sum_{p=1}^P \left[([1, 0, -u_{f,p}](\mathbf{R}_f^T X_p + T_f))^2 + ([0, 1, -v_{f,p}](\mathbf{R}_f^T X_p + T_f))^2 \right]. \quad (3)$$

Note that many combinations of camera f and feature point p will not contribute to the summation, since some points are not visible to some cameras. Also, any errors in the 2-D image coordinates $[u_{f,p}, v_{f,p}]^T$ will prevent this optimization from being satisfied trivially.

C. Uniqueness of Localization

It is easy to show that the solution to (3) is not unique. In particular, given any \mathbf{R}_f, T_f , and X_p satisfying the constraints (1), (2) the substitutions

$$\begin{aligned} \mathbf{R}_f &\leftarrow \mathbf{R}_f \mathbf{R}_*^T \\ T_f &\leftarrow \alpha_* (T_f + \mathbf{R}_f T_*) \\ X_p &\leftarrow \alpha_* \mathbf{R}_* (X_p - T_*) \end{aligned}$$

will also satisfy the constraints. Here T_* , \mathbf{R}_* , and α_* are an arbitrary translation, rotation, and scaling, respectively. Thus, using only image information at each camera, any localization will be relative to a local coordinate system (defined by T_* and \mathbf{R}_*) and scaling (defined by α_*).

In the particular case of a pair of cameras, this relative localization is known as *weak calibration*. We wish to extend this concept to so-called *sparse camera networks* where the fields of view of the cameras do not all coincide. To this end, we introduce some concepts to describe camera network topologies that can be localized.

Definition 1: Two cameras f_i and f_j are *linked* (denoted $f_i \leftrightarrow f_j$) if they each view the same set of six or more feature points.

Faugeras et al. have shown that if two calibrated cameras share six or more common feature points, then their relative orientation is determined, and their relative position is determined up to a scalar [7].

Definition 2: A *sparse camera network* is one in which at least one pair of the cameras is not linked.

Definition 3: A *linked triangle* is a set of three cameras such that all three pairs of cameras within the triangle are linked.

Theorem 4: Consider a linked triangle Δ with the properties that (i) the position and orientation of one of the cameras is known (call it f_1) and (ii) the distance from that camera to one of the other cameras (call it f_2) is known. Then the positions and orientations of all three cameras f_1, f_2, f_3 can be uniquely computed.

Proof: The orientations of f_2 and f_3 can be determined through their relative orientations from f_1 . The position of f_2 can be determined because the distance and direction from f_1 are known. The position of f_3 can be determined through triangulation of intersecting epipolar rays from the known positions f_1 and f_2 . \square

Definition 5: Two cameras f_i, f_j are *triple-wise connected* if there exist linked triangles $\Delta_1, \Delta_2, \dots, \Delta_n$ such that triangles Δ_r and Δ_{r+1} have two cameras in common, and $f_i \in \Delta_1$ and $f_j \in \Delta_n$.

Definition 6: A *microcluster* is a set of cameras M with the special property that if $f_i \in M$ then $f_j \in M$ if and only if f_i is triple-wise connected to f_j .

An example of a microcluster is illustrated in Figure 2.

Theorem 7: All cameras within a given microcluster can be uniquely relatively localized in some coordinate system up to one global scale factor so that the minimum seven degrees of freedom are achieved.

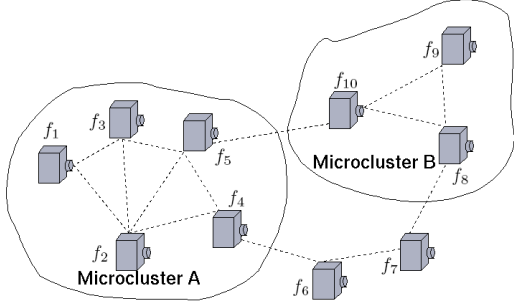


Fig. 2. Microcluster illustration for linked cameras. Cameras within microclusters A and B can potentially determine their relative position and orientation within a single scale factor each. The theory developed here does not apply to cameras f_6 and f_7 . Note that the two microclusters will not necessarily share the same coordinate system.

Proof: In the microcluster M consider a linked triangle $\Delta_1 = \{f_1, f_2, f_3\}$. Define the coordinate system to be a scaled version of camera f_1 's coordinate system such that f_1 and f_2 are unit length apart (the fixed scale factor). Then, by Theorem 4, the positions and orientations of all cameras in Δ_1 are determined. For any other camera f_j there exist linked triangles Δ_1 through Δ_n such that Δ_i and Δ_{i+1} have two cameras in common and $f_j \in \Delta_n$. By Theorem 4, if the positions and orientations of the cameras in Δ_i are known, then so are those in Δ_{i+1} . Since the positions and orientations of the cameras of Δ_1 have been determined, it follows through induction that all cameras in $\Delta_1 \cup \dots \cup \Delta_n$ can be localized. In particular, the position and orientation of f_j are determined. \square

Note that each microcluster will be localized in terms of a potentially different local coordinate system.

III. DISTRIBUTED LOCALIZATION

In this section, we exploit the bilinear constraints (1) and (2) to develop the Distributed Alternating Localization-Triangulation (DALT) algorithm, an iterative technique for estimating the camera network positions and orientations. This algorithm is based on two well-known concepts: triangulation and localization.

A. DALT Algorithm

We use the pseudocode depicted in Algorithm 1 to describe at a high level the algorithm that runs on each node. Through this algorithm, the estimation 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 other nearby camera nodes to triangulate other common points. Now we will give the details behind lines 5, 6, 10, 11, and 12.

Lines 5, 10: Triangulation. Given estimates for the translations and orientations of at least two cameras that view

Algorithm 1 DALT

- 1: **if** localization known **then**
 - 2: *broadcast localization to linked cameras*
 - 3: **else**
 - 4: *wait until receiving localization broadcast of 2 or more linked cameras*
 - 5: *use these neighboring localizations to triangulate 6 or more feature points*
 - 6: *initial self-localization from these points*
 - 7: **end if**
 - 8: **repeat**
 - 9: *broadcast localization estimate of self*
 - 10: *use all available localizations to triangulate as many feature points as possible*
 - 11: *with fixed orientation, optimize translation*
 - 12: *with fixed translation, optimize orientation*
 - 13: **until** localization estimate converges
-

a common feature point, one can estimate that point's 3-D position X by solving a linear least-squares system comprising (1) and (2).

Line 6: Initial Localization. Conversely, if there exist $n \geq 6$ feature points with known 3-D positions, then one can use (1) and (2) to estimate the twelve parameters in \mathbf{R} and T , the orientation and translation of the camera, respectively. This approach is a special case of the Direct Linear Transform (DLT) when the intrinsic calibration parameters are known [8].

This operation turns out to be a null space estimation of the $2n \times 12$ DLT matrix \mathbf{F} that can be accomplished by taking the eigenvector corresponding to the smallest eigenvalue of $\mathbf{F}^T \mathbf{F}$. The sign of the eigenvector is chosen such that the determinant of the orientation is positive. The camera's orientation and translation $\hat{\mathbf{R}}$ and \hat{T} are then determined up to a positive scale factor.

However, the orientation matrix must be orthogonal and hence has only three degrees of freedom. To enforce this condition, consider the singular value decomposition (SVD) of the orientation matrix $U^T \hat{\mathbf{R}} V = \text{diag}(\sigma_1, \sigma_2, \sigma_3)$. Then the closest valid orientation (in Frobenius norm) to the set $\{\alpha \hat{\mathbf{R}} : \alpha > 0\}$ is UV^T with corresponding translation $\frac{1}{\sigma} \hat{T}$ for $\sigma = \frac{1}{3}(\sigma_1 + \sigma_2 + \sigma_3)$.

Line 11: Optimizing Translation. In the case where there already exists some estimate of orientation, the translation of the camera T can be estimated in a manner similar to triangulation while leaving the estimate for orientation fixed. As in triangulation, only two or more points are needed.

Line 12: Optimizing Orientation. To optimize orientation while keeping the translation estimate fixed, we use the Levenberg-Marquart algorithm to optimize each camera's orientation estimate individually [9]. By using constraints (1) and (2) in a least-squares optimization, we can ensure that the new orientation will more closely match those constraints.

In order to work with a well conditioned representation of the orientation matrix \mathbf{R} , we parameterize the orientation in

its exponential matrix form:

$$\mathbf{R} = \begin{bmatrix} r_{1x} & r_{1y} & r_{1z} \\ r_{2x} & r_{2y} & r_{2z} \\ r_{3x} & r_{3y} & r_{3z} \end{bmatrix} = \exp \left(\begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \right)$$

and then optimize the three parameters ω_x , ω_y , and ω_z . 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 will have all the information it needs to perform this optimization locally so communication will not be necessary.

B. Convergence Analysis

In actual deployments, there will be errors in this distributed estimation that could propagate across the network. In this section, we will discuss some of the convergence properties of this iterative algorithm. In particular, we will show that lines 10, 11, and 12 lead to a monotonically decreasing and thus converging cost function as described in (3).

During the re-triangulation in line 10, the estimate for each visible point in the network is re-evaluated so that the cost function is minimized given the new set of camera orientations and translations. Likewise in line 11 and 12, each camera translation and orientation optimizations that occur locally will reduce the squared error terms specific to that camera, because each term in the cost function involves exactly one camera.

The squared error cost function is then a monotonically decreasing and thus converging function. Optimizations made on the orientations of the cameras will not affect the convergence of the algorithm as long as these optimizations are performed separately from the estimations of the positions of points and cameras.

IV. PRACTICAL CONSIDERATIONS

In this section, we first discuss the source and utility of initial information helpful in starting up the DALT algorithm. Then, we discuss how feature points can be matched using motion and the epipolar constraint. Next, we explore the trade-off between computation and communication in the context of a specific part of the algorithm. Finally, we discuss the robustness of the network to link outages or the insertion of new nodes.

A. Initial Information

Initial estimates for the DALT iterations can be provided in several ways. First, in some cases a few ‘‘anchor’’ camera nodes could be able to estimate their positions via global positioning system (GPS) receivers. Second, the absolute orientation of each camera node could be inexpensively estimated with an accelerometer and magnetometer. In the absence of such sensors, the relative orientation between pairs of cameras can be extracted directly from the essential matrices determined from pairwise point correspondences [10].

As we saw above in section III-B, this algorithm will converge regardless of initial information. However, the seven

degrees of freedom discussed in section II-C will more closely resemble their actual values if good initial estimates are available.

B. Feature Point Correspondence

Feature point correspondence between pairs of images poses a significant challenge, especially if the baseline distance between the cameras is long. However, the use of motion can aid this task by extracting one feature point per camera per frame. By communicating with other cameras, a set of feature points from one camera may be either determined to be a match or a miss with a time-matched set of points from another camera.

After extraction and time-matching, a pair of sets is tested for correspondence using the epipolar constraint. This constraint, discussed by Faugeras et al. [11], states that for any normalized image coordinates $M_p = [u_{i,p}, v_{i,p}, 1]^T$ of camera i , and $M'_p = [u_{j,p}, v_{j,p}, 1]^T$ of camera j corresponding to the same point p observed in cameras i and j satisfy:

$$M_p^T \mathbf{E} M'_p = 0 \quad (4)$$

where \mathbf{E} is called the *essential matrix* and is specific to cameras i and j .

This essential matrix can be computed by the eight point algorithm [10] or any of a number of other methods [11]. After computing the best-fit essential matrix for this pair of set of points, $\hat{\mathbf{E}}$, the pair of point sets is determined to be a match if the following quantity is below a certain threshold:

$$\sum_p (M_p^T \hat{\mathbf{E}} M'_p)^2. \quad (5)$$

However, the threshold should depend on the variation of the points because small or trivial motions (i.e., linear motion) would trivially satisfy this constraint.

C. Communication/Computation Tradeoff

A fundamental theme of the sensor network paradigm is the importance of distributed local computation and in-network processing. Clearly, these power savings are not present when the network is small enough or well connected enough that it is just as easy to communicate to a base station node as neighboring camera nodes. Rather, the power efficiency of distributed computation begins to present itself as the network grows and becomes more complex.

Consider the localization of a 300-node network of where each node requires an average of 10 hops to reach the base station. Suppose the average number of feature points observed per camera node is 100. Sending these 2-D feature points to the base station would require 2000 scalars to be transmitted on average. If we were to instead use the DALT algorithm, this number would be reduced to $200 + 6n$ where n is the number of iterations of the algorithm. As the average number of hops to base station grows, this discrepancy will only widen. It is then clear that for sufficiently large and complex networks, we must use a distributed approach.

We wish to perform as many of the computations locally as possible in order to aggressively minimize the communication requirements. For example, line 10 of the DALT algorithm involves an triangulation operation that could be computed by a single camera node that would then broadcast the result to other nodes. Instead, each camera node is required to compute this result so that this broadcast is unnecessary.

To get a concrete idea of the energy savings this redundant computation provides over the otherwise necessary communication in this particular scenario, consider the *moteiv telos* platform [12]. This sensor network architecture utilizes the Texas Instruments MSP430 processor drawing 3mW of power and running at 8MHz. Its Chipcon communication module draws 50mW of power and transfers data at a rate of 250kbps.

The least squares computation necessary to solve this triangulation will take $\frac{2}{3}n^3 + n^2m + nm + 2n^2$ multiply-accumulates for m constraints and n unknowns. In this case, there are a pair of constraints for every camera contributing to this point's triangulation, and three unknowns for the point's position. Supposing that 5 cameras viewed this point, this operation requires under 200 multiply-accumulates for this point. On this 8MHz processor this entire computation would terminate in less than 20 μ s, and consume about 60 nJ of energy per point per iteration on each processor or 300 nJ total. In contrast, sending each 3-D point as three 16 bit scalars would require 250 μ s of the communication system leading to more than 10 μ J per point per iteration. As a conservative estimate, we assume that the camera node only needs to broadcast this vector once, instead of multiple times to each neighboring camera. This rough computation gives an idea of the order of magnitude difference between computation and communication. Note that this factor does not depend on the number of feature points because both energies would scale up linearly. In addition to saving energy, this particular method also simplifies the system because the designation of which motes will perform specific triangulations is no longer necessary.

D. Network Robustness

In addition to the energy saving benefits of the distributed algorithm, it is also advantageous for a camera network to distribute the localization routine so that network outages and node insertions and deletions do not have a catastrophic effect on the algorithm.

If a node is inserted, then not only can it be localized immediately, but also its information can immediately contribute to the improved localization of other cameras in the network rather than requiring coordination from some root node. In fact, the DALT algorithm does not require a comprehensive routing scheme at all because all communication is done between nearby cameras.

In addition to node insertions, the algorithm is capable of handling node deletions. Sometimes in a sensor network, nodes will become unresponsive due to a dead battery, electronic malfunction, radio wave interference, or physical theft. In the case that a set of nodes fail or become unresponsive, a



Fig. 3. One of twenty views of a checkerboard whose corners were treated as feature points to evaluate the DALT algorithm.

given microcluster may break into several smaller microclusters. In this case, each microcluster becomes an autonomous unit capable of localizing itself without intervention from the rest of the network. If a set of nodes become available that cause several microclusters to amalgamate into a single microcluster, the iterative DALT algorithm will bring their localization estimates into a common coordinate system.

V. EXPERIMENTAL RESULTS

To evaluate the DALT algorithm, we used a dataset from the MATLAB camera calibration toolbox containing 20 actual views of a checkerboard pattern with 156 corners, that were tracked to within a pixel and used as feature points [4]. The first of 20 views is shown in Figure 3.

To perform this experiment, we first solved all intrinsic parameters of the camera using the MATLAB toolbox [4]. We then imposed a visibility constraint such that each camera only viewed 30 randomly chosen corners of the 156. This imposed a sparse but random overlapping view structure among the cameras such that each camera was linked to at least 8 and at most 16 other cameras (a median of 11). Then, 4 cameras were initialized whose initial error in orientation corresponds to the orientation error from an essential matrix computation under a non-coplanar point cloud under similar pixel error.² Then, the first set of points are triangulated, followed by the next set of cameras that are localized, iterating across the network. Once all cameras have been localized, the algorithm continues to refine the localization estimate until the algorithm had sufficiently converged.

In order to evaluate this algorithm, we are unable to make a direct least squares comparison between the computed localization estimates and the actual localization because the localization estimates are referenced to some local coordinate system and scale factor. Instead, knowing that our point cloud lies on a planar surface (checkerboard), we use the *planarity metric*, which is invariant to unknown scale, translation, and rotation. This metric is defined as the ratio of the smallest to largest eigenvalue of the covariance matrix of a set of points

²Planar point clouds cause a singularity in the essential matrix computation

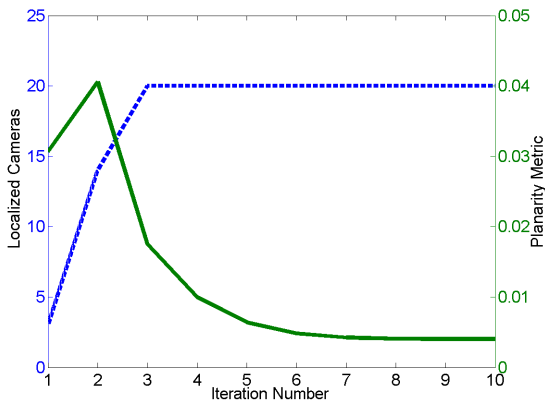


Fig. 4. Results for the DALT localization algorithm using twenty images from the Bouguet dataset[4]. The number of cameras localized and error metric are shown by the dashed and solid lines respectively. The error metric quickly dies to its local minimum, but does not converge to zero due to error in the 2-D feature points. The transient increase of the planarity metric at the second iteration coincides with the introduction of newly triangulated points which contribute to the overall error. All of the cameras have an initial localization estimate by the third iteration of the algorithm.

in 3-D and can be thought of as a measure of flatness. This unitless metric ranges from zero (the points lie on a 2-D plane) to one (the point distribution is radially symmetric). For example, the planarity metric of a set of points uniformly distributed inside a rectangular prism with dimensions $10\text{cm} \times 10\text{cm} \times 2\text{cm}$ would be about $2/\sqrt{10^2 + 10^2} \approx 0.1414$.

Shown in Figure 4, the DALT localization algorithm quickly propagates across the network, and then the error metric settles down at a reasonable value. Note that convergence to zero is prevented by errors in the feature points (in addition to possible convergence to a local minimum).

This error is a quantity relative to the diameter or baseline of the environment. For instance, a planarity error of .25% of a baseline of 3m would be equivalent to about 8mm orthogonal deviation off the plane. Taking into account variation in the other two dimensions would yield a total error of roughly $8\text{mm} \sqrt{3} \approx 14\text{mm}$. Recall that these errors are specific to our chosen scale of 3m and would scale up as the scale of the environment scaled up. For instance, if the scale of the environment were 300m instead of 3m, the error would be on the order of 140cm instead of 14 mm, and so on.

VI. CONCLUSIONS

In this paper, we have developed and evaluated a new method for distributed camera network localization that iteratively refines the previous localization estimate. In addition to analyzing some of the theoretical aspects, we have also explored some of the practical challenges of a sensor network such as energy minimization, and motivated the need for distributed algorithms to meet this challenge.

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 sensor 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.

REFERENCES

- [1] N. Bulusu, D. Estrin, J. Heidemann "GPS-less Low Cost Outdoor Localization For Very Small Devices," *IEEE Pers. Commun.*, 7(5): pp. 28-34, Oct. 2000
- [2] O. Faugeras and Q-T Luong "The Geometry of Multiple Images," *MIT Press*, 2001.
- [3] R. Y. Tsai. "A Versatile Camera Calibration Technique for High-Accuracy 3-D Machine Vision Metrology Using Off-The-Shelf TV Cameras and Lenses," *IEEE Journal Robot. Automat.*, 3(4): 323-344, Aug. 1987
- [4] J. Bouguet, Intrinsic Camera Calibration: a Software Toolbox for Matlab http://www.vision.caltech.edu/bouguetj/calib_doc/
- [5] J. Heikkila and O. Silven , "A Four-step Camera Calibration Procedure with Implicit Image Correction," *Proc. Comp. Vision Pattern Recogn.*, San Juan, Puerto Rico, pp. 1106-12, 1997
- [6] Z. Zhang, "A flexible new technique for camera calibration." *IEEE Trans. Pattern Anal. Machine Intell.*, 22(11):1330-1334, 2000.
- [7] Q.T. Luong and O.D. Faugeras , "Camera Calibration, Scene Motion and Structure recovery from point correspondences and fundamental matrices," *Int. Journal of Computer Vision* 22(3): 261-289, 1997
- [8] Y. Abdel-Aziz and H. Karara "Direct linear transformation into object space coordinates in close-range photogrammetry.," *Proc. Symposium on Close-Range Photogrammetry*, Urbana, Illinois, pp. 1-18, 1971
- [9] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling, "Numerical Recipes in C: The Art of Scientific Computing," Cambridge Univ. Press; 1988
- [10] H. Longuet-Higgins "A computer algorithm for reconstructing a scene from two projections," *Nature*. 293: 193-195, 1981
- [11] O. Faugeras and S. Maybank "Motion from point matches: multiplicity of solutions.," *Int. Journal of Computer Vision*, 4(3):225-246.
- [12] J. Polastre, G. Tolle, and J. Hui "Low Power Mesh Networking with Telos and IEEE 802.15.4," *ACM Conf. on Embedd. Netw. Sensor Systems (SenSys)*, November 2004