Ad-Hoc Sensor Localization in Active Camera Networks

Ryan Farrell
Computer Science Department
University of Maryland, College Park
farrell@cs.umd.edu

Abstract

We present a new approach for self-localization in multi-modal camera networks. Unless measurements and events observed by a sensor can be localized in time and space, the observations are relatively meaningless. We here present what we believe to be the first work utilizing cameras for localization without prior knowledge of their locations. Active Cameras, able to rotate and zoom, are used to locate each sensor in pan-tilt space. With nothing more than these pan-tilt coordinates, cameras and sensors are simultaneously localized within a normalized frame of reference by numerically solving a system of nonlinear geometric constraints. We describe experimental results using two PTZ Cameras and a small wireless sensor network. These results are extended, evaluating the algorithm's scalability with a large simulated network and an empirically-validated noise model. The paper concludes with a discussion of future work.

1. Introduction

With applications ranging from monitoring remote inhospitable environments (see Figure 1) to implanting intelligent and interactive sensors in our homes, wireless sensor networks have emerged as one of the most promising sensing technologies. In the multi-disciplinary endeavor to realize their potential, one of the most active areas of sensor network research has been localization. A sensor's observations acquire meaning when associated with the time and location at which they were recorded. For example, a temperature reading of 62.7° is useless if decoupled from the details of when and where the reading was made.

In our work, we have chosen to focus on the problem of spatial localization. More specifically, we propose a solution for placing the constituent nodes in the sensor network in a common spatial frame of reference.

Figure 1. Cypress-Tupelo Swamp. A potential site for wireless sensor network deployment.

1.1. Our Approach

Developed within a larger framework [7], our technique for spatial localization concentrates on the widespread need for ad-hoc deployment. We assume no prior information whatsoever about the locations of the sensors or cameras[1]. We also require no specialized hardware, using commodity surveillance cameras, wireless motes and LEDs. An outline of our approach is as follows:

- Active pan-tilt-zoom cameras are used to find the angular locations of the sensor nodes in the camera's local pan-tilt parameterization of space. Diffuse LEDs are used to make the sensor nodes visible to the cameras.
- Given a subset of these pan-tilt coordinates, a system of nonlinear geometric constraints is first generated, then solved numerically to provide an estimate of the unknown locations within a common spatial frame.
- Many such random subsets are used together to improve signal-to-noise, producing both locations for the sensors in the network as well as the location and orientation of the cameras.

[1] For simplicity, we temporarily assume that the sensor nodes lie in a some common plane but subsequently assert that this constraint is unnecessary.

1.2. Related Work

As mentioned previously, sensors in a wireless sensor network must be able to describe the events they observe in both time and space. Several techniques have been presented in the literature for time-synchronization [4, 9, 16, 14]. For spatial localization, there have been several innovative approaches suggested. Various technologies such as ultrasound ranging, acoustic time of flight, radio signal strength and interferometry, have been demonstrated to acquire good pairwise distance estimates, often incorporating the acquired range measurements into a planar graph (see [18] and surveys [12, 19]). Poduri *et al.* [20] showed that when nodes are placed in 3D, the number of neighbors that an average node has grows dramatically, an observation that suggests investigation into methods which do not rely on pairwise distances.

Recent research in camera localization includes work by Funiak, *et al.* [8] who proposed a robust probabilistic model for simultaneous localization and tracking. Lymberopoulos, *et al.* [15] considered the use of epipolar geometry to perform localization, using LEDs to assist in imaging other sensors.

Probably the most closely related work to ours is the Spotlight system, designed by Stoleru *et al.* [23]. The primary difference between our approach and this work lies in our focus on truly ad-hoc deployment. We assume that the cameras will have unknown location and orientation whereas the Spotlight system relies on accurate knowledge of both camera position and orientation. Furthermore, in Spotlight the individual sensors detect when a laser or global light source is projected onto them, a design that requires time-synchronization.

2. Optical Self-Localization

In this section, we first describe the underlying theoretical basis of our self-localization approach. We next discuss potential numerical techniques for solving the system of multivariate nonlinear equations resulting from our analysis. While at present, we have restricted ourselves to networks located in the plane, we outline the modifications needed to allow node placement in three dimensions. The importance of intrinsic camera calibration is also addressed.

2.1. Deriving the Nonlinear Formulation

Let us consider a planar network of wireless sensor nodes, $\mathbf{N} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$, where $\mathbf{p}_i = (px_i, py_i, 0)$ denotes the location of node i. For the cameras, we must describe both position and orientation. With each camera j, we therefore associate a position $\mathbf{c}_j = (cx_j, cy_j, cz_j)$ and a 3-DOF rotation $\mathbf{\Theta}_j = (\theta x_j, \theta y_j, \theta z_j)$. As we focus on ad-hoc deployment, the values for $\mathbf{p}_i$, $\mathbf{c}_j$, and $\mathbf{\Theta}_j$ are all unknown.

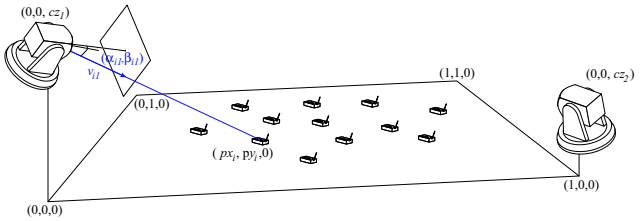


Figure 2. Normalized Coordinate Frame. Illustrates the sensor network $\mathbf{N} = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$, $\mathbf{p}_i = (px_i, py_i, 0)$, with two active cameras, respectively located at $\mathbf{c}_1$ and $\mathbf{c}_2$. Geodetic locations are normalized such that the cameras are separated by unit distance in the plane, thus $\mathbf{c}_1 = (0, 0, cz_1)$ and $\mathbf{c}_2 = (1, 0, cz_2)$. The illustration also depicts node $\mathbf{p}_i$ localized at pan-tilt coordinates $(\alpha_{i1}, \beta_{i1})$ relative to the first camera.

Without any notion of geodetic location or scale, the best that we can hope to recover is a local and relative coordinate frame. Rather than pick some arbitrary scale, we have chosen to normalize the coordinate frame based on the distance between the two cameras. The camera locations are thus $\mathbf{c}_1 = (0, 0, cz_1)$ and $\mathbf{c}_2 = (1, 0, cz_2)$. The camera elevations remain unknown as do the orientation parameters. Figure 2 depicts the normalized coordinate frame (note that the nodes are not constrained to lie within the unit square).

The only values we are able to measure are the angular location of a given node relative to each camera. We parameterize the camera's pan-tilt space with coordinates (α, β), in radians. Camera j observes node i at location $(\alpha_{ij}, \beta_{ij})$, indicating that camera j would need to pan by α_{ij} and tilt by β_{ij} in order to aim directly at node i.

We now derive a relationship between these measurements and the unknown values we wish to determine. In considering the relationship between camera j and node i, for convenience we define the offset vector $\mathbf{v}_{ij} = \mathbf{p}_i - \mathbf{c}_j$. As a vector can also be decomposed into the product of its magnitude with a unit vector in the same direction, we can write $\mathbf{v}_{ij} = ||\mathbf{v}_{ij}|| \cdot \hat{\mathbf{v}}_{ij}$. The unit vector $\hat{\mathbf{v}}_{ij}$ can be expressed in two ways. The first expression is obtained by simply dividing $\mathbf{v}_{ij}$ by its magnitude, $||\mathbf{v}_{ij}||$, yielding

$$\hat{\mathbf{v}}_{ij} = \frac{\mathbf{v}_{ij}}{||\mathbf{v}_{ij}||} = \frac{\mathbf{p}_i - \mathbf{c}_j}{||\mathbf{v}_{ij}||} = \frac{1}{||\mathbf{v}_{ij}||} \begin{pmatrix} px_i - cx_j \\ py_i - 0 \\ 0 - cz_j \end{pmatrix}. \quad (1)$$

The second expression is found by rotating the optical axis of camera j such that it passes through $\mathbf{p}_i$. The standard camera model used in Computer Graphics (e.g. [11], p. 434) defines the local coordinate frame

with the camera looking downward along the z-axis at an image plane parallel to the x-y plane. The optical axis thus points in the $-\hat{z}$ direction in the local coordinate frame. To transform from the camera's local frame into world coordinates, we must apply the three-axis rotation defined by $\mathbf{\Theta}_j$. To now point the optical axis at $\mathbf{p}_i$, we pan and tilt the camera with the rotations corresponding to $(\alpha_{ij}, \beta_{ij})$. At length, we find the second expression

$$\hat{\mathbf{v}}_{ij} = \underbrace{R_{\alpha_{ij}} R_{\beta_{ij}}}_{pan-tilt} \cdot \underbrace{R_{\mathbf{\Theta}_j}}_{world} \cdot \underbrace{(-\hat{z})}_{local}. \tag{2}$$

Now, we equate these two expressions for $\hat{\mathbf{v}}_{ij}$. Merging (1), in terms of $\mathbf{p}_i$ and $\mathbf{c}_j$, with (2), in terms of α_{ij}, β_{ij} and $\mathbf{\Theta}_j$, we derive a relationship between the observations and the unknowns we wish to solve for:

$$\frac{1}{||\mathbf{v}_{ij}||} \begin{pmatrix} px_i - cx_j \\ py_i - 0 \\ 0 - cz_j \end{pmatrix} = R_{\alpha_{ij}} R_{\beta_{ij}} \cdot R_{\mathbf{\Theta}_j} \cdot \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}. \tag{3}$$

where $||\mathbf{v}_{ij}|| = \sqrt{(px_i - cx_j)^2 + py_i^2 + cz_j^2}$.

This relationship embodies three constraints per node-camera pair, one each in x, y and z. As only two measurements are used, however, the three constraints cannot be independent. So, each node-camera pair contributes only two to the rank of the system for a total rank of $4n$. We have $2n + 8$ variables or unknowns to solve for, elevation and 3-DOF orientation (4 parameters) for each camera, and the 2D position of each of the n nodes. For the system to be sufficiently constrained, we must have $4n \geq 2n + 8$ which yields $n \geq 4$. Thus, measurements from at least 4 nodes must be used or the resulting system of nonlinear equations (constraints derived as the components of (3)) will be under-determined. However, as we will see in Section 3, using larger numbers of nodes is preferable, as it helps mitigate the presence of noise in the measurements, greatly improving localization accuracy.

2.2. Solving the Nonlinear System

In the previous section, we derived a system of geometric constraints that govern the relationship between the relative configuration of nodes and cameras and the angular locations of the nodes measured by each camera. As we have noisy measurements, finding a solution to the problem becomes an optimization problem. Optimization is a broad field, and several approaches have been suggested for solving general nonlinear systems.

We chose to base our solver on a variant of Newton's method, one of the most straightforward and well-known techniques for finding roots or function extrema. For a one-dimensional function, Newton's method relies upon the function's derivative to iteratively move toward the nearest root. Newton's method can be extended to multivariate nonlinear systems. Called the Newton-Raphson method, this method logically uses the Jacobian matrix in place of the derivative. While in general the Jacobian must be estimated numerically by finite differences, in our case we know the analytical representation. As with other techniques, Newton's method is prone to convergence at local extrema depending on the initial search location. Our implementation is therefore been based on a globally convergent Newton's approach from Numerical Recipes [21] which attempts to backtrack if it reaches a non-global optimum.

2.3. Extending to 3D

Thus far, we have maintained the assumption that all nodes in the network lie in some common plane, in our normalized coordinate frame, $\forall i, pz_i = 0$. While this may suffice or even be a good approximation for most scenarios, in general, we would like to remove this restriction. Doing so requires minimal change to our model, simply making pz_i an unknown which modifies (3) to produce

$$\frac{1}{||\mathbf{v}_{ij}||} \begin{pmatrix} px_i - cx_j \\ py_i - 0 \\ pz_i - cz_j \end{pmatrix} = R_{\alpha_{ij}} R_{\beta_{ij}} \cdot R_{\mathbf{\Theta}_j} \cdot \begin{pmatrix} 0 \\ 0 \\ -1 \end{pmatrix}. \tag{4}$$

where $||\mathbf{v}_{ij}|| = \sqrt{(px_i - cx_j)^2 + py_i^2 + (pz_i - cz_j)^2}$.

As before, we would still have $4n$ measurements, only now we wish to solve for $3n+8$ variables. This implies that we must have measurements for $n \geq 8$ nodes before the system will be determined.

2.4. Camera Calibration

In order to accurately determine the pan-tilt coordinates of an image point, it is necessary to know the intrinsic or internal characteristics of the camera which include focal length, principal point (the true center of the image) and distortion parameters. Both of cameras used in our experiments were individually calibrated at their widest field-of-view. Many frames of a checkerboard calibration target at different positions and orientations were used as input to the Camera Calibration Toolbox [1] for MATLAB. The points of intersection between neighboring squares on the calibration target are located in each frame and these points are collectively used to estimate the internal parameters for the camera. These parameters are stored in a configuration file and read into the system at run-time.

As the PTZ cameras can also zoom, we first calibrated one camera at various focal lengths. It became obvious that some parameters such as the principal point were not constant across focal lengths, so re-calibration was indeed critical. The most interesting discovery in the calibration process, however, was that even with per-zoom-level calibration, the computation of pan-tile coordinates was error-prone for all but the widest field-of-view zoom. We eventually concluded that the center of rotation coincides with the focal point only for the widest field-of-view. Other zoom levels therefore introduce a translation as the camera pans or tilts about the center of rotation.

3. Experimental Results

Our primary objectives in evaluating our proposed self-localization technique are to quantify its accuracy and confirm its scalability. We first employed a network of 12 MicaZ motes to examine the localization accuracy of our approach under realistic conditions. To overcome the limitations of this small network, we also experimented with a simulated network of 100 motes and found that larger networks offer improved accuracy despite their modest increase in computation overhead.

3.1. Experimental Deployment

Our experimental platform consists of a network of 12 MicaZ motes equipped with omni-directional LEDs and two PTZ cameras: a Sony EVI-D100 and a Sony SNC-RZ25N (see Figure 3). Our application software, written primarily in Java, runs on a standard laptop, using JNI to integrate native code for image processing and for controlling the EVI-D100. Communication with the sensor network is facilitated by another mote wired directly to the laptop. A core component of our system, the nonlinear solver consists of a Java implementation of the globally convergent Newton's method found in Numerical Recipes [21]. Our solver is substantially $(20-25\times)$ faster than MATLAB's Optimization toolbox which we used in initial simulations.

Figure 3. Experimental platform: MicaZ sensor network, laptop, 2 PTZ cameras.

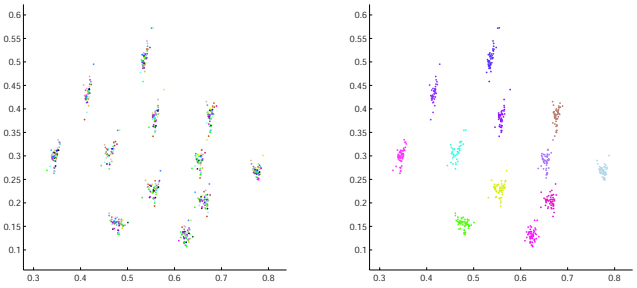
To self-localize a network $\mathbf{N}$ consisting of n nodes, the system follows this procedure:

- A subset of k nodes, namely $S_k = \{\mathbf{p}_{i_1}, \ldots, \mathbf{p}_{i_k}\}$ is selected.
- The nodes in S_k are sequentially interrogated, each ascending an omni-directional LED which the cameras use to pinpoint the node's location in their individual pan-tilt parameterization.
- After finding the pan-tilt location for each node $\mathbf{p}$, the corresponding nonlinear system of equations is solved numerically, yielding location estimates for $\forall \mathbf{p} \in S_k$.
- Many such subsets, $S_k^{(1)}, S_k^{(2)}, \ldots, S_k^{(T)}$, are randomly chosen and location estimates for each subset $S_k^{(t)}$ is determined as above.
- When this process completes, we have several location estimates for each node ($\frac{Tk}{n}$ expected) and average them to derive the final location estimate for each given node.

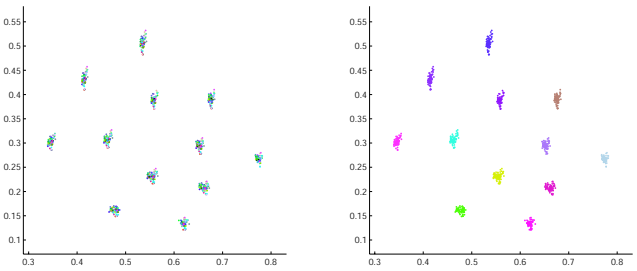
Sample results are presented in Figure 4 for our network of 12 MicaZ motes. Each dot in these graphs shows an estimated mote location over the unit square $[0,1]^2$, as calculated by the system. In Figure 4(a), the localization results for subsets of size $k = 6$ are shown, on the left colored by subset ID and on the right colored by mote ID. Figure 4(b) then shows analogous results for subsets of size $k = 9$, hinting at the improvement in accuracy by using larger subsets. While the absence of ground truth localization prevents us from determining precise localization errors, we can see how the solutions for random subsets collectively suggest an accurate location estimate.

3.2. Empirically-Verified Noise Model

Though unable to quantify the localization accuracy of our experiments due to the lack of ground truth locations (and camera orientations), we nonetheless hoped that we could characterize the observed noise in such a manner that we might produce similar measurements synthetically. Instead of generating the exact pan-tilt measurements that would result from the ground truth locations, we apply an angular perturbation $\sim N(0, \zeta)$ to each measurement, akin to the model proposed by Ercan *et al.* [5, 6], using $\zeta = 0.1^\circ$. We generate synthetic measurements using the mean locations derived from the experimental network using subsets of size $k = 6$. A comparison between the real and synthetic data is presented in Figure 5. The synthetic noise model closely reflects the error characteristics of the observed data.

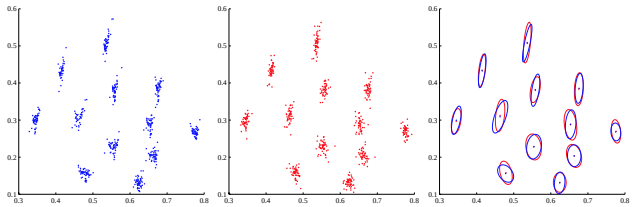


(a) Results with subsets of size $k = 6$ nodes



(b) Results with subsets of size $k = 9$ nodes

Figure 4. Self-Localization Results for our Experimental 12-Mote (MicaZ) Network. (a) shows simultaneous plots of localization from 100 randomly selected 6-mote subsets, on the left colored by subset ID, on the right by mote ID. (b) shows analogous results for 9-mote subsets.



(a) Observed (b) Synthetic (c) Compared

Figure 5. Empirically-validated Noise Model. (a) plots the solutions for 100 subsets using actual measurements obtained from the experimental system (same data displayed in Figure 4(a)). (b) plots the solutions for 100 subsets using simulated measurements obtained using the synthetic noise model. (c) shows a comparison between the the observed and the synthetic data. The mean location for each mote along with the error ellipse indicating 95%-confidence are depicted. As in the first two plots, blue is used for the observed data, red for the synthetic data.

3.3. Simulation

To demonstrate the improved localization accuracy obtained with larger subsets, we turned from our small physical deployment to a larger simulated network. Simulated networks provide us with ground truth mote locations, essential to properly addressing the question of localization error.

The localization accuracy is associated, not with the size of the network, but with the subset size k. So, whether the network has tens, hundreds or thousands of nodes, the localization accuracy will be same for a given subset size k assuming the number of estimates per node is fixed. For a fixed k, the required computation time grows linearly with the size of the network, as the network size n grows, the number of random subsets processed T must increase proportionally.

Our simulated network distributed 100 nodes uniformly at random within a unit square $[0, 1]^2$. The cameras were placed at the same normalized heights as in our experimental setup, the first above the origin $(0, 0)$, the second above $(0, 1)$. The same process of selecting subsets described in Section 3.1, angularly locating the motes and solving the nonlinear system is used to find location estimates within the plane. The noise model used is the same described above in Section 3.2.

In Figure 7 we present results for a 100-node network choosing subsets of size 5, 10, 20 and 50. To articulate these results in real-world terms, if we imagine the network is 100m $\times$ 100m, then these errors would be as depicted in Figure 6 and Table 1. Remarkably, with two cameras positioned 100 meters apart, the system can localize every node to within 15.82 cm of its true location. Moreover, it localizes a majority of them with errors less than 5 cm.

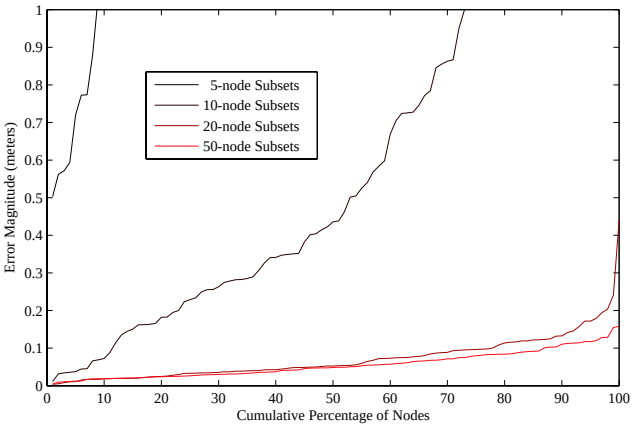


Figure 6. Localization Error Distributions. For a simulated 100 $\times$ 100 meter network containing 100 nodes, the curve showing the percentage of nodes localized within any given error is shown for 5-, 10-, 20- and 50-node subsets.

Subset Size	50% Confidence	95% Confidence
5 nodes	6.70 m	96.25 m
10 nodes	43.57 cm	2.14 m
20 nodes	5.26 cm	17.19 cm
50 nodes	4.85 cm	11.73 cm

Table 1. Errors for $100m \times 100m$ 100-node Network.

4. Future Work

While this work has provided a useful method for network self-localization, there are a number of directions yet to be explored. Much of the future work suggested below is of broader interest, particularly to the Computer Vision community.

4.1. Epipolar Geometry in Pan-Tilt Space

Epipolar Geometry describes the relationship of image locations between two cameras. For standard fixed field-of-view cameras, the epipolar constraint dictates that a spatial point (X, Y, Z) imaged at point (x, y) in the first camera corresponds to a point (x', y') somewhere along the unique epipolar line in the second camera's image. This constraint arises from the simple geometric fact that the point (X, Y, Z) and the two camera centers determine a unique plane, not surprisingly called the epipolar plane. A detailed discussion of epipolar geometry can be found in Chapter 9 of [10].

Epipolar geometry is fundamental to areas of Computer Vision such as Structure from Motion and 3D Scene Reconstruction. We consider it here as it has the potential to solve directly for the camera locations instead of searching over a high-dimensional parameter space. Epipolar geometry has been derived for fixed field-of-view cameras [10] and omni-directional cameras [17] (hemispherical or panoramic images captured using fish-eye or catadioptric lenses). To our knowledge however, epipolar geometry has not been derived for the pan-tilt space parameterization we have used in our work.

4.2. Multiple Cameras and Occlusions

Throughout the work presented in this paper, we have used two cameras and maintained the assumption that the sensors to be localized were visible from both cameras. In practice however, this assumption will rarely hold. We therefore would like to explore an estimation model where there are a large number of cameras and each sensor must be imaged by at least two, but may be occluded from the other cameras. While we have not yet looked closely at such a model, we are optimistic that this may be possible. We also wish to consider integrating a better model for combining multiple measurements, perhaps similar to that used by Stroupe *et al.* [24].

4.3. Distributed Network Calibration

Closely related to the simultaneous localization of a large network of cameras with potentially occluded sensors is the problem of Distributed Network Calibration. Large-scale Camera Network Calibration has been investigate previously [2, 22] and distributed approaches for fixed-field of view cameras have recently been published [3, 8]. We hope to investigate a distributed implementation, perhaps based on the TinyOS implementation of the Levenberg-Marquardt algorithm (see below).

4.4. Levenberg-Marquardt implementation

In closely related work, we modified an existing implementation [13] of the Levenberg-Marquardt algorithm, porting it to run in TinyOS on MicaZ mote [7]. The L-M algorithm is frequently used to numerically solve nonlinear equations in high-dimensions in fields such as Computer Vision [25]. In theory, using the L-M algorithm instead of Newton's method should both improve the running time and facilitate self-localization of larger networks.

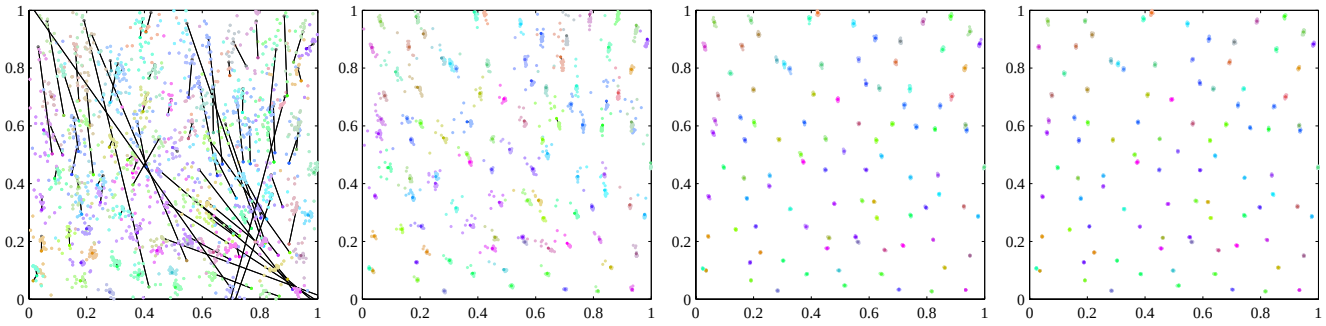
5. Summary

In this paper, we have presented an approach for the optical self-localization of a network of wireless sensor nodes using active cameras. Unlike most localization schemes, this method does not rely on inter-node distance measurements, but rather on the location of a node in each camera's pan-tilt space. Our algorithm numerically solves the systems of nonlinear equations derived from the constraints produced by the analytical model and a subset of the measurements. These solutions are used to estimate the locations of the nodes in the network. We demonstrated results both on a physical deployment and a larger-scale simulation and concluded with a discussion of future work.

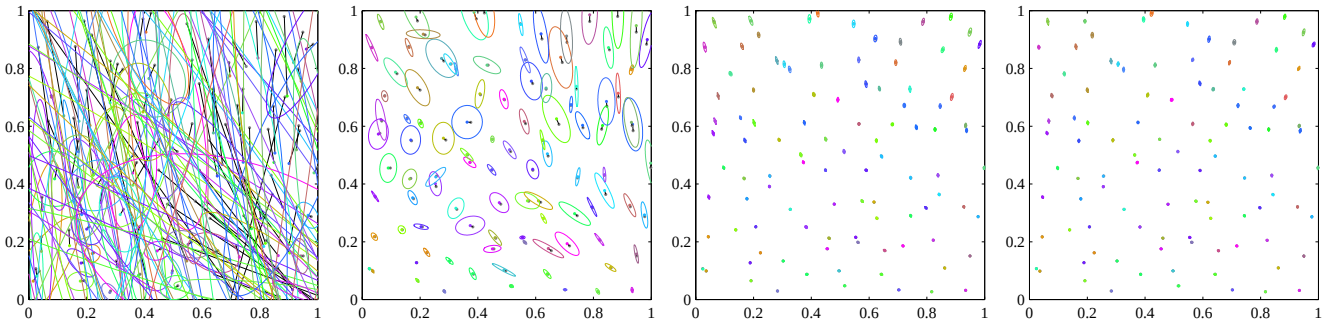
6. Acknowledgments

I want to express my appreciation to those at John's Hopkins University and the JHU Applied Physics Laboratory who have guided and supported this work: Dennis Lucarelli, Cash Costello, Amit Banerjee, I-Jeng Wang and Andreas Terzis. I also wish to thank my advisor, Larry Davis, for his feedback and direction.

I wish to conclude by thanking my family for their unfailing support and unmatched encouragement.



(a) Solutions, $k = 5$ (b) Solutions, $k = 10$ (c) Solutions, $k = 20$ (d) Solutions, $k = 50$



(e) 95% Errors, $k = 5$ (f) 95% Errors, $k = 10$ (g) 95% Errors, $k = 20$ (h) 95% Errors, $k = 50$

Figure 7. Simulated Large-Scale Network. (a)-(d) present localization results for 5-, 10-, 20- and 50-node subsets, respectively. (e)-(h) present the mean of the solutions for each node together with the ellipse of 95% confidence. Each node is plotted in a unique color and a black error line is drawn between each node's ground truth location and the mean of its solutions.

References

[1] J.-Y. Bouget. Camera calibration toolbox for matlab. Available at `http://www.vision.caltech.edu/bouguetj/calib_doc/`.

[2] M. Brand, M. E. Antone, and S. J. Teller. Spectral solution of large-scale extrinsic camera calibration as a graph embedding problem. In *ECCV*, pages 262–273, 2004.

[3] D. Devarajan, R. J. Radke, and H. Chung. Distributed metric calibration of ad hoc camera networks. *ACM Trans. Sen. Netw.*, 2(3):380–403, 2006.

[4] J. E. Elson, L. Girod, and D. Estrin. Fine-grained network time synchronization using reference broadcasts. In *OSDI*, pages 147–163, Dec. 2002.

[5] A. Ercan, A. E. Gamal, and L. Guibas. Camera network node selection for target localization in the presence of occlusions. In *SenSys Workshop on Distributed Cameras*, page to appear, November 2006.

[6] A. Ercan, D. Yang, A. E. Gamal, and L. Guibas. Optimal placement and selection of camera network nodes for target localization. In *DCOSS*, pages 389–404, 2006.

[7] R. Farrell, R. Garcia, D. Lucarelli, A. Terzis, and I.-J. Wang. Localization in multi-modal sensor networks. Submitted to *IPSN*, 2007.

[8] S. Funiak, C. Guestrin, M. Paskin, and R. Sukthankar. Distributed localization of networked cameras. In *IPSN*, pages 34–42, 2006.

[9] S. Ganeriwal, R. Kumar, and M. B. Srivastava. Timing-sync protocol for sensor networks. In *SenSys*, pages 138–149, Nov. 2003.

[10] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, second edition, 2004.

[11] D. Hearn and M. P. Baker. *Computer graphics (2nd ed.)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.

[12] J. Hightower and G. Borriello. Location systems for ubiquitous computing. *IEEE Computer*, 34(8):57–66, August 2001.

[13] M. Lourakis. `levmar`: Levenberg-Marquardt nonlinear least squares algorithms in C/C++. Available at `http://www.ics.forth.gr/~lourakis/levmar/`.

[14] D. Lucarelli and I.-J. Wang. Decentralized synchronization protocols with nearest neighbor communica-

tion. In *SenSys*, pages 62–68, 2004.

[15] D. Lymberopoulos, A. Barton-Sweeney, and A. Savvides. Sensor localization and camera calibration using low power cameras. *ENALAB Tech Report 090105*, September 2005.

[16] M. Maroti, B. Kusy, G. Simon, and A. Ledeczi. The flooding time synchronization protocol. In *SenSys*, pages 39–49, 2004.

[17] B. Micusík and T. Pajdla. Estimation of omnidirectional camera model from epipolar geometry. In *CVPR*, pages 485–490, 2003.

[18] D. Moore, J. Leonard, D. Rus, and S. J. Teller. Robust distributed network localization with noisy range measurements. In *SenSys*, pages 50–61, 2004.

[19] N. Patwari, J. N. Ash, S. Kyperountas, A. O. H. III, R. L. Moses, and N. S. Correal. Locating the nodes: Cooperative localization in wireless sensor networks. In *IEEE Signal Processing Magazine*, volume 22, pages 54–69, 2005.

[20] S. Poduri, S. Pattem, B. Krishnamachari, and G. S. Sukhatme. Sensor network configuration and the curse of dimensionality. In *Workshop on Embedded Networked Sensors (EmNets)*, 2006.

[21] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press, New York, NY, USA, 1992.

[22] S. N. Sinha, M. Pollefeys, and L. McMillan. Camera network calibration from dynamic silhouettes. In *CVPR*, pages 195–202, 2004.

[23] R. Stoleru, T. He, J. A. Stankovic, and D. Luebke. A high-accuracy, low-cost localization system for wireless sensor networks. In *SenSys*, pages 13–26, 2005.

[24] A. W. Stroupe, M. C. Martin, and T. R. Balch. Distributed sensor fusion for object position estimation by multi-robot systems. In *ICRA*, pages 1092–1098, 2001.

[25] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon. Bundle adjustment – A modern synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, pages 298–375. Springer Verlag, 2000.