

Sensor Localization and Camera Calibration in Distributed Camera Sensor Networks

Andrew Barton-Sweeney, Dimitrios Lymberopoulos and Andreas Savvides
Embedded Networks and Applications Lab
Electrical Engineering Department, Yale University
New Haven, CT 06520
Email: abs@cs.yale.edu, {dimitrios.lymberopoulos, andreas.savvides}@yale.edu

Abstract—Camera sensors constitute an information rich sensing modality with many potential applications in sensor networks. Their effectiveness in a sensor network setting however greatly relies on their ability to calibrate with respect to each other, and other sensors in the field. This paper examines node localization and camera calibration using the shared field of view of camera pairs. Using a new distributed camera sensor network we compare two approaches from computer vision and propose an algorithm that combines a sparse set of distance measurements with image information to accurately localize nodes in 3D. Our algorithms are evaluated using a network of iMote2 nodes equipped with COTS camera modules. The sensor nodes identify themselves to cameras using modulated LED emissions. Our indoor experiments yielded a 2-7cm error in a 6x6m room. Our outdoor experiments in a 30x30m field resulted in errors 20-80cm, depending on the method used.

I. INTRODUCTION

Low power, low cost camera sensors are expected to play an important role in many types sensor networks ranging from traditional surveillance to pervasive applications in everyday life situations. This is because, in addition to providing information rich image data, small cameras in sensor networks have potential for facilitating a new family of applications functioning as intelligent, motion discriminative sensors. Small low-power cameras with localized intelligence will act as sensors to provide higher fidelity motion information than existing passive infrared (PIR) at lower power and bandwidth cost than conventional cameras. This and the fact that wireless collaboration with other sensors is expected to allow coverage over large areas, necessitates the consideration of new lightweight algorithms for configuring and operating distributed camera sensor networks. Our research explores ultra-low power biomimetic architectures [18] as well as distributed camera sensor networks that operate on symbolic information rather than images [6] and camera node self-configuration.

In this paper we concentrate our discussion on node localization and camera calibration. Our approach is described in the context of a new lightweight camera sensor network testbed and middleware infrastructure we have developed. The two complementary localization algorithms proposed here are designed to calibrate camera nodes and localize other smaller nodes in the field of view of the cameras. In our setup, a fraction of the nodes (the camera nodes) are equipped with CMOS camera modules build on top of Intel's iMote2 platform

while the rest of the nodes use modulated light emissions from a bright red LED to uniquely identify themselves to the cameras. Pairs of camera nodes exchange information about the nodes observed in their field of view and use the proposed lightweight algorithms to compute their relative rotation and translation matrices while also localizing large numbers of smaller, less powerful nodes in their field of view. We anticipate that our approach would be particularly useful when a large number of small sensor nodes needs to be localized. To further motivate our approach, consider the problem of measuring chemical concentrations in a large water reservoir using hundreds of floating sensors. In such situation, the approach described here will simultaneously localize the sensors and collect the chemical concentration measurements without using a radio, by using a few camera sensor nodes to observe modulated LED emissions from the sensors.

Our study and experimental results show that a few simple deployment considerations and the use of imagers can result in lightweight, yet very accurate 3D localization that bypasses some of the challenges posed by rigidity constraints in the case of distance only localization [2]. Our experimental evaluation also verified that distance measurements between camera nodes are not required if a special pattern that includes distance information is included in the deployment. This can be easily done by placing 3 LEDs in a triangle pattern with known sides on the surface of some of the nodes or by simply making nodes to have a specific shape with known dimensions. This distance information is required since cameras cannot perceive depth. Without distance information, locations and translations can only be computed up to scale. We also propose an algorithm that uses a set of sparse known distance measurements to refine locations.

An equally important contribution of our work is the implementation of a lightweight camera sensor network that features an extensible middleware framework. With this framework one can rapidly develop a suite of localization algorithms as well as other applications for camera sensor networks. As we will describe in section IV, our middleware framework includes a set of drivers, libraries and services that can be rapidly configured into new applications using a scripting interface at the basestation.

This paper is organized as follows. Sections II and III describe the localization problem and surveys the related

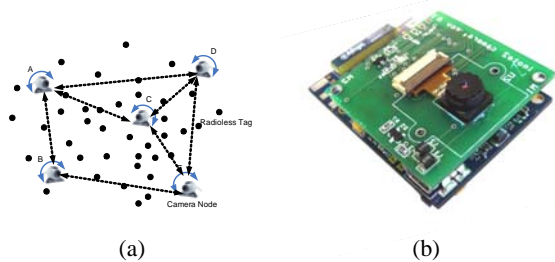


Fig. 1. a) Example localization deployment scenario b) Intel iMote2 node with OV camera module b) localization deployment scenario

work. Section IV describes our sensor network testbed and middleware framework implementation. Sections V and VI described the localization algorithms. Section VIII describes our evaluation in indoor and outdoor environments followed by our discussion and conclusions.

II. PROBLEM STATEMENT

Our presentation in this paper deals with computing the relative translation and rotation matrices between cameras and localizing nodes in 3D, using information between camera views. A practical solution to this problem can easily be extended to multihop scenarios by using the translation and rotation information among adjacent cameras. This process is known as *transfer* in computer vision. With transfer, the coordinate system of any camera i , can be translated to the coordinate system of any other camera j if there is a path of cameras from i to j , on which the relative rotations and translations, among adjacent camera pairs are known. Assuming that transfer is possible because camera nodes can communicate with each other, we state our problem as follows:

Given a network of N sensor nodes $t_1, t_2, t_3 \dots t_N$ where a subset $m < N$ nodes are equipped with cameras, compute all possible node locations and the relative camera translations and rotations with respect to a local coordinate system. A typical deployment setup can be seen in Figure 1b. As we will explain in section 5, the setup we investigate requires some distance information. This can be provided with a distance measurement system on the nodes or by properly placing a set of LEDs in a special pattern on the surface of each node.

Section VII provides an overview of the multihop localization algorithm, but the rest of the discussion and our evaluation focuses on how to exploit camera epipoles when two cameras have one or more nodes in their shared field of view. In such a setup each camera node will compute node positions with respect to its own coordinate system. The knowledge of the relative camera rotations and translations will allow the transformation of coordinates, to either coordinate system. Two algorithms are examined, measured epipoles and estimated epipoles. Camera coverage and mobility issues are discussed in section IX.

III. RELATED WORK

The reconstruction of 3D imagery from images is a problem treated in computer vision for several years. In 1992 the work done by Tomasi and Kanade in [19] has proposed a way

for reconstructing a scene and estimating camera parameters and feature point locations using matrix factorization. This initial work was performed under orthographic projection, and more recent work has treated the paraperspective case [10]. A more complete solution that compensates for the camera depth effects has been described in [15]. More recently, the works of Mantzel et. al. [7] and Devarajan et. al. [1] have begun to consider the problem of camera calibration in a networked setup. Both approaches have examined the problem from a computer vision perspective without the consideration of other sensor node capabilities. The former has proposed an iterative approach for localizing cameras (position, relative translation and orientation up to a scale) using linear relationships and the DLT camera calibration developed by Faugeras in [8]. The work of Devarajan et.al. builds upon, more computationally demanding factorization methods proposed by Sturm and Triggs in [15]. Cameras form microclusters with other nodes in the same coordinate system. The camera localization algorithm requires 4 cameras having at least 12 feature points in their common field of view. The algorithm also provides a scheme for aligning image frames.

Camera calibration using mobile entities such as people has also been considered in the computer vision community. Taylor et. al. at MIT [17] considers camera calibration for cases of non-overlapping field of views. In sensor networks, the problem of ad-hoc node localization has been treated in great detail. Our work is more related to fine grained localization schemes such as the ones that have been demonstrated in MIT's Cricket system [11], [12] and UCLA's AHLoS system [14]. In fact, many of the concepts presented here will be interoperable with the Cricket localization schemes recently presented in [9] or any other similar scheme.

Our approach would also be interpretable with some other approaches such as the one used with MIT Crickets in [9]. It could also be applied on other platforms such as Cyclops [13]. Our work builds up on concepts from computer vision but it is also bears a few differences. We perform an in-situ evaluation in the context of sensor network applications based on the actual camera technology we intend to use. We examine the feasibility of solving the problem using small sensor nodes, and propose an algorithm for reconstructing node coordinates that utilizes deployment information that is more lightweight when compared to the stratification approaches used in computer vision.

IV. TESTBED AND MIDDLEWARE IMPLEMENTATION

To support our research, we have developed a camera sensor network testbed that follows an asynchronous stream-based architecture. The testbed consists of Intel's iMote2 nodes equipped with a COTS camera module we have developed based on Omnivision's OV7649 VGA camera shown in Figure 1b. Our reconfigurable stream-based architecture is implemented using modules in the SOS operating system [3]. Each module in the system adheres to a generic structure that supports a publish/subscribe service model.

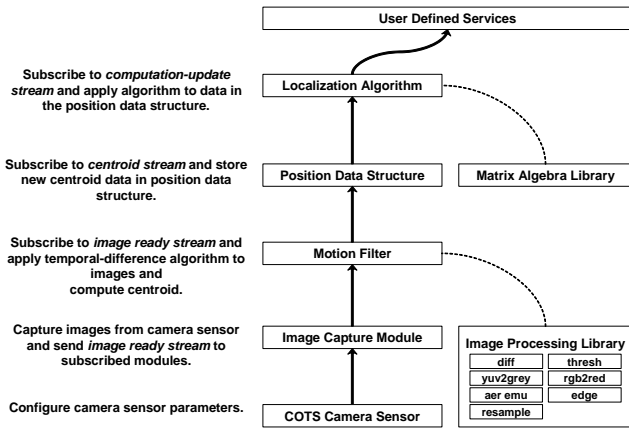


Fig. 2. Module stack inside a sensor node

Using this service model, modules and other entities can subscribe modules to receive stream outputs from other modules by submitting subscribe requests through messages. Messages are interpreted by a generic message handler that inserts the module ID of the subscribing module to the output table of the module that received the subscribe request. The module that implements a service will send a stream of messages to the modules registered in its output table.

By adopting this simple structure, any module can subscribe to the outputs of any other module in the network. The subscriber module can be found on the same node, a different node or a remote base-station. This convention allows the online development of distributed camera applications in a deployed network by facilitating remote debugging. The developer can subscribe to the outputs of any module in the module hierarchy and check on its output inside the live network.

The wiring of the modules can be done in two ways. During development, the developer can issue messages from a basestation to edit the wiring configuration of the modules on a node. During deployment, a composer module on a node can setup the wiring configuration of the node. The composer module will wire the modules together and initialize them every time the node boots.

The internal software architecture for our camera localization scheme is shown in Figure 2. All measurements, node locations and camera rotations and translations are stored in the position data structure. The position data structure triggers a new localization computation every time new measurements are available from the sensor. The localization computation process is illustrated in Figure 3.

Our system also implements a small image processing library and a matrix manipulation library. The image processing library includes functions such as temporal differencing, thresholding, edge detection, yuv2grey, rgb2red and re-sample operations. These together with another library are also used to provide emulation of address-event image sensors that are used in other parts of our research [18].

Building up on the services provided here, one could use

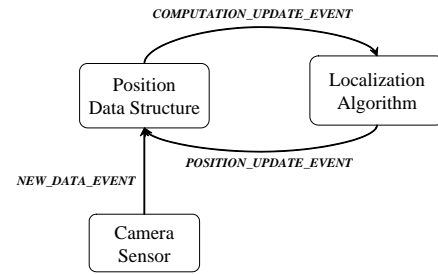


Fig. 3. Localization computation process

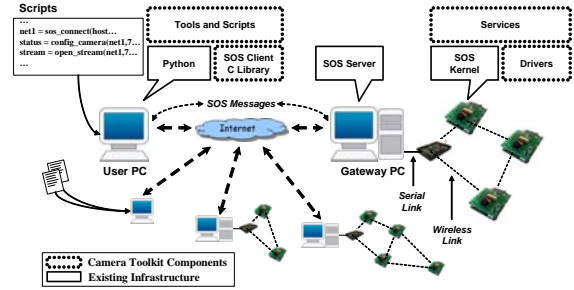


Fig. 4. Distributed camera sensor network components

our camera sensor network as a toolkit for developing other types of localization algorithms as well as a wide variety of other applications in security and assisted living. For instance, any node or a basestation can request an image acquisition from any camera node by sending a capture message. Once the image is acquired, the requestor can either receive the raw image or it can request other operations to be performed on that image. Additional sensing modalities and localization algorithms are easily integrated into our system by loading the sensor or algorithm modules onto the node and registering (wiring) the new module to the position data-structure module. The sensor and algorithm modules do not need to reside on the same node as the position data structure module, and may even run at a remote basestation.

At the basestation, our system provides a scriptable interface to the network that allows a developer to easily reconfigure the network and collect sensor data. The scripts are written in Python and connect to the network using a C library that communicates with a remote SOS server. The system components are shown in Figure 4. The developer can run the scripts interactively to wire modules in the network and reconfigure the sensor, or as a daemon process in the background to store data from the network to a database or automatically provide configuration or calibration data to the network on demand from the sensors.

Modules in the sensor network communicate by SOS messages that contain a simple eight-byte header and a data payload. A gateway node in the sensor network connects through a serial cable to a PC that runs the SOS server program. The SOS server accepts connections from remote computers

Fig. 5. The system components for the camera sensor network and remote scriptable interface.

through TCP sockets, and the clients can send and receive SOS messages to modules in the sensor network through the server as shown in Figure 4. We created a C library of routines to communicate with the sensor network through the SOS server. To facilitate rapid development of drivers and services in the sensor network, we wrapped the SOS client library as an extension module for the Python scripting language. We chose Python because the language is scriptable very easy to learn and use. The connections to sensor networks are represented as Python objects and a Python script can simultaneously connect to multiple sensor networks. The Python language allows us to quickly create scripts that configure the network, collect data and apply complex algorithms. Using the Python scripts, we can prototype new services on a PC that seamlessly interact with modules in a remote sensor network by sharing SOS messages as if the services are running inside the sensor network.

Given this camera sensor network, we now focus or discussion on node localization and camera calibration. More information about our camera sensor network infrastructure can be found at <http://enaweb.eng.yale.edu/drupal/node/CameraNetwork.htm>.

V. SENSOR ASSISTED CAMERA LOCALIZATION

A. Background

The extrinsic camera calibration parameters of interest are a rotation matrix $R_{3 \times 3}$ and a translation vector $T_{3 \times 1}$, also referred to as the extrinsic calibration parameters. In the absolute sense, the two matrices R and T represent the camera coordinates with respect to a 3-D world origin O . In general, our discussion about cameras will deal with three types of coordinates. World coordinates w are the coordinates given with respect to a real 3-D world origin. The camera also has its own camera centric coordinates \bar{w} that can also be expressed in world coordinates with (1).

$$\bar{w} = R w + T \quad (1)$$

The image coordinates u, v observed by the camera, can also be related to the camera centric coordinate system though the following equations.

$$\frac{u}{f} = \frac{\bar{w}_x}{\bar{w}_z}, \frac{v}{f} = \frac{\bar{w}_y}{\bar{w}_z} \quad (2)$$

where f is the focal length of the camera. The above expressions can be obtained by considering the ratio of triangles as shown in figure 6.

Our work uses the epipoles between pairs of cameras. The epipoles between a pair of cameras are defined as the points where a straight line connecting the two camera centers intersect the image plane of each camera [4] as shown in Figure 6.

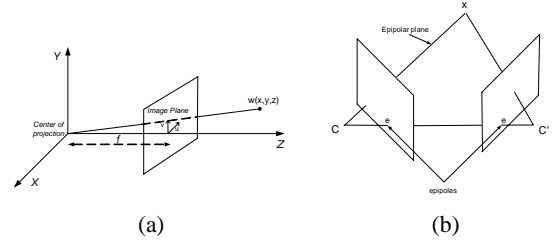


Fig. 6. a) Camera diagram. Camera coordinate system z -axis is piercing the normalized (u, v) image plain in the image plain origin, b)epipoles between a pair of cameras .

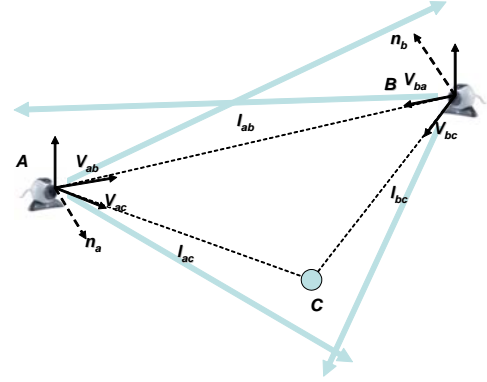


Fig. 7. Measured epipole: A tag can be observed by 2 cameras that can also observe each other

Throughout the paper we use the following notation:

l_{ab} - distance between nodes A and B

v_{ab} - unit vector from A to B

u_a, v_a - x, y coordinates of an object on the image plane of camera A .

B. Localization Based on Two Camera Views

We now examine two methods for extracting camera epipoles, direct observation, and fundamental matrix estimation.

1) Direct Epipole Observation - Measured Epipoles(ME):

: If a tag C is observed by two cameras A and B that can also observe each other, the distances between A, B, C can be determined up to a scale. This was demonstrated in [16].

If cameras A and B can observe each other and a tag C , then we can derive the unit vectors v_{ab}, v_{ac}, v_{ba} (see Fig. 7), and v_{bc} . From these we can derive the normalized versions $n_a = v_{ab} \times v_{ac}$ and $n_b = v_{ba} \times v_{bc}$. Note that v_{ab}, v_{ba}, n_a and n_b are related by a rotation matrix R_{ab} , the relative orientation matrix between cameras A and B .

$$v_{ab} = -R_{ab} v_{ba}$$

$$n_a = -R_{ab} n_b$$

From the two perpendicular unit vectors v_{ab} and n_a , we can construct the orthonormal matrices R_a and R_b . $R_a = \begin{bmatrix} v_{ab} & n_a & (v_{ab} \times n_a) \end{bmatrix}$ and $R_b = \begin{bmatrix} v_{ba} & -n_b & (v_{ba} \times n_b) \end{bmatrix}$.

Substituting we get the relative orientation as

$$R_{ab} = R_a(R_b)^T \quad (3)$$

from this we can extract the following linear system

$$l_{ab}v_{ab} + l_{bc}(R_{ab}v_{bc}) - l_{ca}v_{ac} = 0 \quad (4)$$

Solving these equations we can get l_{ab} , l_{ac} and l_{bc} up to a scale. Since we also know the distance l_{ab} we can solve to Euclidian scale.

2) *Extracting the Epipoles from the Fundamental Matrix - Estimated Epipoles (EE)*: : If two cameras are not facing each other, then the epipoles cannot be directly observed. Nonetheless, the camera epipoles can still be determined if the camera observations can provide enough information to compute the fundamental matrix between two cameras [5]. Although this can be done with a minimum of 5 nodes in the common field of view of two cameras, the resulting problem is highly non-linear and very difficult to solve. In our implementation we chose to use the widely used normalized eight-point algorithm proposed by Hartley in 1997 [4].

The eight-point algorithm uses eight or more points in the common field of view of a pair of cameras to estimate the fundamental matrix F between them. F is defined by the equation

$$u'^T F u = 0$$

u' and u are corresponding feature points in the images of the two cameras. The epipoles of the two cameras can be then extracted from F . For any point x , the epipolar line $l' = Fx$ contains the epipole e' . Thus $e'^T(Fx) = (e'^T F)x = 0$ for all x . From this it follows that $e'F = 0$ and $Fe = 0$, thus the epipoles of the two cameras, e' and e are the left and right null vectors of F respectively [5]. The estimation of both camera's epipoles allows us to compute the rotation between the two cameras and the distances from both cameras to all the nodes up to a scale using the formulation described in the previous subsection. The only difference in this case is that instead of using the measured epipoles, as in ME, we use the estimated, from the fundamental matrix, epipoles.

VI. OPTIMIZED ESTIMATED EPIPOLES (OEE)

The resulting epipole estimates are noisy and thus will provide noisy distance estimates. An illustrative example of the error based on our testbed measurements is shown in Fig. 8. The figure shows the distance error ratio for ground truth (GT), ME and EE when the distances between the cameras A and B and the nodes are computed (by equation 4). The figure shows that ME produces accurate measurements comparable to ground truth while EE has noisy measurements. Furthermore note that the error from EE at the two cameras is complimentary. If a distance between camera A and a node is overestimated then the distance between the same node and camera B will be underestimated and vice-versa. This is of course an artifact of the lightweight algorithm we use. It does however suggest that refinement is possible by attempting to match the two camera views that have complimentary error

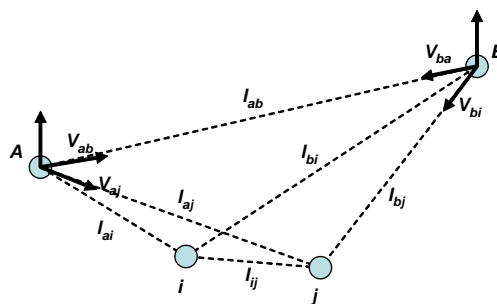


Fig. 10. Imposing the constraint that two cameras should agree on the l_{ij} estimate

as shown in Fig. 9. The alternative is to use stratification, a much more complex algorithm from computer vision [5], that is not suitable for small sensor nodes.

To reduce this error, we formulate a more constrained optimization problem, where the distances between the observed nodes and the camera nodes are estimated simultaneously. To enforce matching views between the cameras, we also consider the additional constraints that the pairwise distances between all the nodes observed by the two cameras should be equal from the view point of both cameras. Consider the scenario in Figure VI with cameras A and B and observed nodes i and j . The distance l_{ij} can be estimated by both cameras as

$$\begin{aligned} {}^A l_{ij} &= \|l_{ai}v_{ai} - l_{aj}v_{aj}\| \\ {}^B l_{ij} &= \|l_{bi}v_{bi} - l_{bj}v_{bj}\| \end{aligned} \quad (5)$$

If we assume that some of the l_{ij} distances are known, we can impose these as constraints. The rest of the analysis is identical for both cameras, therefore we will focus on camera A . The required camera-to-node distances can be estimated up to a scale with respect to camera A 's view point as:

$$\begin{aligned} l_{ab}v_{ab} + l_{bi}(R_{ab}v_{bi}) - l_{ai}v_{ai} &= 0 \\ l_{ab}v_{ab} + l_{bj}(R_{ab}v_{bj}) - l_{aj}v_{aj} &= 0 \end{aligned} \quad (6)$$

As it was described in the previous section, in the case of the estimated epipoles method(EE), the computed distances from the camera to the nodes will be erroneous. These erroneous distances can be refined if a subset of distances between the nodes seen by the camera is known. Assuming that n distances $d_{i_x j_x}$, $x = 1, 2, \dots, n$ are known, the following equations should hold:

$${}^A l_{i_x j_x} = \|l_{ai_x}v_{ai_x} - l_{aj_x}v_{aj_x}\| = d_{i_x j_x} \quad (7)$$

where x is an index running over all n known edges and i_x , j_x are the nodes connected by the x^{th} edge. In order to refine the distance estimates l_{ai_x} and l_{aj_x} based on the known n distances we would need to minimize the following function:

$$L = \min \sum_x (d_{i_x j_x} - \|l_{ai_x}v_{ai_x} - l_{aj_x}v_{aj_x}\|)^2 \quad (8)$$

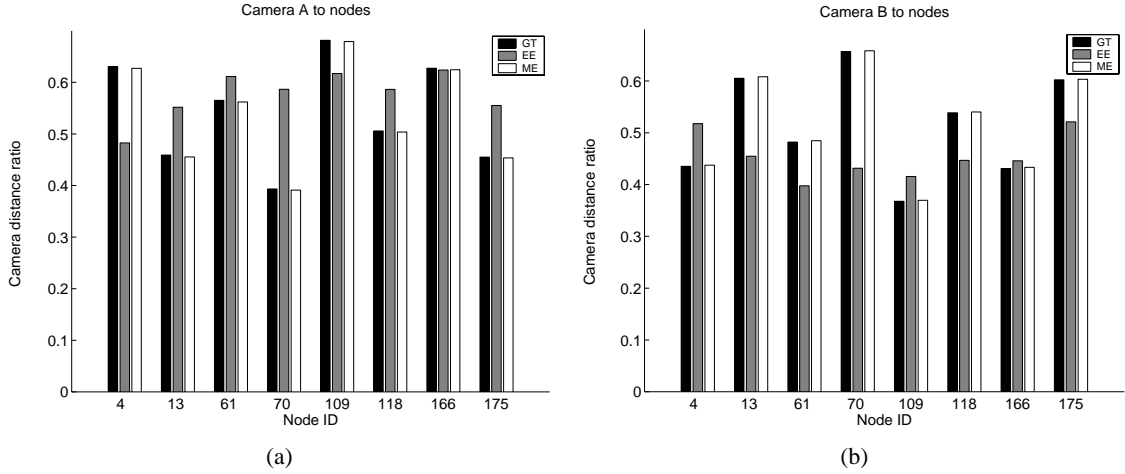


Fig. 8. Estimated distances between two camera nodes and other observed nodes. GT - Ground truth, EE - estimated epipoles, ME - measured epipoles

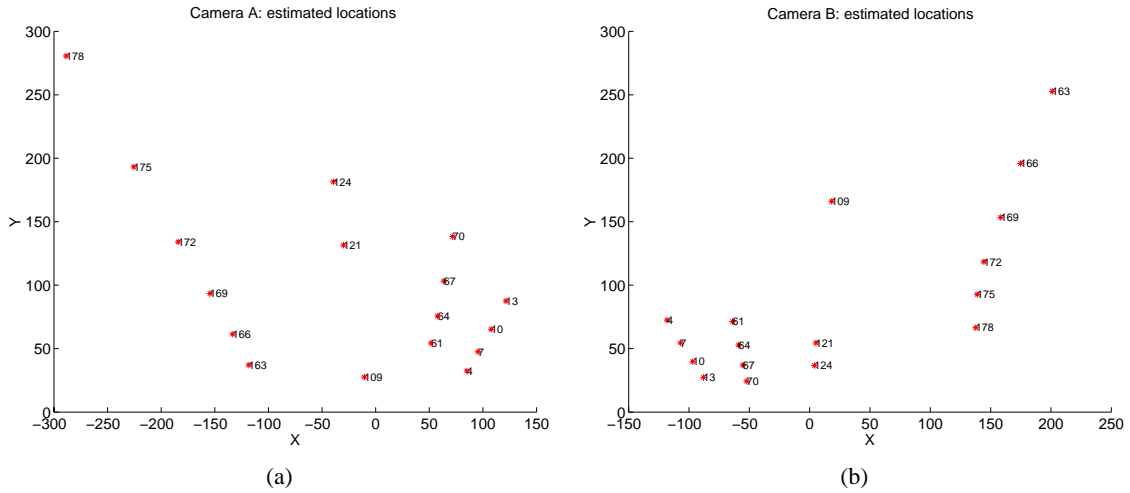


Fig. 9. Estimated locations using an optimized version of the distances in Fig. 8

Since x is running over all n known distances, function L is basically a set of n equations where the number of unknowns depends on the known edges. If all the n known edges are independent¹ the number of variables is $2n$ because in that case each edge (i, j) would involve two unique unknowns: l_{ai} and l_{aj} . Since the number of equations is less than the number of unknowns the minimization of function L is impossible. However, if different known edges share common nodes then the minimization of L is possible. The simplest case, where the minimum number of edges is needed and function L can be minimized, is when all the distances among three nodes are known. In other words, when all the edges of a triangle that is formed by three nodes are known the set of equations represented by L can be solved. Each edge of the triangle provides an equation giving a total of 3 equations. For each node in the triangle there is only one unknown. Therefore the number of unknowns is equal to the number of equations and the distances from camera A to the nodes forming the triangle can be refined. This set of equations is a non-linear set of equations that can be solved with an Extended Kalman

Filter(EKF) or another gradient descent method.

Note that any other closed geometric shape (square, pentagon, hexagon etc) could be used in the optimization process. We use triangles because they require the minimum number of known distances (only 3) and therefore they require the minimum number of local information. Therefore based on local triangles we can optimize the distances between the cameras and the non-camera nodes with respect to the camera's coordinate system. After the refinement of the distances between the camera nodes and the non-camera nodes, the rotation matrix between two camera nodes with overlapping fields of view can be refined using equation 6 and the already refined distances.

The preceding ME, EE and OEE algorithms provide us with a means of bootstrapping a coordinate system and also computes the relative rotation and translation between a pair of cameras. Next section shows how these two components can be applied on a distributed localization algorithm.

VII. MULTIHOP LOCALIZATION ALGORITHM OVERVIEW

Although our focus in this paper is on how to perform calibration based on two camera views, we briefly describe an

¹Two edges are independent if they do not share any common nodes

example of a distributed localization algorithm. This algorithm assumes the existence of a coordinate transformation service that runs in the background and distributes the rotations and transformations between pairs of cameras so that nodes can be localized on demand in multiple coordinate systems, using *transfer*.

Immediately after deployment(or when triggered), each node broadcasts a list of IDs for all the nodes it can observe to its radio neighbors. The camera nodes process these broadcasts and creates a vision graph that connects camera nodes that have one or more nodes in their shared field of view. The same information also allows each camera node to identify which other nodes can use its observations to run the ME or EE algorithm. Once a camera node identifies these nodes, it forwards them the observations (node id and image coordinates) for the observed nodes in their shared field of view.

After this phase, each camera node has the required observations to bootstrap its local coordinate system by running ME or OEE. ME and OEE provide a distance(magnitude) and a vector(direction) for each node they consider. This allows the camera node to estimate the coordinates of each of the considered nodes in its own coordinate system. The relative rotation between the two nodes is also computed, and its passed to the coordinate transformation service. At this point the coordinate transformation service is also notified to check if the new information generated will allow computing camera transformations from other nodes.

If a new node appears in a camera’s field of view, the camera repeats a similar process to discover if any of its neighbors can observe the same node. Depending on the number of nodes available in the shared field of view each camera node may execute ME, EE or simply compute the node’s location based on previously computed rotation information. Do to space limitations, the details and evaluation of the multihop localization and the coordinate transformation service are omitted in this paper. These will be treated in a subsequent paper.

VIII. EVALUATION

The evaluation of the the ME, EE and OEE algorithms is performed on real measurements obtained from an indoor and outdoor dataset collected using our camera enabled sensor nodes. Each node is equipped with an extension board carrying a COTS OV7649 camera module from OmniVision. The node processor can acquire frames from the camera and perform some basic feature extraction, such as sobel edge detection, differencing to detect motion and LED identification. All nodes also carry a Lumex CCI-CRS10SR omnidirectional LED with an axial intensity of 40 millicandellas. In our lab setup, the cameras nodes can clearly observe these LEDs for distances up to 4 meters. In our tests, camera nodes can uniquely identify the nodes using an asynchronous protocol which can read the node id of each node by toggling the LED to send bits across.

Using this platform we acquired 2 datasets, one indoor and one outdoor. The indoor scenario was comprised of 2 camera nodes and 16 non-camera nodes identified with blinking LEDs. The two camera nodes were placed at different heights and angles with respect to the non-camera nodes yielding four different datasets. For the outdoor scenario, we placed 80 bright orange postit nodes in an outdoor plaza next to our building. In this outdoor test we were interested in the accuracy and range of our system, and we assumed that the correspondences for each post it node were known ². The layout of the outdoor scenario is shown in Fig VIII.

Before evaluating our algorithms, we also verified that the error varies linearly with camera resolution. We verified this with a characterization of the measurement error across four different camera resolutions: 640 x 480 (VGA), 352 x 288 (CIF), 240 x 180, and 128 x 96 (SQCIF). Our evaluation was done using the ME method using a topology of two camera nodes and 8 tags as shown in Figure 11a. The maximum error was 3.32 cm in the lowest resolution. Figure 11b describes the measurement error as a function of the normalized pixel area for each resolution, as computed using the pairwise distance measurements. With the exception of the VGA resolution, the error follows a linear trend $y = 2.8 \cdot 10^5 x + 2$. This result verifies our intuition that error scales with resolution and also indicates that the lens distortion effects are almost negligible.

A. ME, EE, OEE Evaluation

The performance of the ME, EE and OEE algorithms in the case of one of the indoor experiments can be seen in Figure 13. The accuracy of the ME method shows that the camera is a very reliable measuring modality. On the other hand, the poor performance of the EE algorithm shows that the estimates of the epipoles are erroneous and therefore unreliable. However, as it can be seen in Figure 13 the OEE method proposed in this paper reduces drastically the error and performs almost equally well to the ME method.

The drastic reduction in the error in the case of the OEE method can be seen better in Figures 14 and 15 where the empirical CDF of the node-to-node distance error is shown. In the indoor setup, EE reports an error of 60cm with probability 90% while OEE reports an error of 7cm with the same probability. Given that the average node-to-node distance in the indoor setup is approximately 85cm the OEE algorithm performs fairly well. The ME algorithm has the best performance reporting an error of 2cm with 90% probability. Figures 14 and 15 show very similar results for the outdoor experiment. Again, ME has the best performance(20cm error with 90% probability) but OEE performs comparably well reporting an error of 60cm with 90% probability given that the average node-to node distance is approximately 297cm. In the case of the outdoor setup a more detailed view of the performance of all the methods can be seen in Figure 16.

To compare our results to other approaches we also evaluate

²The image correspondences can be found by methods from computer vision not described here

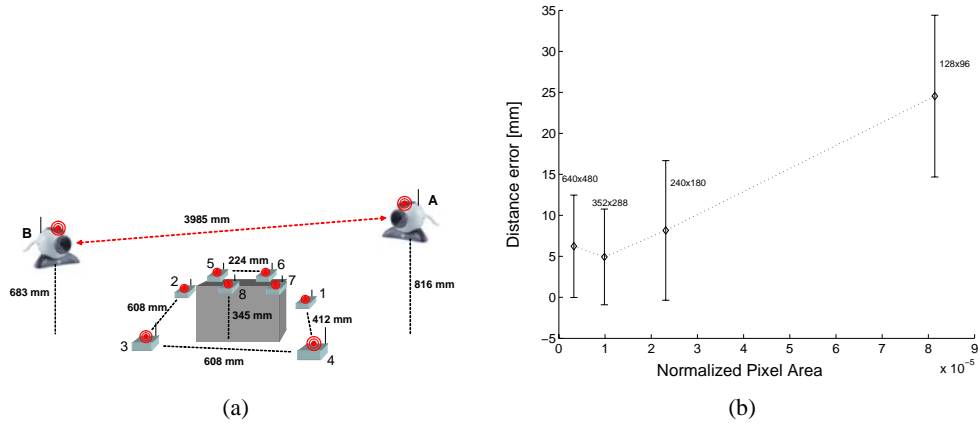


Fig. 11. a) Characterization scenario using 8-tags and 2 camera nodes, b) Error across different resolutions

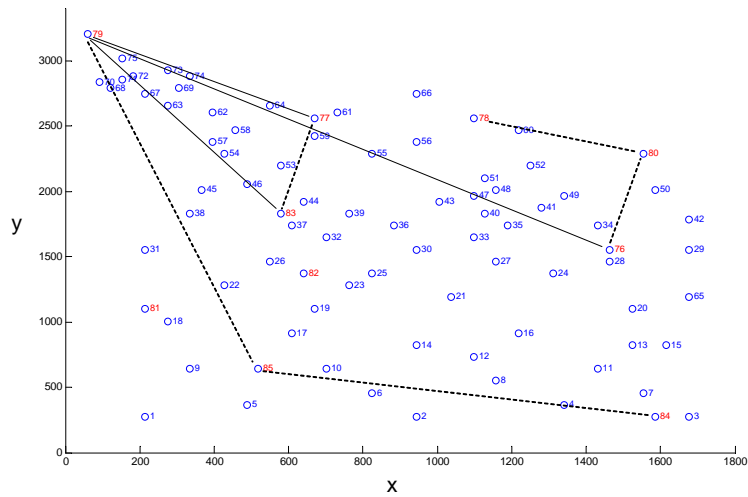


Fig. 12. Layout of the outdoor setup. Camera node pairs that can run ME are connected with solid lines and camera pairs that can run EE and REE are connected with dotted lines. The units are in cm.

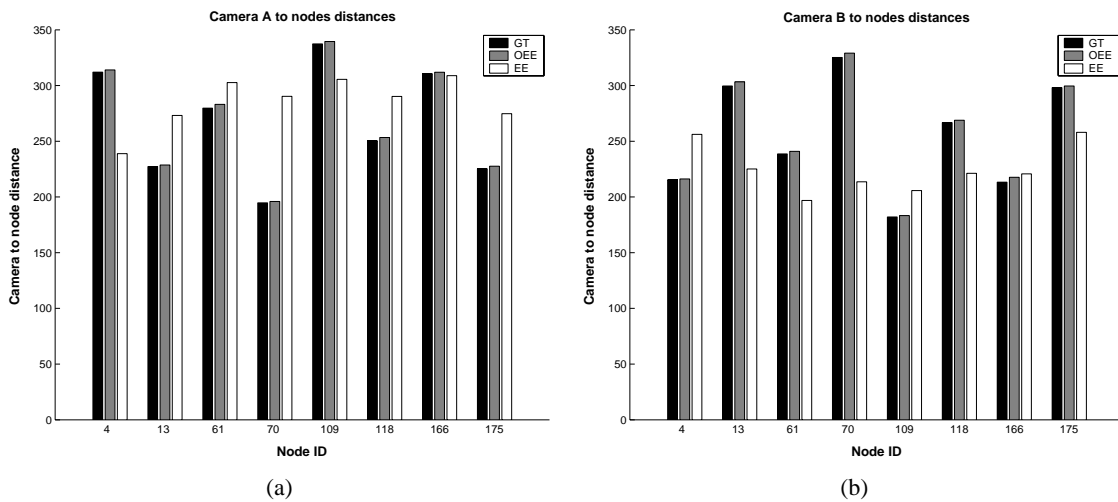


Fig. 13. Estimated distances between two camera nodes and other observed nodes for the indoor experiment. GT - Ground truth, OEE - optimized estimated epipoles, EE - estimated epipoles. All distances are in cm.

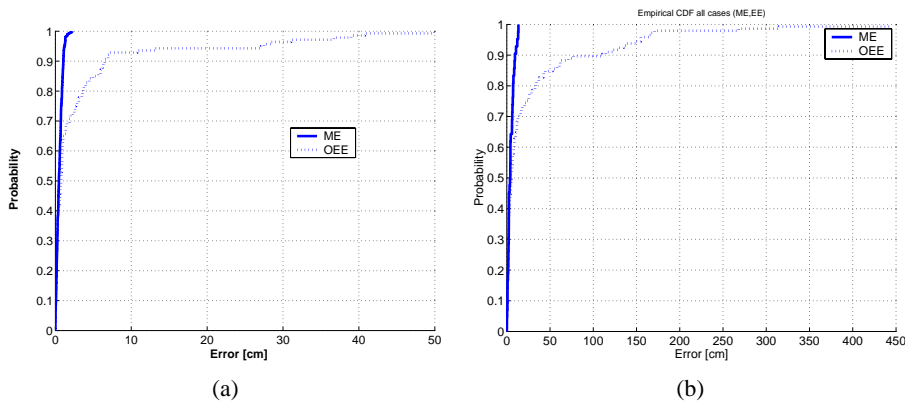


Fig. 14. Empirical CDF for node to node distance estimates a) all indoor scenarios with optimized EE, b) all outdoor scenarios with optimized EE

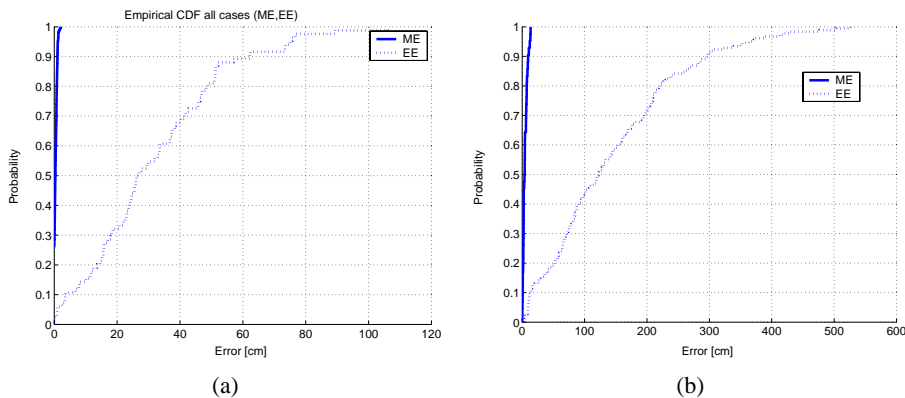


Fig. 15. Empirical CDF for node to node distance estimates a) all indoor scenarios with unoptimized EE, b) all outdoor scenarios with unoptimized EE

the estimated internode distance error with respect to the follow two metrics:

$$p = \sum_{i=1}^N \frac{|l_i - \hat{l}_i|}{N} \quad (9)$$

and

$$q = \sqrt{\sum_{i=1}^N \frac{(l_i - \hat{l}_i)^2}{N}}. \quad (10)$$

N is the number of pairwise distances, l_i are the measured distances and \hat{l}_i are the estimated pairwise distances computed from the localization results of the algorithm. Metric (9) is the *mean error*. Metric (10) is the *square root of the mean square error* and it is also used in [9]. The standard deviation of the error, noted by σ , is also computed. Table I summarizes our results and compares to the ultrasonic approach from [9]. At a room level scale, the ME and OEE approaches are more accurate than ultrasound. In the outdoor scenarios the error is much higher. This is expected since the average internode distances are much larger. In this case camera accuracy degrades since the small angular error of the camera measurements increases tangentially with distance.

Algorithm	Indoor		Outdoor	
	$p(cm)$	$q(cm)$	$p(cm)$	$q(cm)$
ME	4.0165	0.3779	17.9794	1.2493
OEE	0.9714	1.4159	29.6057	51.16818
Ultrasound	7.02	5.18	N/A	N/A

TABLE I

THE p AND q DISTANCE METRICS FOR THE ME AND OEE ALGORITHMS AS WELL AS FOR THE ULTRASOUND TECHNOLOGY IN THE CASE OF BOTH INDOOR AND OUTDOOR EXPERIMENTS.

IX. DISCUSSION

Localization and camera calibration when sensors can generate stimuli that allows them to communicate with cameras is appealing in some applications since it implies one could easily build very simple nodes that cost a few cents each with today's technologies. Imagine for instance, thousands of floatable chemical sensors dispersed to measure chemical concentrations in the sea, in water reservoirs or large tanks. Such nodes would consist of a small battery, a tiny 4-bit microcontroller and an LED serving both as a communication and localization device. A smaller set of camera nodes can observe them.

During our evaluation we also realized that camera placement is a main problem. To have adequate field of view, the cameras need to be mounted on the walls or the ceiling or need to be placed in positions higher than other nodes facing down.

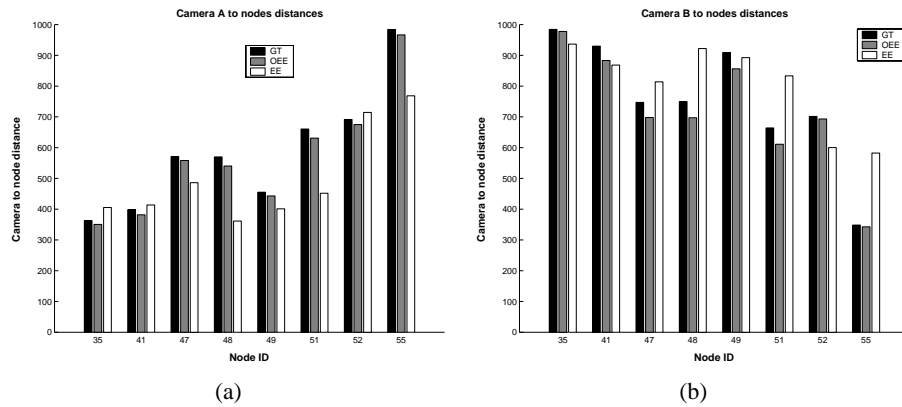


Fig. 16. Estimated distances between two camera nodes and other observed nodes for the outdoor experiment. GT - Ground truth, OEE - optimized estimated epipoles, EE - estimated epipoles. All distances are in cm.

This together with the limited field of view of cameras suggest that articulation, and autonomous motion are necessary for many camera enabled sensor networks. It also makes a case for using cameras together with other sensors that can help overcome the limited field of view of the cameras. Another issue in the deployment of camera networks is privacy. We plan to handle this issue with the custom designed cameras under development in our project [18]. These cameras will be blind to images and will only extract features and sensor stimuli, preserving the privacy of their users.

X. CONCLUSIONS AND FUTURE WORK

In this paper we have compared using a real sensor network two basic computer vision algorithms from computer vision. Using the results of our evaluation we have developed a refinement algorithm that is more lightweight than traditional computer vision algorithms and therefore more suitable for resource constrained sensor nodes. As part of our future work, we plan to use these results and our reconfigurable middleware framework to localize and track events, other than sensor nodes. The current infrastructure will allow us to leverage the collaboration of imager and non-imager sensors to identify other events, targets and behaviors in sensor networks.

ACKNOWLEDGEMENT

This work was funded in part by the National Science Foundation CISE/CNS award #0448082. The authors are also indebted to Lama Nachman from Intel for her help with iMote2.

REFERENCES

- [1] D. Devarajan and R. Radke. Distributed metric calibration for large-scale camera networks. In *First Workshop on Broadband Advanced Sensor Networks (BASENETS) San Jose conjunction with BroadNets 2004*, October 25 2004.
- [2] D. Goldenberg, A. Krishnamurthy, W. Maness, Y. R. Yang, A. Young, A. Morse, A. Savvides, and B. Anderson. Network localization in partially localizable networks. In *To appear in the proceedings of INFOCOM 2005*, April 2005.
- [3] C.-C. Han, R. Kumar, R. Shea, E. Kohler, and M. Srivastava. A dynamic operating system for sensor nodes. In *Proceedings of the Third International Conference on Mobile Systems, Applications, And Services (Mobisys)*, 2005.

- [4] R. Hartley. In defense of the eight-point algorithm. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, June 1997.
- [5] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge Press, 2003.
- [6] D. Lymberopoulos, A. Ogale, A. Savvides, and Y. Aloimonos. A sensory grammar for inferring behaviors in sensor networks. In *Proceedings of Information Processing in Sensor Networks (IPSN)*, April 2006.
- [7] W. Mantzel, H. Choi, and R. Baraniuk. Distributed camera network localization. In *Proceedings of the 38th Asilomar Conference on Signals, Systems and Computers*, November 2004.
- [8] S. Maybeck and O. Faugeras. A theory of self-calibration of a moving camera. In *International Journal of Computer Vision* 8(2):123-151, 2004.
- [9] D. Moore, J. Leonard, D. Rus, and S. Teller. Robust distributed network localization with noisy range measurements. In *Proceedings of ACM SenSys, Baltimore, Maryland*, November 2004.
- [10] C. Poelman and T. Kanade. A paraperspective factorization method for shape and motion recovery. In *IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 19, NO 3*, March 1997.
- [11] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *Proceedings of 6th ACM Mobicom, Boston, MA*, August 2000.
- [12] N. B. Priyantha, A. Miu, H. Balakrishnan, and S. Teller. The cricket compass for context-aware mobile applications. In *Proceedings of the 7th ACM Mobicom, Rome, Italy*, July 2001.
- [13] M. Rahimi, D. Estrin, R. Baer, H. Uyeno, and J. Warrior. Cyclops: image sensing and interpretation in wireless networks. In *Second ACM Conference on Embedded Networked Sensor Systems, SenSys, Baltimore, MD*, November 2004.
- [14] A. Savvides, C. Han, and M. B. Srivastava. Dynamic fine grained localization in ad-hoc sensor networks. In *Proceedings of the Fifth International Conference on Mobile Computing and Networking, Mobicom 2001, Rome, Italy*, pages pp. 166–179, July 2001.
- [15] P. Strum and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *In Proceedings of European Conference on Computer Vision*, pages 709–720, 1996.
- [16] C. Taylor. A scheme for calibrating smart camera networks using active lights. In *SenSys04 Demonstration Session, Baltimore, MD*, November 2004.
- [17] C. Taylor, A. Rahimi, J. Bachrach, and H. Shrobe. Simultaneous localization and tracking in an ad hoc sensor network. In *AI Lab Technical Report AIM-2005-016*, 2005.
- [18] T. Teixeira, E. Culurciello, D. L. J. Park, A. Barton-Sweeney, and A. Savvides. Address-event imagers for sensor networks: Evaluation and programming. In *Proceedings of Information Processing in Sensor Networks (IPSN)*, April 2006.
- [19] C. Tomassi and T. Kanade. Shape and motion from image streams: a factorization method. In *Carnegie Mellon Technical Report CMU-CS-92-104*, 1992.