# Calibrating Distributed Camera Networks

Dhanya Devarajan, Zhaolin Cheng, and Richard J. Radke*
Department of Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute, Troy, NY 12180

**Abstract**

Recent developments in wireless sensor networks have made feasible distributed camera networks, in which cameras and processing nodes may be spread over a wide geographical area, with no centralized processor and limited ability to communicate a large amount of information over long distances. This paper overviews distributed algorithms for the calibration of such camera networks- that is, the automatic estimation of each camera's position, orientation, and focal length. In particular, we discuss a decentralized method for obtaining the vision graph for a distributed camera network, in which each edge of the graph represents two cameras that image a sufficiently large part of the same environment. We next describe a distributed algorithm in which each camera performs a local, robust nonlinear optimization over the camera parameters and scene points of its vision graph neighbors to obtain an initial calibration estimate. We then show how a distributed inference algorithm based on belief propagation can refine the initial estimate to be both accurate and globally consistent.

## I. Introduction

Modern urban life is characterized by the ubiquity of digital cameras. We are constantly imaged by our friends' cell phones and digital cameras, the surveillance cameras in subway stations, busy streets, and shopping malls, and even mapping cameras on trucks or satellites. Many of these devices can now communicate wirelessly, so that the set of cameras can be viewed as a wide-area sensor network with thousands of nodes. Clearly, the proliferation of large numbers of interconnected cameras in the public sphere raises legitimate privacy concerns (e.g., see the article by Widen in this special issue).

In contrast, here we are motivated by scenarios in which a camera network may be the best way to obtain time-critical information about an emergent situation where the safety or security of human lives

* Please address correspondence to R.J. Radke, rjradke@ecse.rpi.edu. Dhanya Devarajan is now with Immersive Media, Portland, OR. Zhaolin Cheng is now with Captira Analytical, Albany, NY. This work was supported in part by the US National Science Foundation, under the award IIS-0237516.

is at stake, such as a natural disaster site, an urban combat zone, or a battlefield. Camera networks will be essential for 21st century military, environmental, and surveillance applications [1], but pose many challenges to traditional computer vision.

Until recently, computer vision research on collections of tens or hundreds of cameras has generally taken place in a controlled environment with a fixed camera configuration. For example, many research labs have designed rooms in which the walls and ceiling are studded with cameras, for the purposes of 3D model acquisition and virtual reality (e.g., [2], [3], [4]). To undertake a computer vision task, images from all cameras are quickly communicated to a central processor.

The same experimental assumptions clearly do not apply to real-world scenarios in which battery-powered cameras are spread over a wide geographical area. For example, camera nodes may be quickly deployed by soldiers moving rapidly through hostile terrain, or first responders moving through a dangerous disaster zone. Accurate initial positions and orientations of such cameras will be unknown; even if some nodes are equipped with GPS receivers, these systems cannot be assumed to be highly accurate and reliable [5], nor do they operate indoors. The nodes are unsupervised after deployment, and generally have no knowledge about the topology of the broader network [6]. Most nodes are unable to communicate beyond a short distance due to power limitations and short-range antennas, and communication must be kept to a minimum because it is power-intensive. Furthermore, a realistic camera network is constantly in motion. The number and location of cameras changes as old cameras wear out and new cameras are deployed to replace them. Precipitation, wind, and seismic events will jolt the cameras, or remote directives may reposition or reorient them for a variety of tasks.

Our main interest in this paper is the calibration of a distributed camera network. That is, how can we automatically estimate the three-dimensional location and orientation of each camera, as well as any variable intrinsic parameters of the cameras such as their focal lengths? Accurate estimates of these parameters are critical for enabling good performance on higher-level collaborative computer vision tasks that the camera network may undertake, such as multiple object tracking, three-dimensional reconstruction of scene objects, novel view synthesis, or efficient image-based routing.

To make our algorithms applicable to emerging real-world scenarios, we began our research with two guiding principles. First, the estimation problems we consider must be solved using *distributed algorithms*, as opposed to requiring images or data from all cameras to be transmitted to a single, centralized processor. Second, we strive for algorithms that make *efficient use* of the underlying communication links in the network, instead of assuming that large amounts of data can be transmitted by the sensor nodes with no penalty.

The paper is organized as follows. Section II introduces notation and terminology related to camera calibration. Section III reviews approaches to estimating the *vision graph* for a distributed camera network, in which each camera is represented by a node, and an edge appears between two nodes if the two cameras jointly image a sufficiently large part of the environment. We emphasize an algorithm we recently proposed in [7]. In Section IV we review approaches for distributed estimation of the calibration parameters of all cameras in the network based on the vision graph. We focus on an efficient method for initializing and refining the calibration estimates we recently proposed in [8], [9]. Throughout Sections III and IV, we illustrate our results on a running example of a set of images acquired from cameras distributed throughout RPI's campus. Section V concludes the paper with discussion and ideas for future work.

## II. Notation and Terminology

Formally, *camera calibration* is the estimation of the parameters that describe how a perspective camera projects 3-D points in a fixed world coordinate system to 2-D points on an image plane. These calibration parameters can be divided into four or more *internal* parameters related to a camera's optics, such as its principal point, aspect ratio, lens distortion, and focal length, and six *external* parameters, namely the rotation angles and translation vector that relate the camera coordinate frame to the world coordinate frame. These various parameters can be encapsulated in a $3 \times 4$ camera matrix $P = K[R \mid t]$, where $K$ is an upper triangular matrix that is purely a function of the internal parameters, $R$ is a rotation matrix parameterized by three angles, and $t$ is a translation vector in $\mathbb{R}^3$. Since the fixed internal parameters of a given camera can usually be estimated individually prior to deployment [10], [11], [12], we are mainly concerned with estimating the external parameters, though we include the focal length as an unknown parameter to be estimated.

We model a camera network with two undirected graphs: a *communication graph* and a *vision graph*. We illustrate the idea in Figure 1 with a hypothetical network of ten cameras. Figure 1a shows a snapshot of the locations and orientations of the cameras. Figure 1b illustrates the communication graph for the network; an edge appears between two cameras in this graph if they have one-hop direct communication. This is a common abstraction in wireless ad-hoc networks (see [13] for a review). The communications graph is mostly determined by the locations of the nodes and the topography of the environment; in a wireless setting, the instantaneous power each node can expend towards communication is also a factor.

Figure 1c illustrates the *vision graph* for the network; an edge appears between two cameras in this graph if they observe some of the same scene points from different perspectives. We note that the presence of an edge in the communication graph does not imply the presence of the same edge in the vision graph,

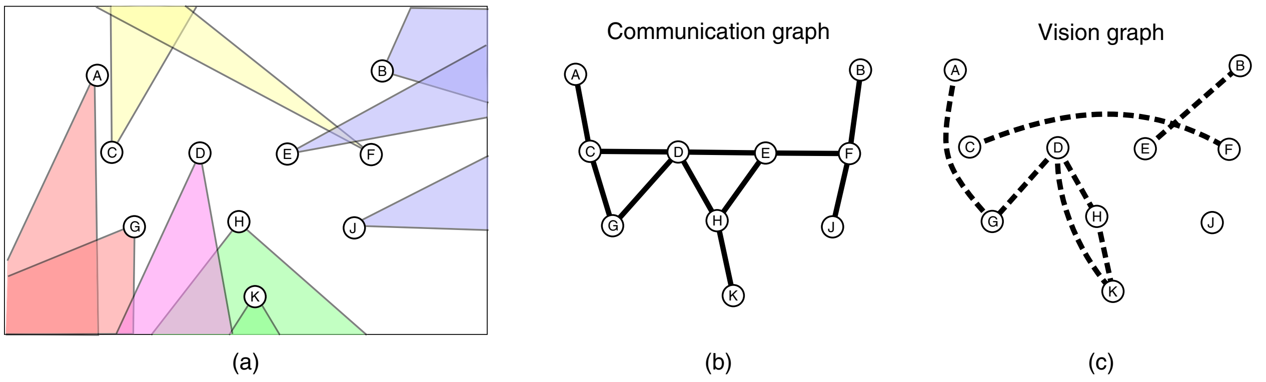

Fig. 1. (a) A snapshot of the instantaneous state of a camera network, indicating the fields of view of ten cameras. (b) The associated communication graph. (c) The associated vision graph. Note the presence of an edge in one graph does not imply the presence of the same edge in the other graph.

since the cameras may be pointed in different directions (for example, cameras $A$ and $C$). Conversely, an edge can connect two cameras in the vision graph despite a lack of physical proximity between them (for example, cameras $C$ and $F$). Figure 2 illustrates the communication and vision graphs for a simulated example in which 30 cameras are scattered around several buildings. We can see that the communication and vision graphs are not highly correlated.

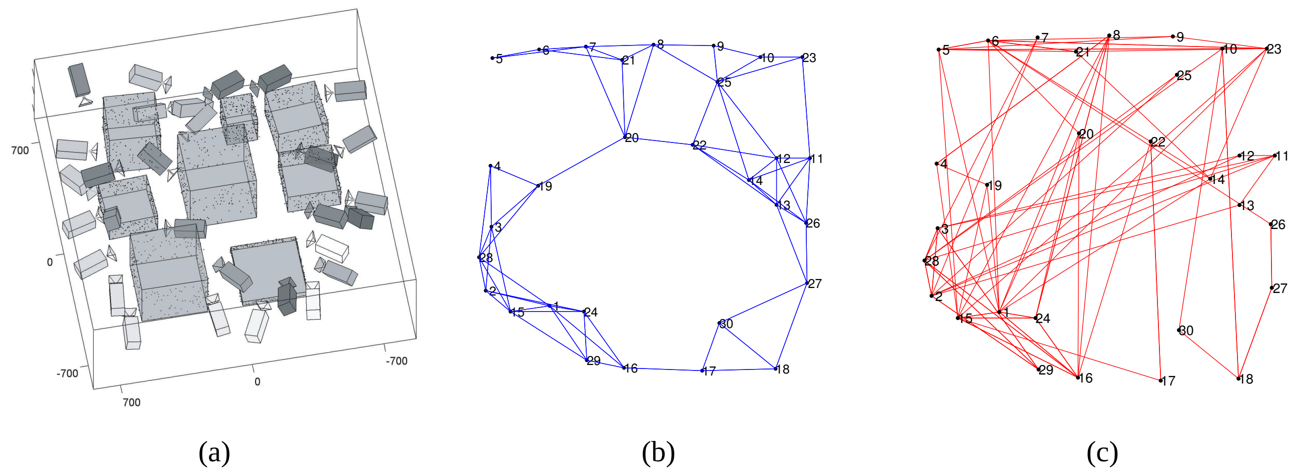

Fig. 2. (a) A simulated camera network (the focal lengths of the cameras have been exaggerated). (b) The corresponding communication graph, assuming each camera has the same fixed antenna range. (c) The corresponding vision graph.

We note that our research focus is at the application layer of a camera network architecture, as opposed to the physical layer comprised of the cameras themselves or the communication layer that defines networking protocols. Considerations of these layers are addressed at length in other articles in this special issue. An increasing number of options are available to implement the physical layer of a camera network, including the Cyclops [14], CMUCam [15], Panoptes [16], or Mesheye [17] systems. Small embedded sensor network platforms such as Crossbow motes [18] are frequently augmented with cameras to create visual sensor networks.

## III. Estimating the Vision Graph

Antone and Teller [19] used a camera adjacency graph similar to our vision graph to calibrate hundreds of still omnidirectional cameras in the MIT City project. However, this adjacency graph was obtained from *a priori* knowledge of the cameras' rough locations acquired by a GPS sensor, instead of estimated from the images themselves. Similarly, Sharp et al. [20] addressed how to distribute errors in estimates of camera calibration parameters with respect to a vision graph, but this graph was manually constructed.

Kulkarni et al. [21] proposed a method in which a reference object is moved to many positions around the cameras' environment. The detected presence of the object in different views at the same time instant provides an approximate initialization for the cameras' regions of overlap.

Many researchers have approached the vision graph estimation problem using the spatio-temporal paths of tracked objects that enter and exit the field of view of each camera [22], [23]. In some cases, this is referred to as estimating the topology of the network. Special cases of such objects include identifiable people [24], pedestrians [25] or vehicles [26]. In some cases, the objects are assumed to move on a ground plane that is known [27], [28] or estimated from object tracks [29], [30]. Van den Hengel et al. [31] took a slightly different approach in which the vision graph was initially assumed to be fully connected, and edges were removed that are contradicted by observed evidence over time. Most of these approaches either assume that the cameras' fields of view are non-overlapping or that a common dominant plane exists in the scene (Meingast et al. [32] is one exception). Neither scenario is our primary interest in this paper, since in the first case, the cameras' positions and orientations are difficult to estimate accurately from the images alone, and in the second case, it is more appropriate to relate the cameras by image homographies rather than full 3D calibration.

Graph relationships on image sequences are frequently encountered in image mosaicking applications, e.g., [33], [34], [35]. However, in such cases, adjacent images can be assumed to have connecting edges, since they are closely-sampled frames of a smooth camera motion. Furthermore, a chain of homographies can usually be constructed that gives reasonable initial estimates for where other graph edges occur. The problem considered here is substantially more complicated, since a camera network generally contains a set of unordered images taken from different viewpoints. The images used to localize the network may even be acquired at different times, since we envision that a wireless camera network would be realistically deployed in a time-staggered fashion (e.g., by soldiers advancing through territory or an autonomous unmanned vehicle dropping camera nodes from the air), and that new nodes will occasionally be deployed to replace failing ones.

Brown and colleagues [36], [37] addressed multi-image matching for the problem of constructing mosaics from an unordered set of images, though the vision graph is not explicitly constructed in either case. Also in the unordered case, Schaffalitzky and Zisserman [38] used a greedy algorithm to build a spanning tree (i.e., a partial vision graph) on a set of images, assuming the multi-image correspondences were available at a single processor.

An alternate method for distributed feature matching (which implicitly defines a vision graph) was described by Avidan et al. [39], who used a probabilistic argument based on random graphs to analyze the propagation of wide-baseline stereo matching results obtained for a small number of image pairs to the remaining cameras. The results were based on simulated data and not applied to the camera calibration problem.

In the rest of this section, we summarize our basic approach to estimating the vision graph, which is more fully described in [7]. First, each camera detects a set of distinctive feature points in its image that are likely to match other images of the same scene. Both the number of features and the length of each feature descriptor are substantially reduced to form a fixed-length "feature digest" that the camera broadcasts to the rest of the network. Each receiver camera decompresses the feature digest to recover the approximate feature descriptors, which are matched with its own features to generate putative matches. If enough matches are found, a vision graph edge is established.

## A. *Feature Detection and Description*

Our first step in estimating the vision graph is the detection of high-quality features at each camera node, i.e. regions of pixels representing scene points that can be reliably, unambiguously matched in other images of the same scene.

Several researchers have obtained point correspondences for topology and calibration estimation using modulated light sources moved throughout a darkened room [40], [41], [42] or placed on the cameras themselves [43]. Maas [44] used a moving reference bar of known length. It is common to acquire matching feature points using images of one or more planar targets containing parallel lines, a "checkerboard" image, or uniquely identifiable tags [45], [46], [47]. However, for large-scale outdoor camera networks, we believe it is important to develop techniques in which neither the cameras nor the scene are altered from their natural state, with no assumptions about camera synchronization or interactions between the user and the environment.

A recent focus in the computer vision community has been on different types of “invariant” detectors that select image regions that can be robustly matched even between images where the camera perspectives or zooms are quite different. Our approach is based on the popular and successful Scale-Invariant Feature Transform (SIFT) detector/descriptor proposed by Lowe [48]. Mikolajczyk and Schmid [49] showed that this combination outperformed most other detector/descriptor combinations in their experiments; for a broad survey of other modern feature detectors, see [50].

The basic idea of the SIFT detector is to process the image at multiple scales with a difference-of-Gaussian filter, and return scale-space extrema of the result. Qualitatively, this results in the detection of “blobs” that are distinguishable from their background at different scales. The number of features detected by the sending camera, which we denote $N$, is determined by the number of scale-space extrema of the image and user-specified thresholds to eliminate feature points that have low contrast or too closely resemble a linear edge (see [48] for more details). For a typical image, $N$ is on the order of hundreds or thousands.

Once feature locations and regions of support have been determined, each region must be described with a finite number of scalar values– this set of numbers is called the descriptor for the feature. The simplest descriptor is just a set of image pixel intensities; however, the intensity values alone are unlikely to be robust to scale or viewpoint changes. The SIFT feature descriptor that we use is a histogram of gradient orientations designed to be invariant to scale and rotation of the feature. Typically, the algorithm takes a $16 \times 16$ grid of samples from the the gradient map at the feature’s scale, and uses it to form a $4 \times 4$ aggregate gradient matrix. Each element of the matrix is quantized into 8 orientations, producing a descriptor of dimension 128.

## *B. Feature Digest Construction*

The next step is to select a subset containing $M$ of the $N$ features for the feature digest, such that the selected features are both highly distinctive and spatially well-distributed across the image (in order to maximize the probability of a match with an overlapping image). We measure feature distinctiveness using a strength measure based on the eigenvalues of the local gradient matrix at each feature [37].

If the digest is to contain $M$ features, we could just send the $M$ strongest features. However, in practice, there may be clusters of strong features in small regions of the image that have similar textures, and would unfairly dominate the feature list. Therefore, we need a way to distribute the features more fairly across the image. In our approach, we use a 2-dimensional k-d tree [51] containing $c$ cells constructed from the image coordinates of feature points. For each nonterminal node, we partition the node's data along the dimension that has larger variance. Finally, we select the $\lfloor \frac{M}{c} \rfloor$ strongest features from each k-d cell to add to the feature digest. An alternate approach would be to use the adaptive non-maximal suppression approach described by Brown et al. [37].

Once the $M$ features have been selected, we compress them so that each is represented with $K$ parameters (instead of the usual 128-dimensional SIFT descriptor). We do so by projecting each feature descriptor onto the top $K$ principal component vectors computed over the descriptors of the $N$ original features. Specifically, the feature digest is given by $\{\bar{\mathbf{v}}, \mathbf{Q}, \mathbf{p}_1, \ldots, \mathbf{p}_M, (x_1, y_1), \ldots, (x_M, y_M)\}$, where $\bar{\mathbf{v}} \in \mathbb{R}^{128}$ is the mean of the $N$ SIFT descriptors, $\mathbf{Q}$ is the $128 \times K$ matrix of principal component vectors, $\mathbf{p}_j = \mathbf{Q}^T(\mathbf{v}_j - \bar{\mathbf{v}}) \in \mathbb{R}^K$, where $\mathbf{v}_j$ is the $j$th selected feature's SIFT descriptor $\in \mathbb{R}^{128}$, and $(x_j, y_j)$ are the image coordinates of the $j$th selected feature. Thus, the explicit relationship between the feature digest length $L$, the number of features $M$, and the number of principal components $K$ is

$$L = b(128(K+1) + M(K+2)), \tag{1}$$

where $b$ is the number of bytes used to represent a real number. Therefore, for a fixed $L$, there is a tradeoff between sending many features (thus increasing the chance of matches with overlapping images) and coding the feature descriptors accurately (thus reducing false or missed matches). These tradeoffs are analyzed in detail in [7]. We note that our concept of a feature digest seems related to Jannotti and Mao's concept of a "geographic hash table" [52].

## *C. Feature Matching and Vision Graph Edge Formation*

When the sending camera's feature digest is received at a given camera node, the goal is to determine whether a vision graph edge is present. In particular, for each sender/receiver image pair where it exists, we want to obtain a stable, robust estimate of the epipolar geometry [53] based on the sender's feature digest and the receiver's complete feature list. We also obtain the correspondences between the sender and receiver that are consistent with the epipolar geometry, which are used to provide evidence for a vision graph edge.

Based on the sender's message, each receiving node generates an approximate descriptor for each incoming feature as $\widehat{\mathbf{v}}_j = \mathbf{Q}\mathbf{p}_j + \bar{\mathbf{v}}$. If we denote the receiving node's features by SIFT descriptors $\{\mathbf{r}_i\}$, then we compute the nearest $(\mathbf{r}_j^1)$ and the second nearest $(\mathbf{r}_j^2)$ receiver features to feature $\widehat{\mathbf{v}}_j$ based on the Euclidean distance between SIFT descriptors in $\mathbb{R}^{128}$. Denoting these distances $d_j^1$ and $d_j^2$ respectively,

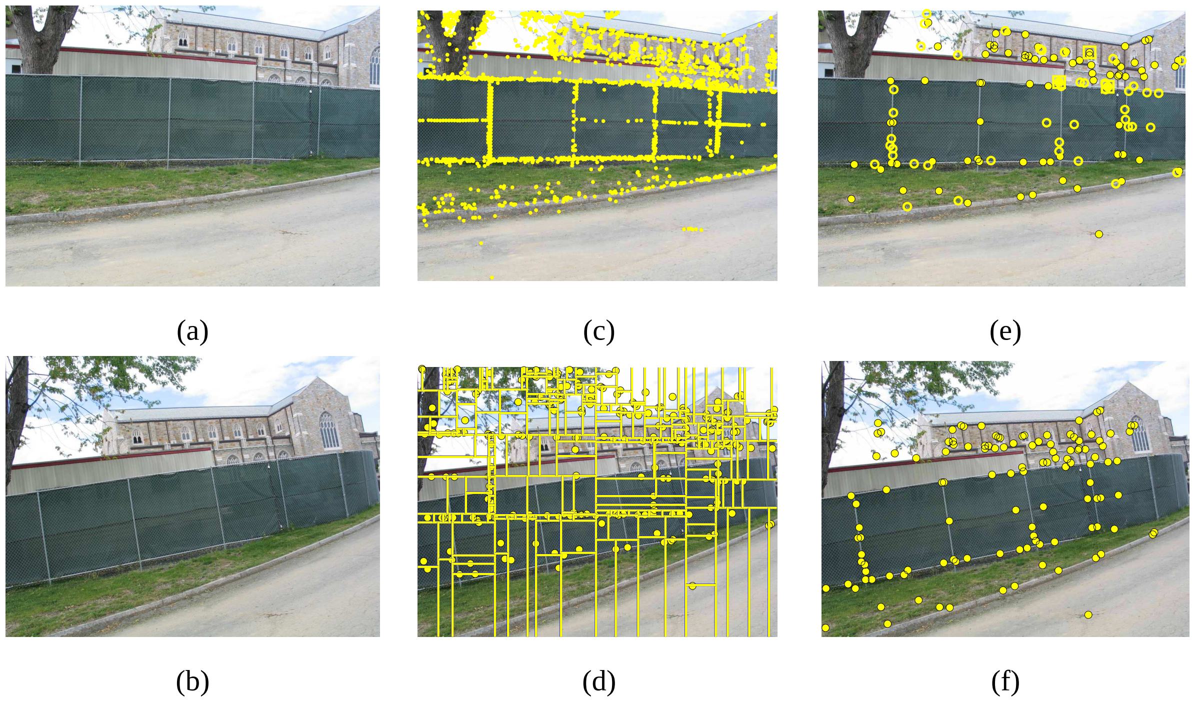

Fig. 3. Example results of image matching from a pair of images. (a) Image 1. (b) Image 2. (c) The 1976 detected features in Image 1. (d) The k-d tree and corresponding 256-feature digest in image 1. (e) The dots indicate 78 features in Image 1 detected as correspondences in Image 2, using the minimal Euclidean distance between SIFT descriptors and the ratio criterion with a threshold of 0.6. The 3 squares indicate outlier features that were rejected. The circles indicate 45 new correspondences that were grown based on the epipolar geometry, for a total of 120 correspondences. (f) The positions of the 120 corresponding features in Image 2. (Figure from [7].)

we accept $(\widehat{\mathbf{v}}_j, \mathbf{r}_j^1)$ as a match if $d_j^1/d_j^2$ is below a certain threshold. The rationale, as described by Lowe [48], is to reject features that may ambiguously match several regions in the receiving image (in this case, the ratio $d_j^1/d_j^2$ would be close to 1). However, it is possible that this process may reject correctly matched features or include false matches (also known as outliers). To combat the outlier problem, we robustly estimate the epipolar geometry, and reject features that are inconsistent with it [54]. To make sure we find as many matches as we can, we add feature matches that are consistent with the epipolar geometry and for which the ratio $d_j^1/d_j^2$ is suitably low. This process is illustrated in Figure 3.

Based on the grown matches, we establish a vision graph edge if the number of final feature matches exceeds a threshold $\tau$, since it is highly unlikely that a large number of good matches consistent with the epipolar geometry occur by chance. We model the establishment of vision graph edges as a typical detection problem [55], and analyze the performance at a given parameter combination as a point on a Receiver-Operating-Characteristics (ROC) curve. This curve plots the probability of detection $p_d$ (i.e.,

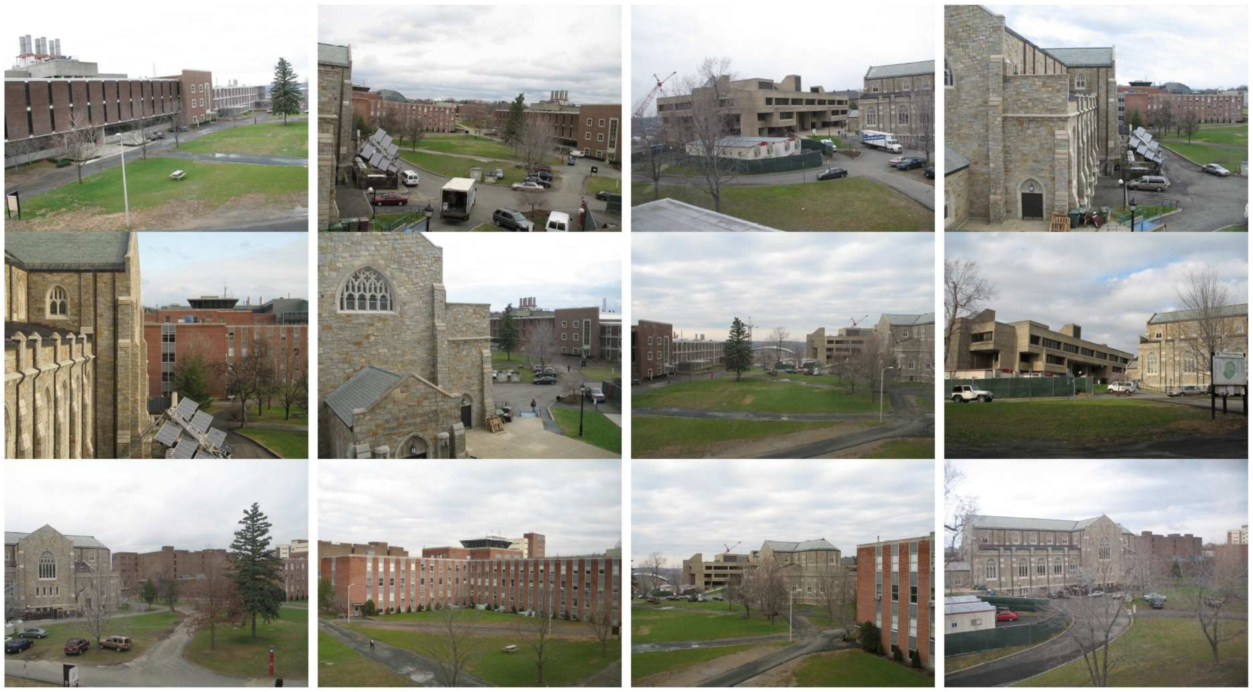

Fig. 4. Sample images from the 60-image test set gathered on the RPI campus.

the algorithm finds an edge while there is actually an edge) against the probability of false alarm $p_{fa}$ (i.e., the algorithm finds an edge while the two images actually have little or no overlap). The user can select an appropriate point on the ROC curve based on application requirements on the performance of the predictor.

We simulated an outdoor camera network using a set of 60 widely-separated images taken on the RPI campus, acquired with a Canon PowerShot G5 digital camera in autofocus mode (so that the focal length for each camera is different and unknown) and an image resolution of 1600 $\times$ 1200. Figure 4 shows some example images from the test set. Figure 5 illustrates the ROC curves resulting from the vision graph generation algorithm using fixed message sizes of length $L =$80, 100, and 120 kilobytes. We can see that for all message lengths, the algorithm has good performance, since high probabilities of detection can be achieved with low probabilities of false alarm (e.g. $p_d \geq 0.8$ when $p_{fa} = 0.05$). As the message length increases, the detector performances become more similar (since the message length is not as limiting), and detection probability approaches that which can be achieved by sending all features with no compression at all (the upper line in Figure 5). Once the vision graph is established, we can use feedback in the network with less message compression to refine edge decisions, moving closer to the ideal ROC curve.

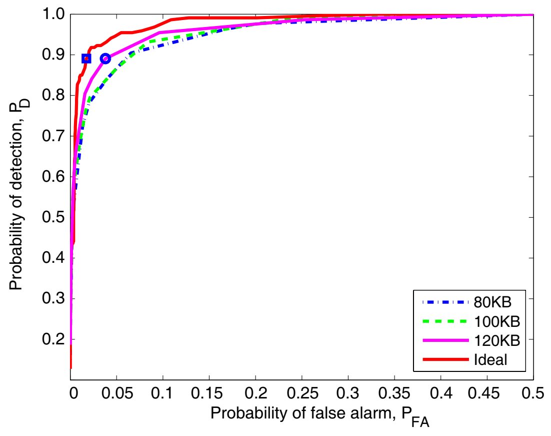

Fig. 5. Best achievable ROC curves for message lengths 80KB, 100KB and 120KB. The “ideal” curve is generated by applying our algorithm using all features from each image and no compression. Once the vision graph is established, false edges can be removed based on further messages (e.g., moving from the circular to the square operating point). (Figure from [7].)

## IV. Calibrating the Camera Network

Once the vision graph is established, the next step is to obtain an estimate of the calibration parameters for all cameras. A substantial body of work exists on the distributed localization of nodes in wireless sensor networks based on considerations such as the time-of-flight between nodes (see, e.g., [56]). Our interest is in calibration techniques for a visual sensor network that use the cameras’ images as the basis for estimation.

The classical problem of externally calibrating a pair of cameras is well-understood [57]; the parameter estimation usually requires a set of feature point correspondences in both images (we described one approach to obtaining a set of such correspondences in Section III). When no points with known 3-D locations in the world coordinate frame are available, the cameras can be calibrated up to a similarity transformation [54]. That is, the cameras’ positions and orientations can be accurately estimated relative to each other, but not in an absolute sense. Without metric information about the scene, an unknown scale parameter also remains; for example, the same set of images would be produced by cameras twice as far away from a scene that is twice as large.

Multi-camera calibration can be accomplished by minimizing a nonlinear cost function of the calibration parameters and a collection of unknown 3-D scene points projecting to matched image correspondences; this problem is also known as *Structure from Motion (SFM)*. The optimization process for estimating the parameters is called *bundle adjustment* [58]. Good results are achievable when the images and correspondences are all accessible to a powerful, central processor. It is beyond the scope of this paper to survey the centralized SFM problem here, but the book by Hartley and Zisserman [54] is an excellent reference. Also, Ma et al. [59] outlined a complete step-by-step recipe for the SFM problem, assuming the information from all cameras is collected in one place. We note that SFM is closely related to a problem in the robotics community called *Simultaneous Localization and Mapping (SLAM)* [60], [61], in which mobile robots must estimate their locations from sensor data as they move through a scene. SFM also forms the fundamental core of commercial software packages such as Boujou or SynthEyes for the problems of "matchmoving" or camera tracking, which are used to insert digital effects into Hollywood movies.

However, the unsupervised calibration of a distributed camera network is a new and challenging problem. There is no centralized collection of instantaneous images from each camera, and no user to help select feature points in corresponding images. Communication between cameras is expensive and implemented on a hop-by-hop basis, so it is impractical for each camera to obtain detailed knowledge about the images available throughout the entire network. On the other hand, to be useful, the results of a distributed algorithm must be comparable to the results that can be obtained with a centralized algorithm. Relatively few algorithms have been proposed for fully distributed 3D calibration of a large-scale camera network. We do not consider algorithms in which each camera independently calibrates itself based on images of a placed reference device with known world coordinates (e.g., [62]) as fundamentally distributed.

Several researchers have addressed camera network calibration based on an object that moves through a planar environment, which is less general than the 3D camera calibration problem addressed here (e.g., [63]). An excellent algorithm of this type called SLAT (Simultaneous Location and Tracking) was proposed by Funiak et al. [64]. The key innovations were a relative overparameterization of the cameras that allows Gaussian densities to be used, and a linearization procedure to address the uncertainty in camera angle for proper use of a Kalman filter. It would be very interesting to see how such an approach would scale to full calibration of large outdoor camera networks with many moving objects.

Many distributed algorithms for full 3D camera network calibration rely on using image correspondences to estimate the epipolar geometry between image pairs. When the intrinsic camera parameters are known, this information can be used to extract estimates of the rotation and translation between each camera pair. For example, Barton-Sweeney et al. [42] used an Extended Kalman Filtering framework on the estimated epipolar geometry to estimate the rotation and translations between a camera pair. Mantzel et al. [65] proposed an algorithm called DALT (Distributed Alternating Localization and Triangulation) for camera network calibration. As the the title suggests, each camera calibrates itself by alternating between 1) triangulating 2D image projections into 3D space, assuming the camera matrices are known, and 2) estimating the camera matrices based on putative image-to-world correspondences. While their optimization approach is simpler than bundle adjustment, this work includes some interesting analysis of the computational and power requirements that would be incurred by the algorithm on a realistic platform. We finally note that these algorithms use a user-inserted calibration object as the basis for establishing point correspondences, instead of extracting such correspondences from images of an unstructured environment as described in the previous section.

In the rest of this section, we summarize our basic approach to the distributed calibration of camera networks, which is more fully described in [8], [9], [66]. Our goal is to design a calibration system in which each camera only communicates with (and possesses knowledge about) those cameras connected to it by an edge in the vision graph. The result will be that each camera has an estimate of 1) its own location, orientation, and focal length, 2) the corresponding parameters for each of its neighbors in the vision graph, and 3) the 3D positions of the image feature points it has in common with its neighbors. It is important to obtain the reconstruction in a *metric* framework, where the recovered geometry of the cameras/scene differs from the truth only by an unknown rotation, translation, and scale.

## A. *Initializing the Calibration*

The first step our algorithm is the formation of clusters. Each node independently forms a cluster $\mathcal{C}_i$ depending on the number of correspondences detected along each vision graph edge. (We assume these correspondences were formed during the process of estimating the vision graph, and retained for the purposes of calibration.) Initially, this cluster is formed as $\mathcal{C}_i = \{j \mid j \in Nb(i)\}$, where $Nb(i)$ denotes the set of nodes neighboring $i$ in the vision graph. However, nodes that share only a few corresponding points with node $i$ are removed from the cluster in order to ensure a minimum *nucleus* of corresponding points seen by all cameras in the cluster. The minimum cluster and nucleus size required for viable calibration depend on the number of parameters to be estimated at each node [67], [68]. In our experiments, we use a minimum cluster size of 3 cameras and a nucleus of at least 8 corresponding points.

Next, the parameters of the cluster are estimated through an iterative minimization process called bundle adjustment [58]. We denote $\{P_1, \ldots, P_m\}$ as the cameras in $i$'s cluster, where $m = |\{i, \mathcal{C}_i\}|$. Similarly, we denote $\{X_1, \ldots, X_n\}$ as the 3D points that are seen by at least 3 cameras in the cluster. We define a binary indicator function $\chi_{jk}$ such that $\chi_{jk} = 1$ if point $k$ is observed by camera $j$ and 0 otherwise.

Node $i$ must now solve the structure-from-motion problem, i.e., estimate the camera parameters $P$ as well as the unknown scene points $X$ using only the 2D image correspondences $\{u_{jk}, j = 1, \ldots, m, k = 1, \ldots, n \mid \chi_{jk} = 1\}$. If $\hat{u}_{jk}$ represents the projection of $\hat{X}_k^i$ onto $\hat{P}_j^i$ for some estimate $(\hat{P}^i, \hat{X}^i)$, then the bundle adjustment cost function that is minimized at each cluster $i$ is given by

$$\min_{\substack{\{\hat{P}_j^i\}, j \in \{i, \mathcal{C}_i\} \\ \{\hat{X}_k^i\}, k=1,\ldots,n}} \sum_j \sum_k \chi_{jk} (\hat{u}_{jk} - u_{jk})^T \Sigma_{jk}^{-1} (\hat{u}_{jk} - u_{jk}) \tag{2}$$

where $\Sigma_{jk}$ is the $2 \times 2$ covariance matrix associated with the noise in the measurement $u_{jk}$. The quantity inside the sum is called the Mahalanobis distance between $\hat{u}_{jk}$ and $u_{jk}$. The minimization is taken over the 3D points seen by at least three cameras, as well as the focal lengths, rotation matrix parameters and the translation vectors of all cameras in the cluster. The optimization is solved using a sparsity-exploiting Levenberg-Marquardt algorithm, as described by Hartley and Zisserman [54].

As with any nonlinear minimization algorithm, the critical issue is obtaining a "good" initialization point such that the optimization converges to a desirable local minimum. For the calibration problem, we obtain this initial point in three steps.

First, we find the set of $n'$ points seen by all the cameras in the cluster (which we call the nucleus) and form a $3m \times n'$ measurement matrix $W$ created from the $u_{jk}$'s in the nucleus, represented in homogenous coordinates. The goal is to factorize this matrix into a product of the camera matrices and the homogeneous 3D scene points via the relationship

$$W = \begin{pmatrix} \lambda_{11}u_{11} & \lambda_{12}u_{12} & \cdots & \lambda_{1n}u_{1n'} \\ \lambda_{21}u_{21} & \lambda_{22}u_{22} & \cdots & \lambda_{2n}u_{2n'} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{m1}u_{m1} & \lambda_{m2}u_{m2} & \cdots & \lambda_{mn}u_{mn'} \end{pmatrix} = \begin{pmatrix} P_1 \\ P_2 \\ \vdots \\ P_m \end{pmatrix} \begin{pmatrix} X_1 & X_2 & \cdots & X_{n'} \end{pmatrix}. \tag{3}$$

By (3), the measurement matrix is ideally of rank four, but the factors $\lambda_{ij}$ (called the projective depths) are unknown. To accomplish the factorization, we use the algorithm proposed by Sturm and Triggs [69], [70] that alternates between updating the projective depths and applying the singular value decomposition to find the best rank-4 approximation of the measurement matrix.

After this step, the cameras are recovered up to a projective transformation. This means that while some geometric properties of the reconstructed configuration will be correct compared to the truth (e.g., the order of 3D points lying along a straight line), others will not (e.g., the angles between lines/planes or the relative lengths of line segments). The mathematical cause of this ambiguity is that we can substitute all $P_j$ with $P_jH^{-1}$ and all $X_k$ with $HX_k$ for any $4 \times 4$ nonsingular matrix $H$ and get the same set of $u_{jk}$.

In order to make the reconstruction useful (i.e., to recover the correct camera configuration up to an unknown rotation, translation, and scale), we need to estimate the matrix $H$ that turns the projective factorization into a metric factorization. There are several methods for this process (sometimes called auto-calibration); all depend heavily on projective geometry too complex to describe here. The approach we favor is based on estimating a $4 \times 4$ symmetric rank-3 matrix called the absolute dual quadric, which is constrained by the camera matrices and some known properties of each camera's fixed internal parameters (for example, that the elements of the CCD array are square). Detailed descriptions of metric-from-perspective recovery based on this approach are given by Pollefeys et al. [67], [68].

Finally, 3D scene points not in the nucleus that are seen by at least two cameras can be triangulated [71], [72] and incorporated into the bundle adjustment cost function (2), which is then minimized from the initial point obtained from factorization.

We reported detailed results of testing this initial calibration algorithm on synthetic camera/scene configurations in [8], showing that the calibration accuracy was quite good and declined gracefully with additional measurement noise. We also analyzed the algorithm's cost in terms of the numbers of messages each camera would transmit/receive compared to those of a centralized algorithm where one node acts as a "master". We found that the distributed algorithm makes fairer use of the underlying communication links, and reduces the maximum number of messages per node/edge, which would be a a particularly important issue for the longevity of battery-operated or otherwise power-constrained wireless sensor networks [73].

Here, we illustrate the performance of the initialization algorithm on the experimental testbed of real campus images described in the previous section, focusing on 15 images of a particular church-like building in the scene. Figure 6a shows an example image of this building with some typical feature points (i.e., the $u_{jk}$) overlaid. Figure 6b shows an overhead view of the reconstructed 3D scene and camera configuration obtained from applying the distributed calibration algorithm, aligning each camera's reconstructed scene points to the same frame and displaying the average position/orientation of each camera. No single camera would have full knowledge about the entire scene as shown, and each camera really only knows its location relative to its neighbors and reconstructed scene points. While the ground truth for the configuration is unknown, the quality of the structure recovery is apparent; for example, the right angles of the building faces are clearly well-estimated. Furthermore, the Euclidean reprojection error, obtained by averaging the values of $\|\hat{u}_{jk} - u_{jk}\|$ for every camera/point combination, was computed as 0.59 pixels, meaning the reprojections are accurate to within less than a pixel. This compares favorably with the reprojection error of 0.34 pixels that was obtained from a centralized bundle adjustment computation.

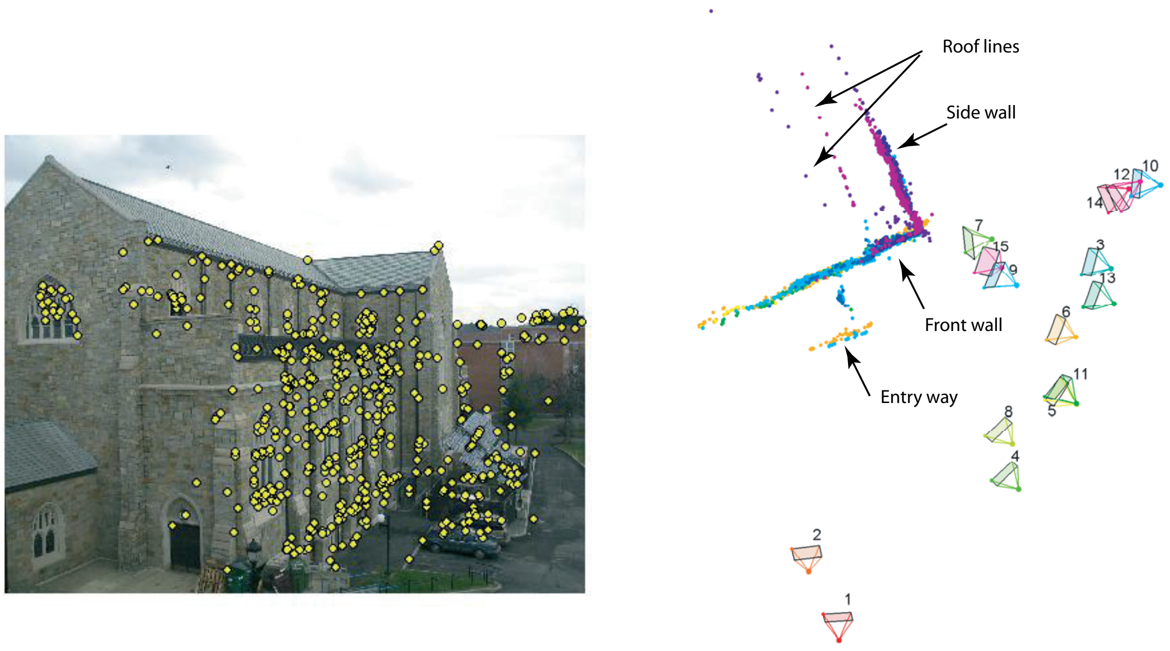

Fig. 6. (a) Original image, with detected feature points overlaid. (b) Top view of the reconstructed 3D scene and camera configuration for the real experiment. The color of each scene point indicates to which of the clusters it belongs, illustrating that points from different clusters are correctly merged in the final scene. (Figure (b) from [8].)

## *B. Refining the Calibration*

Even when the initial local calibrations are reasonably accurate, the estimates of the same parameter at different nodes will generally be inconsistent. That is, estimates of the same camera parameters by different neighbors will vary slightly due to the different underlying data used for the neighbors' local computations. A simple approach to obtaining consistency would be to collect and average the inconsistent estimates of each parameter and redistribute the result. However, this is only statistically optimal when the joint covariances of all the camera parameter estimates are identical, which is never the case, and in practice performs poorly [66]. In this section, we show how the camera parameter estimates can be made more globally consistent using a probabilistic framework that combines the parameter covariance matrices properly.

Many researchers have proposed methods for dealing with inconsistent estimates in wireless sensor networks. These are typically based on verifying whether estimates of the same parameter are sufficiently close [74], [75], [76] and/or consistent with prior knowledge [77]. Most such approaches are well-suited for sensors that measure a scalar quantity such as temperature or pressure. In contrast, in camera networks we face inconsistencies in a continuous, high-dimensional parameter space, and require principled statistical methods for resolving them.

We propose a method based on belief propagation [78], a method for probabilistic inference in networks that several researchers have recently applied to discrete or low-dimensional sensor networking and robotics problems (e.g., [79], [80], [81], [82]).

Let $Y_i$ represent the true state vector at node $i$ that collects the parameters of that node's camera matrix $P_i^i$ as well as those of its neighbors $P_j^i, j \in Nb(i)$, and let $Z_i$ be the noisy "observation" of $Y_i$ that comes from the local initial calibration process. That is, the observations arise out of local bundle adjustment on the image projections of common scene points $\{u_{jk}\}$ that are used as the basis for the initial calibration. Our goal is to estimate the true state vector $Y_i$ at each node given all the observations by calculating the marginal

$$p(Y_i|Z_1,\dots,Z_M) = \int_{\{Y_j, j\neq i\}} p(Y_1,\dots,Y_M|Z_1,\dots,Z_M)\ dY_j. \tag{4}$$

Recently, belief propagation has proven effective for marginalizing state variables based on local message-passing; we briefly describe the technique below. According to the Hammersley-Clifford theorem [83], [84], a joint density is factorizable if and only if it satisfies the pair-wise Markov property,

$$p(Y_1, Y_2, ...Y_M) \propto \prod_{i\in V} \phi_i(Y_i) \prod_{(i,j)\in E} \psi_{ij}(Y_i, Y_j), \tag{5}$$

where $\phi_i$ represents the belief (or evidence) potential at node $i$, and $\psi_{ij}$ is a compatibility potential relating each pair of nodes $(i,j) \in E$. Pearl [85] later proved that an inference on this factorized model is equivalent to a message-passing system, where each node updates its belief by obtaining information (or "messages") from its neighbors. This process is what is generally referred to as belief propagation. The marginalization is then achieved through the update equations

$$m_{ij}^t(Y_j) \propto \int_{Y_i} \psi(Y_i, Y_j)\phi(Y_i) \prod_{k\in Nb(i)\backslash j} m_{ki}^{t-1}(Y_i)\ dY_i \tag{6}$$

$$b_i^t(Y_i) \propto \phi(Y_i) \prod_{j\in Nb(i)} m_{ji}^t(Y_i), \tag{7}$$

where $m_{ij}^t$ is the message that node $i$ transmits to node $j$ at time $t$, and $b_i^t$ is the belief at node $i$ about its state, which is the approximation to the required marginal density $p(Y_i)$ at time $t$. This algorithm is also called the sum-product algorithm.

In our problem, the joint density in (4) can be expressed as

$$\begin{aligned} p\left(Y_1, Y_2, \ldots Y_M | Z_1, \ldots, Z_M\right) &\propto p\left(Y_1, Y_2, \ldots Y_M, Z_1, \ldots, Z_M\right) && (8)\\ &= \prod_{i \in V} p(Z_i|Y_i) \prod_{(i,j)\in E} p(Y_i, Y_j). && (9) \end{aligned}$$

Here, $Z_i$ is observed and hence the likelihood function $p(Z_i|Y_i)$ is a function of $Y_i$. Similar factorizations of the joint density are common in decoding systems [86].

$p(Y_i, Y_j)$ encapsulates the constraints between the variables $Y_i$ and $Y_j$. That is, the random vectors $Y_i$ and $Y_j$ may share some random variables that must agree. We enforce this constraint by defining binary selector matrices $C_{ij}$ based on the vision graph as follows. Let $M_{ij}$ be the number of variables that $Y_i$ and $Y_j$ have in common. Then $C_{ij}$ is a binary $M_{ij} \times |Y_i|$ matrix such that $C_{ij}Y_i$ selects these common variables. Then we assume

$$P(Y_i, Y_j) \propto \delta(C_{ij}Y_i - C_{ji}Y_j) \tag{10}$$

where $\delta(x)$ is 1 when all entries of $x$ are 0 and 0 otherwise. The joint density (10) makes the implicit assumption of a uniform prior over the true state variables; i.e. it only enforces that common parameters match. In our experiments, we found this formulation to result in equal or better performance than a softer constraint in which the compatibility potential is higher for more similar parameter estimates.

Therefore, we can see that (9) is in the desired form of (5), identifying

$$\begin{aligned} \phi_i(Y_i) &\propto p(Z_i|Y_i) && (11)\\ \psi_{ij}(Y_i, Y_j) &\propto \delta(C_{ij}Y_i - C_{ji}Y_j). && (12) \end{aligned}$$

Based on this factorization, it is possible to perform the belief propagation directly on the vision graph edges using the update equations (6) and (7). Figure 7 illustrates one step of the message passing procedure, indicating the actual camera parameters that are involved in each message. For example, in order to calibrate, Camera 1 must collect other nodes' estimates of its own parameters, as well as estimates of its neighbors' parameters (i.e., those of Cameras 2, 3, and 8). Each of its vision graph neighbors sends Camera 1 the subset of these parameters about which it has information in the form of a message $m_{j1}, j \in Nb(1)$.

In our problem, we modeled the likelihood densities $p(Z_i|Y_i)$ as Gaussian; in this case, (6) and (7) reduce to simply passing and updating the first two moments of each $Y_i$. Let $\mu_i$ represent the mean of $Y_i$ and $\Sigma_i$ the corresponding covariance matrix. Node $i$ receives estimates $\mu_i^j$ and $\Sigma_i^j$ from each of its

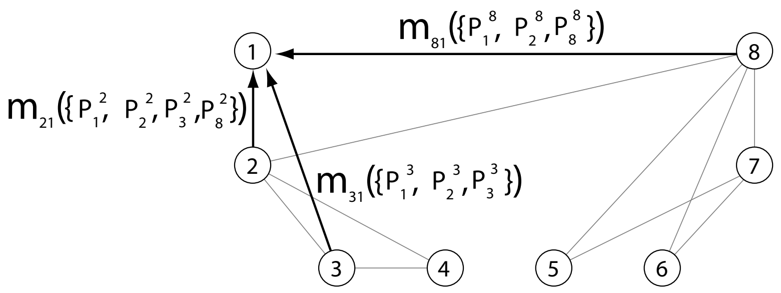
$m_{81}(\{P_1^8, P_2^8, P_8^8\})$
$m_{21}(\{P_1^2, P_2^2, P_3^2, P_8^2\})$
$m_{31}(\{P_1^3, P_2^3, P_3^3\})$
1
2
3
4
5
6
7
8

Fig. 7. An intermediate stage of message-passing. The $P_i^j$ indicate the camera parameters that are passed between nodes.

neighbors $j \in Nb(i)$. Then the update equations (6) and (7) reduce to minimizing the sum of the KL divergences between the updated Gaussian density and each incoming Gaussian density. Therefore, the belief update reduces to the well-known equations [87]:

$$\mu_i^i \leftarrow \left[\Sigma_i^{-1} + \sum_{j \in Nb(i)} (\Sigma_i^j)^{-1}\right]^{-1} \left(\Sigma_i^{-1}\mu_i + \sum_{j \in Nb(i)} (\Sigma_i^j)^{-1}\mu_i^j\right) \quad (13)$$

$$\Sigma_i^i \leftarrow \left[\Sigma_i^{-1} + \sum_{j \in Nb(i)} (\Sigma_i^j)^{-1}\right]^{-1}. \quad (14)$$

We note that (13)-(14) can be iteratively calculated in pairwise computations, instead of computed in batch, and that this pairwise fusion is invariant to the order in which the estimates arrive.

We obtain the mean and covariance of the assumed Gaussian density $p(Z_i|Y_i)$ based on forward covariance propagation from the bundle adjustment process. That is, the covariances of the noise in the image correspondences used for bundle adjustment are propagated through the bundle adjustment cost functional (2) to obtain a joint covariance matrix for the structure-from-motion parameters at each node [54]. Since we are predominantly interested in localizing the camera network, we marginalize out the reconstructed 3D structure to obtain covariances of the camera parameters alone. While the $\Sigma_i^j$ obtained by covariance propagation through the bundle adjustment cost functional are not usually diagonal (thus implying non-negligible dependencies among the parameters), we found that we could substantially improve the estimation accuracy, global consistency, and computational speed of the refinement algorithm by substituting each $\Sigma_i^j$ by its diagonal approximation.

The belief propagation framework as described above is generally applicable to many information fusion applications. However, when the beliefs represent distributed estimates of camera parameters, there are several additional issues that must be addressed, including:

1) *Minimal parameterizations.* Even if each camera matrix is parameterized minimally at node $i$ (i.e. 1

parameter for the focal length, 3 parameters for the camera center, 3 parameters for the rotation matrix), there are still 7 degrees of freedom corresponding to an unknown similarity transformation of all cameras in $Y_i$. Without modification, the covariance matrices in (13)-(14) have null spaces of dimension 7 and cannot be inverted. We address this issue by placing node $i$'s camera at the origin and fixing the scale of the cluster before each fusion.

2) *Frame alignment.* Since we assume there are no landmarks in the scene with known 3D positions, the camera motion parameters can be estimated only up to a similarity transformation, and this unknown similarity transformation will differ from node to node. The estimates $Y_i^i$ and $Y_i^j, j \in Nb(i)$ must be brought to a common coordinate system before every fusion step.

The details of how we address each of these issues can be found in [9].

We judge the algorithm's performance by evaluating the consistency of the estimated camera parameters throughout the network both before and after the belief propagation algorithm (these results were obtained using the full covariance matrices, not the diagonal approximation). For the 15-image dataset discussed in the previous section, we found that the standard deviation between multiple estimates of the same camera's center decreased by a factor of 6, that of the camera rotation matrix by a factor of 2, and that of the focal length by a factor of 1.5. When compared to the standard deviation of the centralized estimate, we found that the belief propagation standard deviations were larger by factors of 1.2 for the camera center parameters, 0.9 for the rotation matrix parameters, and 0.7 for the focal lengths, indicating that the message-passing algorithm approaches a result that is as certain as that of the centralized algorithm.

Figure 8 shows the multiple estimates of a subset of the cameras (aligned to the same coordinate frame) both before and after the calibration refinement algorithm. Before belief propagation, the estimates of each camera's position are somewhat spread out and there are several outliers (e.g., one estimate of camera 13 is far from the other two, and very close to the corner of the building). After belief propagation, the improvement in consistency is apparent; multiple estimates of the same camera are tightly clustered together. This improvement would clearly be important for good performance on higher-level vision tasks in a real camera network.

The computational cost of the distributed initialization and refinement algorithms are quite favorable compared to the centralized algorithm. The centralized algorithm suffers from needing to solve a very large (albeit sparse) problem with thousands of variables for which it is very difficult to obtain a good initial point. On the other hand, the distributed initialization algorithm is much faster (by a factor of 10-15 in our experiments) than the centralized algorithm since it works with smaller problems that are more easily initialized. While the distributed refinement algorithm is slower due to the large matrices involved,

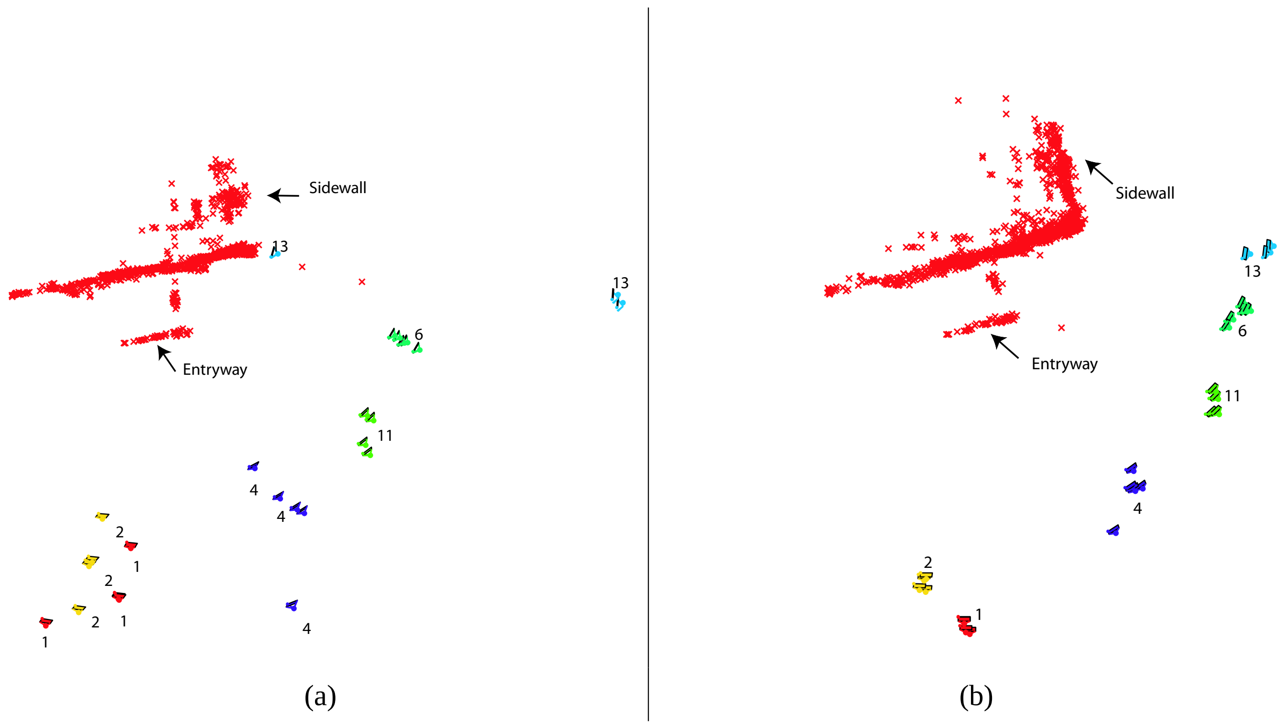

Fig. 8. Multiple camera estimates of a subset of cameras (a) before and (b) after the belief propagation refinement algorithm. (Figure from [9].)

the execution time is still moderately less than the centralized bundle adjustment (and is faster by a factor of 6 when the covariance matrices are approximated as diagonal). Furthermore, these estimates are based on our serial simulation of a distributed algorithm; in a real sensor network, first the initialization and then the refinement algorithm would be run in parallel on each node, resulting in substantial increases in actual execution speed.

## V. Discussion and Conclusions

We proposed a complete framework for the calibration of a distributed camera network, starting with the discovery of visual overlap between cameras and ending with an accurate, consistent estimate of all camera parameters in the network. The calibration process be viewed as a scalable, parallel algorithm that has complexity, memory, and networking advantages over a centralized calibration. The local nature of the calibration allows cameras that view part of the same scene to exchange and interpret visual information necessary for a higher-level task, such as handing off the tracking of a target that moves through the field of cameras, without requiring any single node to know the global configuration of the entire network.

The algorithms described here made the assumption that the camera nodes and vision graph were fixed. However, cameras in a real network might change position or orientation after deployment in response to either external events (e.g., wind or explosions) or remote directives from a command-and-control center. One simple way to extend our results to dynamic camera networks would be for each camera to broadcast any new features that appear in its image to the network in a short message, in order to keep the vision graph up-to-date. Similarly, the distributed calibration estimates could be kept up-to-date by occasional iterations of the message passing algorithm described in Section IV-B. Overall, the transient messaging load on the network and the priority of the calibration process at each processor would be proportional to the magnitude of the camera dynamics. It would be interesting to formalize this intuition of a continuous, adaptive, and efficient calibration background process in an actual dynamic camera network with power-constrained sensor nodes. We also note that if the cameras can be actively actuated to change their orientation and zoom, the calibration problem can be made easier [88].

For distributed wireless camera networks deployed in the real world, it will be important to consider calibration and other computer vision tasks in the context of the many other tasks the node processors and network need to undertake. This will entail a tight coupling between the vision algorithms and the MAC, network, and link-layer protocols, organization, and channel conditions of the network, as well as the power supply, transmitter/receiver and kernel scheduler of each node. This tight integration would be critical for making the algorithms described here viable for embedded platforms, such as networks of cell-phone cameras [89]. The ROC curves in Section III address this issue to some extent for the vision graph estimation problem (i.e., suggesting what combination of descriptor length and number of descriptors should be chosen to achieve the best performance given a fixed length for the feature digest). It would be an interesting and challenging problem to adaptively optimize a trade-off between the calibration performance at the application layer with the power consumption at the communication/physical layers.

We finally note that several researchers have recently applied more sophisticated and robust distributed inference algorithms than belief propagation to sensor fusion problems; the work of Paskin, Guestrin, and McFadden [90], [91] and Dellaert et al. [92] is notable. Future distributed camera calibration techniques could benefit from these new algorithms.

To conclude, while centralized computer vision algorithms for determining visual overlap and estimating camera calibration parameters are quite mature, and physical and communications-layer issues for visual sensor networks have made great progress in recent years, a substantial amount of research remains to be done in integrating the application, communication, and physical layers of a visual sensor network to produce powerful, self-organizing wireless camera nodes. Such research requires teaming of experts in computer vision, embedded systems, networking, and sensor design, and is beginning to take place in new conference venues aimed at bringing these groups together.

## References

[1] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," *Computer Networks*, vol. 38, pp. 393–422, 2002.

[2] L. Davis, E. Borovikov, R. Cutler, D. Harwood, and T. Horprasert, "Multi-perspective analysis of human action," in *Proceedings of the Third International Workshop on Cooperative Distributed Vision*, November 1999.

[3] T. Kanade, P. Rander, and P. Narayanan, "Virtualized reality: Constructing virtual worlds from real scenes," *IEEE Multimedia, Immersive Telepresence*, vol. 4, no. 1, pp. 34–47, January 1997.

[4] S. Moezzi, L.-C. Tai, and P. Gerard, "Virtual view generation for 3D digital video," *IEEE Multimedia*, vol. 4, no. 1, pp. 18–26, Jan.-March 1997.

[5] A. Savvides, C. C. Han, and M. B. Srivastava, "Dynamic fine-grained localization in ad-hoc wireless sensor networks," in *International Conference on Mobile Computing and Networking (MobiCom) 2001*, July 2001.

[6] D. Estrin *et al.*, *Embedded, Everywhere: A Research Agenda for Networked Systems of Embedded Computers*. National Academy Press, 2001.

[7] Z. Cheng, D. Devarajan, and R. Radke, "Determining vision graphs for distributed camera networks using feature digests," *EURASIP Journal of Applied Signal Processing, Special Issue on Visual Sensor Networks*, 2007, article ID 57034.

[8] D. Devarajan and R. Radke, "Calibrating distributed camera networks using belief propagation," *EURASIP Journal of Applied Signal Processing, Special Issue on Visual Sensor Networks*, 2007, article ID 60696.

[9] D. Devarajan, R. Radke, and H. Chung, "Distributed metric calibration of ad-hoc camera networks," *ACM Transactions on Sensor Networks*, vol. 2, no. 3, pp. 380–403, August 2006.

[10] R. T. Collins and Y. Tsin, "Calibration of an outdoor active camera system," in *IEEE Computer Vision and Pattern Recognition 1999*, 1999.

[11] R. Hartley, "Self-calibration from multiple views with a rotating camera," in *Proc. ECCV '94*, vol. 1, 1994, pp. 471–478.

[12] G. P. Stein, "Accurate internal camera calibration using rotation, with analysis of sources of error," in *ICCV*, 1995, pp. 230–236.

[13] Z. Haas *et al.*, "Wireless ad hoc networks," in *Encyclopedia of Telecommunications*, J. Proakis, Ed. John Wiley, 2002.

[14] M. Rahimi, R. Baer, O. Iroezi, J. Garcia, J. Warrior, D. Estrin, and M. Srivastava, "Cyclops: In situ image sensing and interpretation in wireless sensor networks," in *Proceedings of the ACM Conference on Embedded Networked Sensor Systems*, 2005.

[15] A. Rowe, C. Rosenberg, and I. Nourbakhsh, "A low cost embedded color vision system," in *Proceedings of IROS 2002*, 2002.

[16] W.-C. Feng, B. Code, E. Kaiser, W.-C. Feng, and M. Le Baillif, "Panoptes: Scalable low-power video sensor networking technologies," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 1, no. 2, pp. 151–167, May 2005.

[17] S. Hengstler and H. Aghajan, "A smart camera mote architecture for distributed intelligent surveillance," in *ACM SenSys Workshop on Distributed Smart Cameras*, 2006.

[18] "Crossbow wireless sensor platforms," in *http://www.xbow.com*, 2008.

[19] M. Antone and S. Teller, “Scalable, extrinsic calibration of omni-directional image networks,” *International Journal of Computer Vision*, vol. 49, no. 2/3, pp. 143–174, September/October 2002.

[20] G. Sharp, S. Lee, and D. Wehe, “Multiview registration of 3-D scenes by minimizing error between coordinate frames,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, Copenhagen, Denmark, 2002, pp. 587–597.

[21] P. Kulkarni, P. Shenoy, and D. Ganesan, “Approximate initialization of camera sensor networks,” in *Proceedings of the 4th European Conference on Wireless Sensor Networks*, 2007.

[22] D. Makris, T. Ellis, and J. Black, “Bridging the gaps between cameras,” in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2004.

[23] D. Marinakis and G. Dudek, “Topology inference for a vision-based sensor network,” in *In Proc. of Canadian Conference on Computer and Robot Vision*, May 2005.

[24] X. Zou, B. Bhanu, B. Song, and A. Roy-Chowdhury, “Determining topology in a distributed camera network,” in *Proceedings of the IEEE International Conference on Image Processing*, 2007.

[25] O. Javed, Z. Rasheed, K. Shafique, and M. Shah, “Tracking across multiple cameras with disjoint views,” in *Proceedings of the 9th International Conference on Computer Vision*, Nice, France, 2003.

[26] C. Niu and E. Grimson, “Recovering non-overlapping network topology using far-field vehicle tracking data,” in *Proceedings of the 18th International Conference on Pattern Recognition*, 2006.

[27] C. Jaynes, “Multi-view calibration from planar motion for video surveillance,” in *Second IEEE Workshop on Visual Surveillance (VS’99)*, 1999, pp. 59–66.

[28] A. Rahimi, B. Dunagan, and T. Darrell, “Simultaneous calibration and tracking with a network of non-overlapping sensors,” in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, 2004, pp. 187–194.

[29] J. Black, T. Ellis, and P. Rosin, “Multi view image surveillance and tracking,” in *Proceedings of the Workshop on Motion and Video Computing*, 2002, pp. 169–174.

[30] L. Lee, R. Romano, and G. Stein, “Monitoring activities from multiple video streams: Establishing a common coordinate frame,” *IEEE Transactions on Pattern Recognition and Machine Intelligence*, vol. 22, no. 8, August 2000.

[31] A. van den Hengel, A. Dick, H. Detmold, A. Cichowski, and R. Hill, “Finding camera overlap in large surveillance networks,” in *Proceedings of the 8th Asian Conference on Computer Vision*, November 2007.

[32] M. Meingast, S. Oh, and S. Sastry, “Automatic camera network localization using object image tracks,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV) Workshop on Visual Representations and Modeling of Large-scale Environments*, October 2007.

[33] E. Kang, I. Cohen, and G. Medioni, “A graph-based global registration for 2D mosaics,” in *Proceedings of the 15th International Conference on Pattern Recognition (ICPR)*, 2000, pp. 257–260.

[34] R. Marzotto, A. Fusiello, and V. Murino, “High resolution video mosaicing with global alignment,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2004, pp. 692–698.

[35] H. Sawhney, S. Hsu, and R. Kumar, “Robust video mosaicing through topology inference and local to global alignment,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 1998, pp. 103–119.

[36] M. Brown and D. Lowe, “Recognising panoramas,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2003.

[37] M. Brown, R. Szeliski, and S. Winder, “Multi-image matching using multi-scale oriented patches,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, San Diego, CA., 2005, pp. 510–517.

[38] F. Schaffalitzky and A. Zisserman, “Multi-view matching for unordered image sets,” in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2002, pp. 414–431.

[39] S. Avidan, Y. Moses, and Y. Moses, “Centralized and distributed multi-view correspondence,” *International Journal of Computer Vision*, vol. 71, no. 1, pp. 49–69, 2007.

[40] P. Baker and Y. Aloimonos, “Complete calibration of a multi-camera network,” in *Proceedings of IEEE Workshop on Omnidirectional Vision 2000*, 2000.

[41] X. Chen, J. Davis, and P. Slusallek, “Wide area camera calibration using virtual calibration objects,” in *IEEE Comp. Soc. Conf. on Computer Vision and Pattern Recognition*, 2000.

[42] A. Barton-Sweeney, D. Lymberopoulos, and A. Savvides, “Sensor localization and camera calibration in distributed camera sensor networks,” in *Proceedings of the 3rd International Conference on Broadband Communications, Networks and Systems (BROADNETS)*, San Jose, CA, USA, 2006, pp. 1–10.

[43] C. J. Taylor and B. Shirmohammadi, “Self localizing smart camera networks and their applications to 3D modeling,” in *Proceedings of the International Workshop on Distributed Smart Cameras*, October 2006.

[44] H.-G. Maas, “Image sequence based automatic multi-camera system calibration techniques,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 54, no. 5-6, pp. 352–359, December 1999.

[45] P. Baker and Y. Aloimonos, “Calibration of a multicamera network,” in *Proceedings of the IEEE Workshop on Omnidirectional Vision*, 2003.

[46] I. M. Rekletis and G. Dudek, “Automated calibration of a camera sensor network,” in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Aug. 2-6 2005, pp. 401–406.

[47] I. Rekleitis, D. Meger, and G. Dudek, “Simultaneous planning, localization, and mapping in a camera sensor network,” *Robotics and Autonomous Systems*, vol. 54, no. 11, pp. 921–932, November 2006.

[48] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[49] K. Mikolajczyk and C. Schmid, “A performance evaluation of local descriptors,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.

[50] ——, “Scale and affine invariant interest point detectors,” *International Journal of Computer Vision*, vol. 60, no. 1, pp. 63–86, October 2004.

[51] J. H. Freidman, J. L. Bentley, and R. A. Finkel, “An algorithm for finding best matches in logarithmic expected time,” *ACM Transactions on Mathematical Software*, vol. 3, no. 3, pp. 209–226, September 1977.

[52] J. Jannotti and J. Mao, “Distributed calibration of smart cameras,” in *Proceedings of the Workshop on Distributed Smart Cameras*, October 2006.

[53] Z. Zhang, “Determining the epipolar geometry and its uncertainty: A review,” *International Journal of Computer Vision*, vol. 27, no. 2, pp. 161–195, 1998.

[54] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[55] H. V. Poor, *An Introduction to Signal Detection and Estimation*. Springer, 1998.

[56] K. Langendoen and N. Reijers, “Distributed localization in wireless sensor networks: A quantitative comparison,” *Computer Networks*, vol. 43, no. 4, pp. 499–518, November 2003.

[57] R. Tsai, “A versatile camera calibration technique for high-accuracy 3-D machine vision metrology using off-the-shelf TV cameras and lenses,” in *Radiometry – (Physics-Based Vision)*, L. Wolff, S. Shafer, and G. Healey, Eds. Jones and Bartlett, 1992.

[58] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, “Bundle adjustment – A modern synthesis,” in *Vision Algorithms: Theory and Practice*, ser. LNCS, W. Triggs, A. Zisserman, and R. Szeliski, Eds. Springer Verlag, 2000, pp. 298–375.

[59] Y. Ma, S. Soatto, J. Košecká, and S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer, 2004, ch. 11.

[60] J. Leonard and H. Durrant-Whyte, “Simultaneous map building and localization for an autonomous mobile robot,” in *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems*, 1991, pp. 1442–1447.

[61] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics: Intelligent Robotics and Autonomous Agents*. MIT Press, 2005.

[62] X. Liu, P. Kulkarni, P. Shenoy, and D. Ganesan, “Snapshot: A self-calibration protocol for camera sensor networks,” in *Proceedings of IEEE/CreateNet BASENETS 2006*, 2006.

[63] H. Lee and H. Aghajan, “Collaborative node localization in surveillance networks using opportunistic target observations,” in *Proceedings of the 4th ACM International Workshop on Video Surveillance and Sensor Networks*, 2006, pp. 9–18.

[64] S. Funiak, C. Guestrin, M. Paskin, and R. Sukthankar, “Distributed localization of networked cameras,” in *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks*, April 2006.

[65] 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.

[66] D. Devarajan, “Distributed localization of camera networks,” Ph.D. dissertation, Rensselaer Polytechnic Institute, 2007.

[67] M. Pollefeys, R. Koch, and L. J. Van Gool, “Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters,” in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1998, pp. 90–95.

[68] M. Pollefeys, F. Verbiest, and L. J. V. Gool, “Surviving dominant planes in uncalibrated structure and motion recovery,” in *Proceedings of the 7th European Conference on Computer Vision-Part II (ECCV ’02)*. London, UK: Springer-Verlag, 2002, pp. 837–851.

[69] P. Sturm and B. Triggs, “A factorization based algorithm for multi-image projective structure and motion,” in *Proceedings of the 4th European Conference on Computer Vision (ECCV ’96)*, 1996, pp. 709–720.

[70] B. Triggs, “Factorization methods for projective structure and motion,” in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR ’96)*. San Francisco, CA, USA: IEEE Comput. Soc. Press, 1996, pp. 845–51.

[71] M. Andersson and D. Betsis, “Point reconstruction from noisy images,” *Journal of Mathematical Imaging and Vision*, vol. 5, pp. 77–90, January 1995.

[72] R. Hartley and P. Sturm, “Triangulation,” *Computer Vision and Image Understanding*, vol. 68, no. 2, pp. 146–157, 1997.

[73] D. Estrin, R. Govindan, J. S. Heidemann, and S. Kumar, “Next century challenges: Scalable coordination in sensor networks,” in *Proceedings of ACM/IEEE Conference on Mobile Computing and Networking (MobiCom 99)*, Seattle, WA, USA, 1999, pp. 263–270.

[74] N. Sastry, U. Shankar, and D. Wagner, “Secure verification of location claims,” in *Proceedings of WiSe ’03, the 2nd ACM workshop on Wireless Security*, 2003, pp. 1–10.

[75] S. Capkun and J.-P. Hubaux, “Secure positioning of wireless devices with application to sensor networks,” in *Proceedings of IEEE INFOCOM*, 2005.

[76] Y. Wei, Z. Yu, and Y. Guan, “Location verification algorithms for wireless sensor networks,” in *Proceedings of the 27th International Conference on Distributed Computing Systems*, 2007, p. 70.

[77] W. Du, L. Fang, and N. Peng, “LAD: localization anomaly detection for wireless sensor networks,” *J. Parallel Distrib. Comput.*, vol. 66, no. 7, pp. 874–886, 2006.

[78] K. P. Murphy, Y. Weiss, and M. I. Jordan, “Loopy belief propagation for approximate inference: An empirical study,” in *Proceedings of Uncertainty in Artificial Intelligence (UAI ’99)*, 1999, pp. 467–475.

[79] S. Venkatesh, M. Alanyali, and O. Savas, “Distributed detection in sensor networks with packet losses and finite capacity links,” *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4118–4132, November 2006.

[80] C. Christopher and P. Avi, “Loopy belief propagation as a basis for communication in sensor networks,” in *Proceedings of the 19th Annual Conference on Uncertainty in Artificial Intelligence (UAI-03)*. San Francisco, CA: Morgan Kaufmann Publishers, 2003, pp. 159–166.

[81] J. Thorpe and R. McEliece, “Data Fusion Algorithms for Collaborative Robotic Exploration,” *Interplanetary Network Progress Report*, vol. 149, pp. 1–14, Jan. 2002.

[82] A. Ihler, J. Fisher, R. Moses, and A. Willsky, “Nonparametric belief propagation for self-calibration in sensor networks,” in *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks*, Berkeley, CA, April 2004.

[83] J. Besag, “Spatial interaction and the statistical analysis of lattice systems,” *Journal of the Royal Statistical Society*, vol. B, no. 36, pp. 192–236, 1974.

[84] J. Hammersley and P. E. Clifford, “Markov fields on finite graphs and lattices,” Unpublished article, 1971.

[85] J. Pearl, *Probablistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

[86] F. Kschischang, B. J. Frey, and H.-A. Loeliger, “Factor graphs and the sum-product algorithm,” *IEEE Transactions on Information Theory*, vol. 47, 2001.

[87] R. Smith, M. Self, and P. Cheeseman, “Estimating uncertain spatial relationships in robotics,” in *Autonomous Robot Vehicles*. New York: Springer-Verlag, 1990, pp. 167–193.

[88] J. V. Mascia, “Actuation-assisted external calibration of distributed camera sensor networks,” Master’s thesis, University of California Los Angeles, 2007.

[89] P. Bolliger, M. Köhler, and K. Römer, “Facet: Towards a smart camera network of mobile phones,” in *Proceedings of Autonomics 2007 (ACM First International Conference on Autonomic Computing and Communication Systems)*, Rome, Italy, October 2007.

[90] M. A. Paskin and C. E. Guestrin, “Robust probabilistic inference in distributed systems,” in *Proceedings of the Twentieth Conference on Uncertainty in Artificial Intelligence (UAI ’04)*, 2004, pp. 436–445.

[91] M. A. Paskin, C. E. Guestrin, and J. McFadden, “A robust architecture for inference in sensor networks,” in *4th International Symposium on Information Processing in Sensor Networks (IPSN ’05)*, 2005.

[92] F. Dellaert, A. Kipp, and P. Krauthausen, “A multifrontal QR factorization approach to distributed inference applied to multirobot localization and mapping.” in *Proceedings of National Conference on Artificial Intelligence (AAAI 05)*, 2005, pp. 1261–1266.