Determining Vision Graphs for Distributed Camera Networks Using Feature Digests

Zhaolin Cheng, Dhanya Devarajan, and Richard J. Radke*
Department of Electrical, Computer, and Systems Engineering
Rensselaer Polytechnic Institute, Troy, NY 12180
chengz@rpi.edu, devard@rpi.edu, rjradke@ecse.rpi.edu

* This paper has been accepted for publication in *EURASIP Journal of Applied Signal Processing, Special Issue on Visual Sensor Networks*, to appear Fall 2006. Please address correspondence to Richard Radke.

Abstract

We propose a method for obtaining 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. The technique is decentralized, requires no ordering on the set of cameras, and assumes that cameras can only communicate a finite amount of information with each other in order to establish the vision graph. Each camera first detects a large number of feature points that are approximately scale- and viewpoint-invariant. Both the number of features and the length of each feature descriptor are substantially reduced to form a fixed-length "feature digest" that is broadcast to the rest of the network. Each receiver camera decompresses the feature digest to recover approximate feature descriptors, robustly estimates the epipolar geometry to reject incorrect matches and grow additional ones, and decides whether sufficient evidence exists to form a vision graph edge. We use receiver-operating-characteristics (ROC) curves to analyze the performance of different message formation schemes, and show that high detection rates can be achieved while maintaining low false alarm rates. Finally, we show how a camera calibration algorithm that passes messages only along vision graph edges can recover accurate 3D structure and camera positions in a distributed manner. We demonstrate the accurate performance of the vision graph generation and camera calibration algorithms using a simulated 60-node outdoor camera network. In this simulation, we achieved vision graph edge detection probabilities exceeding 0.8 while maintaining false alarm probabilities below 0.05.

I. Introduction

The automatic calibration of a collection of cameras (i.e. estimating their position and orientation relative to each other and to their environment) is a central problem in computer vision that requires techniques for both detecting/matching feature points in the images acquired from the collection of cameras and for subsequently estimating the camera parameters. While these problems have been extensively studied, most prior work assumes that they are solved at a single processor after all of the images have been collected in one place. This assumption is reasonable for much of the early work on multi-camera vision in which all the cameras are in the same room (e.g. [8], [19]). However, recent developments in wireless sensor networks have made feasible a *distributed camera network*, 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. We will require new techniques for correspondence and calibration that are well-suited to such distributed camera networks– techniques that take explicit account of the underlying communication network and its constraints.

In this paper, we address the problem of efficiently estimating the *vision graph* for an ad-hoc camera

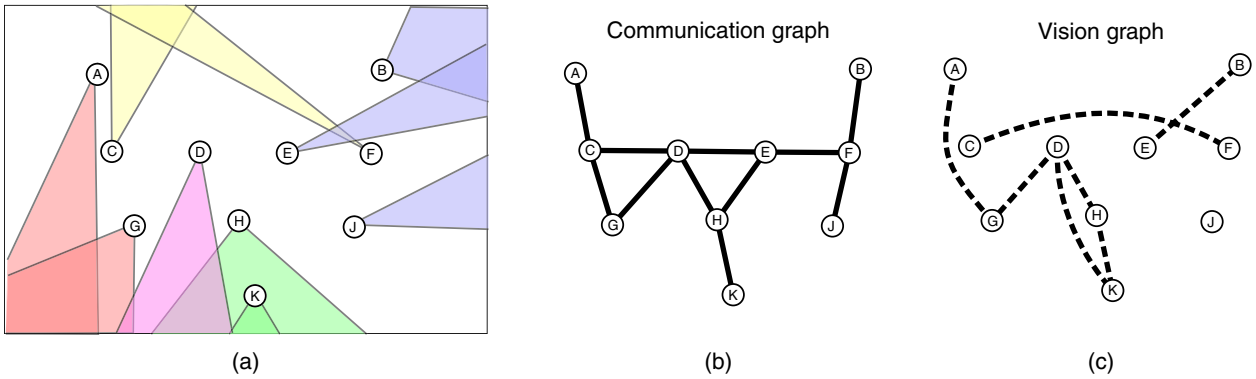


Fig. 1. (a) A snapshot of the instantaneous state of a camera network, indicating the fields of view of ten cameras. (b) A possible communication graph. (c) The associated vision graph.

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 (more precisely, an edge exists if a stable, accurate estimate of the epipolar geometry can be obtained). This graph will be necessary for camera calibration as well as subsequent higher-level vision tasks such as object tracking or 3D reconstruction. We can think of the vision graph as an overlay graph on the underlying *communication graph*, which describes the cameras that have direct communication links. We note that since cameras are oriented, fixed-aperture sensors, an edge in the communication graph does not always imply an edge in the vision graph, and vice versa. For example, Figure 1 illustrates a hypothetical network of ten cameras. We note that cameras E and H, while physically proximate, image no common scene points, while cameras C and F image some of the same scene points despite being physically distant.

The main contribution of the paper is the description and analysis of an algorithm for estimating the vision graph. The key motivation for the algorithm is that we seek a *decentralized technique* in which an *unordered* set of cameras can only communicate a *finite amount* of information with each other in order to establish the vision graph and mutual correspondences. The underlying communication constraint is not usually a consideration in previous work on image correspondence from the computer vision community, but would be critical to the success of actual field implementations of wireless camera networks. Each camera independently composes a fixed-length message that is a compressed representation of its detected features, and broadcasts this "feature digest" to the whole network. The basic idea is to select a spatially-well-distributed subset of distinctive features for transmission to the broader network, and compress them with principal component analysis. Upon receipt of a feature digest message, a receiver node compares its own features to the decompressed features, robustly estimates the epipolar geometry, and decides whether

the number of robust matches constitutes sufficient evidence to establish a vision graph edge with the sender.

The paper is organized as follows. Section II reviews prior work related to the estimation of vision graphs, and Section III discusses methods from the computer vision literature for detecting and describing salient feature points. Section IV presents the key contribution of the paper, our framework for establishing the vision graph, which includes message formation, feature matching, and vision graph edge detection. In Section V, we briefly describe how the camera network can be calibrated by passing messages along established vision graph edges. The calibration approach is based on our previously published work [9], which assumed that the vision graph was given. The distributed algorithm results in a metric reconstruction of the camera network, based on structure-from-motion algorithms. Section VI presents a performance analysis on a set of 60 outdoor images. For the vision graph estimation algorithm, we examine several tradeoffs in message composition including the spatial distribution of features, the number of features in the message, the amount of descriptor compression, and the message length. Using receiver-operating-characteristics (ROC) curves, we show how to select the feature messaging parameters that best achieve desired tradeoffs between the probabilities of detection and false alarm. We also demonstrate the accurate calibration of the camera network using the distributed structure-from-motion algorithm, and show that camera positions and 3D structures in the environment can be accurately estimated. Finally, Section VII concludes the paper and discusses directions for future work.

II. Related Work

In this section, we review work from the computer vision community related to the idea of estimating a vision graph from a set of images. We emphasize that in contrast to the work described here, communication constraints are generally not considered in these approaches, and that images from all the cameras are typically analyzed at a powerful, central processor.

Teller and Antone [38] 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. [33] addressed how to distribute errors in estimates of camera calibration parameters with respect to a vision graph, but this graph was manually constructed. We also note that Huber [18] and Stamos and Leordeanu [36] considered graph formalisms for matching 3D range datasets. However, this problem of matching 3D sub-shapes is substantially different from the problem of matching patches of 2D images (for example, there are virtually no difficulties with

illumination variation or perspective distortion in range data).

Graph relationships on image sequences are frequently encountered in image mosaicking applications, e.g. [20], [24], [30]. 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 in this paper 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.

A related area of research involves estimating the homographies that relate the ground plane of an environment as imaged by multiple cameras. Tracking and associating objects moving on the ground plane (e.g. walking people) can be used to estimate the visual overlap of cameras in the absence of calibration (e.g. see [21], [7]). Unlike these approaches, the method described here requires neither the presence of a ground plane or the tracking of moving objects.

The work of Brown and colleagues [4], [5] represents the state of the art in 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 [31] 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 than what we propose was described by Avidan et al. [2], 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. However, the results in that work were only validated on synthetic data, and did not extend to the demonstration of camera calibration discussed here.

III. Feature Detectors and Descriptors

The 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. 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. An early approach was the Harris corner detector [14], which finds locations where both eigenvalues of the local gradient matrix (see (1) below) are large. Mikolajczyk and Schmid [25] later extended Harris corners to a multi-scale setting. An alternate approach is to filter the image at multiple scales with a Laplacian-of-Gaussian (LOG) [22] or Difference-of-Gaussian (DOG) [23] filter; scale-space extrema of the filtered image give the locations of the interest points. A broad survey of modern feature detectors was given by Mikolajczyk and Schmid [26]. As described below, we use Difference-of-Gaussian (DOG) features in our framework.

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. Schmid and Mohr [32] proposed a descriptor that was invariant to the rotation of the feature. This was followed by Lowe's popular SIFT feature descriptor [23], which is a histogram of gradient orientations designed to be invariant to scale and rotation of the feature. Typically, the algorithm takes a 16×16 grid of samples from the the gradient map at the feature's scale, and uses it to form a 4×4 aggregate gradient matrix. Each element of the matrix is quantized into 8 orientations, producing a descriptor of dimension 128. Baumberg [3] and Schaffalitzky and Zisserman [31] applied banks of linear filters to affine invariant support regions to obtain feature descriptors.

In the proposed algorithm, we detect DOG features and compute SIFT descriptors as proposed by Lowe [23]. Mikolajczyk and Schmid [27] showed that this combination outperformed most other detector/descriptor combinations in their experiments. As will be discussed in Section IV-A, we also apply an image-adaptive principal component analysis [10] to further compress feature descriptors.

IV. The Feature Digest Algorithm

When a new camera enters the network, there is no way to know *a priori* which other network cameras should share a vision graph edge with it. Hence, it is unavoidable that a small amount of information from the new camera is disseminated throughout the entire network. We note that there is substantial research in the networking community on how to efficiently deliver a message from one node to all other nodes in the network. Techniques range from the naïve method of flooding [34] to more recent, power-efficient methods such as Heinzelman et al.'s SPIN [17] or LEACH [16]. Our focus here is not on the mechanism of broadcast but on the efficient use of bits in the broadcast message. We show how the most useful information from the new camera can be compressed into a fixed-length feature message (or "digest"). We assume that the message length is determined beforehand based on communication and power constraints.

Our strategy is to select and compress only highly-distinctive, spatially well-distributed features which are likely to match features in other images. When another camera node receives this message, it will decide whether there is sufficient evidence to form a vision graph edge with the sending node, based on the number of features it can robustly match with the digest. Clearly, there are tradeoffs for choosing the number of features and the amount of compression to suit a given feature digest length; we explore these tradeoffs in Section VI. We now discuss the feature detection and compression algorithm that occurs at each sending node and the feature matching and vision graph edge decision algorithm that occurs at each receiving node in greater detail.

A. Feature Subset Selection and Compression

The first step in constructing the feature digest at the sending camera is to detect Difference-of-Gaussian (DOG) features in that camera's image, and compute a SIFT descriptor of length 128 for each feature. 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 [23] for more details). For a typical image, N is on the order of hundreds or thousands.

The next step is to select a subset containing M of the N features for the 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 characterize feature distinctiveness using a strength measure defined as follows. We first compute the local gradient matrix

$$\mathbf{G} = \frac{1}{|W|^2} \begin{bmatrix} \sum_W g_x g_x & \sum_W g_x g_y \\ \sum_W g_y g_x & \sum_W g_y g_y \end{bmatrix}, \tag{1}$$

where g_x and g_y are the finite difference derivatives in the x and y dimensions respectively, and the sum is computed over an adaptive window W around each detected feature. If the scale of a feature is σ, we found a window side of $|W| = \sqrt{2}\sigma$ to be a good choice that captures the important local signal variation around the feature. We then define the strength of feature i as

$$s_i = \frac{det\ \mathbf{G}_i}{tr\ \mathbf{G}_i}, \tag{2}$$

which was suggested by Brown et al. [5].

If the digest is to contain M features, we could just send the M strongest features using the above strength measure. However, in practice, there may be clusters of strong features in small regions of the

(a) (b)

Fig. 2. The goal is to select 256 representative features in the image. (a) The 256 strongest features are concentrated in a small area in the image–more than 95% are located in the tree at upper left. (b) After applying the k-d tree partition with 128 leaf nodes, the features are more uniformly spatially distributed.

image that have similar textures, and would unfairly dominate the feature list (see Figure 2a). Therefore, we need a way to distribute the features more fairly across the image.

We propose an approach based on k-d trees to address this problem. The k-d tree is a generalized binary tree that has proven to be very effective for partitioning data in high-dimensional spaces [12]. The idea is to successively partition a dataset into rectangular regions such that each partition cuts the region with the current highest variance in two, using the median data value as the dividing line. In our case, we use a 2-dimensional k-d tree containing c cells constructed from the image coordinates of feature points. In order to obtain a balanced tree, we require the number of leaf nodes to be a power of 2. For each nonterminal node, we partition the node's data along the dimension that has larger variance. The results of a typical partition are shown in Figure 2b. Finally, we select the $\lfloor \frac{M}{c} \rfloor$ strongest features from each k-d cell to add to the feature digest. Figure 2 compares the performance of the feature selection algorithm *with* and *without* the k-d tree. One can see that with the k-d tree, features are more uniformly spatially distributed across the image, and thus we expect a higher number of features may match any given overlapped image. This is similar to Brown et al.'s approach, which uses Adaptive Non-Maximal Suppression (ANMS) to select spatially-distributed multi-scale Harris corners [5]. Clearly, there will be a performance tradeoff between the number of cells and the number of features per cell. While there is probably no optimal number of cells for an arbitrary set of images, by using a training subset of 12 overlapping images (in total 132 pairs), we found that $c = 2^{\lfloor log_2(M) \rfloor}$ gave the most correct matches.

Once the M features have been selected, we compress them so that each is represented with K

parameters (instead of the 128 SIFT parameters). 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 $\{\overline{\mathbf{v}}, \mathbf{Q}, \mathbf{p}_1, \ldots, \mathbf{p}_M, (x_1, y_1), \ldots, (x_M, y_M)\}$, where $\overline{\mathbf{v}} \in \mathbb{R}^{128}$ is the mean of the N SIFT descriptors, $\mathbf{Q}$ is the $128 \times K$ matrix of principal component vectors, $\mathbf{p}_j = \mathbf{Q}^T(\mathbf{v}_j - \overline{\mathbf{v}}) \in \mathbb{R}^K$, where $\mathbf{v}_j$ is the jth selected feature's SIFT descriptor $\in \mathbb{R}^{128}$, and (x_j, y_j) are the image coordinates of the jth 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

$$L = b(128(K+1) + M(K+2)), \tag{3}$$

where b is the number of bytes used to represent a real number. In our experiments, we chose $b = 4$ for all parameters; however, in the future, coding gains could be obtained by adaptively varying this parameter. 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). We analyze these tradeoffs in Section VI.

B. Feature Matching and Vision Graph Edge Detection

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 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 $\widehat{\mathbf{v}}_j = \mathbf{Q}\mathbf{p}_j + \overline{\mathbf{v}}$. 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, 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 [23], 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). In our experiments, we used a threshold of 0.6. However, it is possible that this process may reject correctly matched features or include false matches (also known as outliers). Furthermore, correct feature matches may exist at the receiver that are not the closest matches in terms of Euclidean distance between descriptors. To combat the outlier problem, we robustly estimate

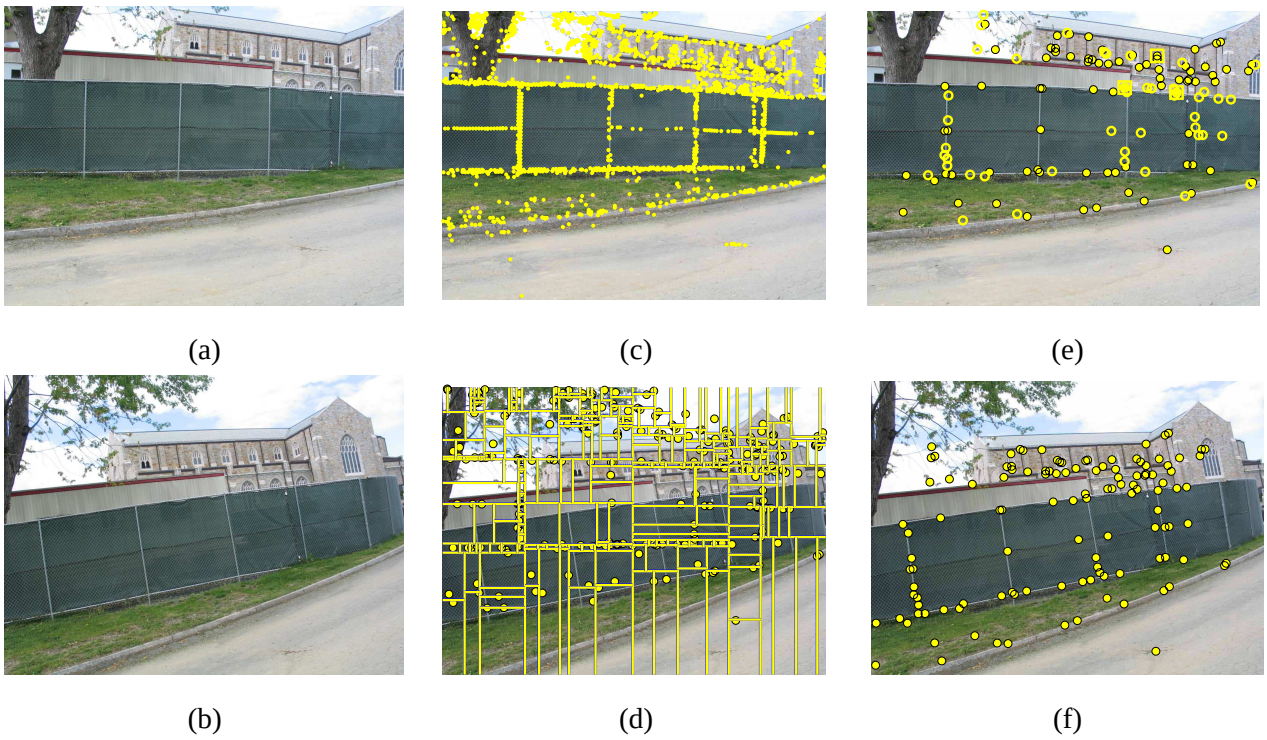


Fig. 3. Example results of image matching from a pair of images. (a) Image 1. (b) Image 2. (c) The 1976 detected features in Image 1. (d) The k-d tree and corresponding 256-feature digest in image 1. (e) The dots indicate 78 features in Image 1 detected as correspondences in Image 2, using the minimal 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.

the epipolar geometry, and reject features that are inconsistent with it [15]. 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 the ratio d_j^1/d_j^2 is suitably low. This process is illustrated in Figure 3.

Based on the grown matches, we simply declare a vision graph edge if the number of final feature matches exceeds a threshold τ, since it is highly unlikely that a large number of good matches consistent with the epipolar geometry occur by chance. In Section VI, we investigate the effects of varying the threshold on vision graph edge detection performance.

We note that it would be possible to send more features for the same K and L if we sent only feature descriptors and not feature locations. However, we found that being able to estimate the epipolar geometry at the receiver definitely improves performance, as exemplified by the number of accurately grown correspondences in Figure 3e.

Once the vision graph is established, we can use feedback in the network to refine edge decisions. In particular, false vision graph edges that remain after the process described above can be detected and removed by sending uncompressed features from one node to another and robustly estimating the epipolar geometry based on all of the available information (see Section VI). However, such messages would be accomplished via more efficient point-to-point communication between the affected cameras, as opposed to a general feature broadcast.

V. Calibrating the Camera Network

Next, we briefly describe how the camera network can be calibrated, given the vision graph edges and correspondences estimated above. We assume that the vision graph $G = (V, E)$ contains m nodes, each representing a perspective camera described by a 3×4 matrix P_i:

$$P_i = K_i R_i^T \left[I \ \ -C_i \right]. \tag{4}$$

Here, $R_i \in SO(3)$ and $C_i \in \mathbb{R}^3$ are the rotation matrix and optical center comprising the external camera parameters. K_i is the intrinsic parameter matrix, which we assume here can be written as $diag(f_i, f_i, 1)$, where f_i is the focal length of the camera. (Additional parameters can be added to the camera model, e.g. principal points or lens distortion, as the situation warrants.)

Each camera images some subset of a set of n scene points $\{X_1, X_2, \ldots, X_n\} \in \mathbb{R}^3$. This subset for camera i is described by $V_i \subset \{1, \ldots, n\}$. The projection of X_j onto P_i is given by $u_{ij} \in \mathbb{R}^2$ for $j \in V_i$:

$$\lambda_{ij} \begin{bmatrix} u_{ij} \\ 1 \end{bmatrix} = P_i \begin{bmatrix} X_j \\ 1 \end{bmatrix}, \tag{5}$$

where λ_{ij} is called the projective depth [37].

We define the neighbors of node i in the vision graph as $N(i) = \{j \in V | (i, j) \in E\}$. To obtain a distributed initial estimate of the camera parameters, we use the algorithm we previously described in [9], which operates as follows at each node i:

1) Estimate a projective reconstruction based on the common scene points shared by i and $N(i)$ (these points are called the "nucleus"), using a projective factorization method [37].
2) Estimate a metric reconstruction from the projective cameras, using a method based on the dual absolute quadric [28].
3) Triangulate scene points not in the nucleus using the calibrated cameras [1].
4) Use RANSAC [11] to reject outliers with large reprojection error, and repeat until the reprojection error for all points is comparable to the assumed noise level in the correspondences.

5) Use the resulting structure-from-motion estimate as the starting point for full bundle adjustment [39]. That is, if $\hat{u}_{jk}$ represents the projection of the estimate $\hat{X}_k^i$ onto the estimate $\hat{P}_j^i$, then a nonlinear minimization problem is solved at each node i, given by

$$\min_{\substack{\{P_j^i\},j\in\{i,N(i)\}\\ \{X_k^i\},k\in\cap V_j}} \sum_j \sum_k (\hat{u}_{jk}-u_{jk})^T \Sigma_{jk}^{-1} (\hat{u}_{jk}-u_{jk}) \tag{6}$$

where Σ_{jk} is the 2×2 covariance matrix associated with the noise in the image point u_{jk}. The quantity inside the sum is called the Mahalanobis distance between $\hat{u}_{jk}$ and u_{jk}.

If the local calibration at a node fails for any reason, a camera estimate is acquired from a neighboring node prior to bundle adjustment. At the end of this initial calibration, each node has estimates of its own camera parameters P_i^i as well as those of its neighbors in the vision graph $P_j^i, j\in N(i)$.

VI. Experimental Results

We simulated an outdoor camera network using a set of 60 widely-separated images acquired from a Canon PowerShot G5 digital camera in autofocus mode (so that the focal length for each camera is different and unknown), using an image resolution of 1600 $\times$ 1200. Figure 4 shows some example images from the test set. The scene includes several buildings, vehicles, and trees, and many repetitive structures (e.g. windows). A calibration grid was used beforehand to verify that for this camera, the skew was negligible, the principal point was at the center of the image plane, the pixels were square, and there was virtually no lens distortion. Therefore, our pinhole projection model with a diagonal K matrix is justified in this case. We determined the ground truth vision graph manually by declaring a vision graph edge between two images if they have more than about 1/8 area overlap. Figure 5 shows the ground truth expressed as a sparse matrix.

We evaluated the performance of the vision graph generation algorithm using fixed message sizes of length L =80, 100, and 120 kilobytes. Recall that the relationship between the message length L, the number of features M and the number of PCA components K is given by (3). Our goal here is to find the optimal combination of M and K for each L. We model the establishment of vision graph edges as a typical detection problem [29], 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 (i.e. the algorithm finds an edge while there is actually an edge) against the probability of false alarm (i.e. the algorithm finds an edge while the two images actually have little or no overlap). We denote the two probabilities as p_d and p_{fa} respectively. Different points on the curve are generated by choosing different thresholds for the number of matches necessary to provide sufficient evidence for an edge.

Fig. 4. Sample images from the 60-image test set.

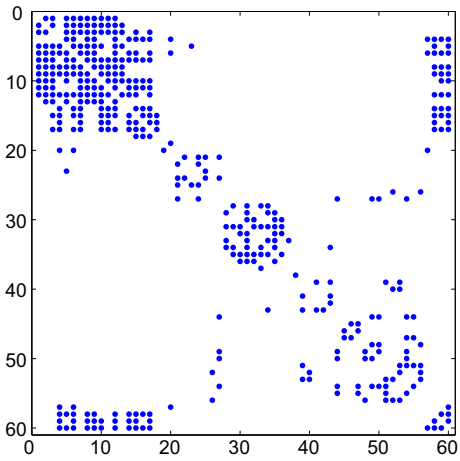
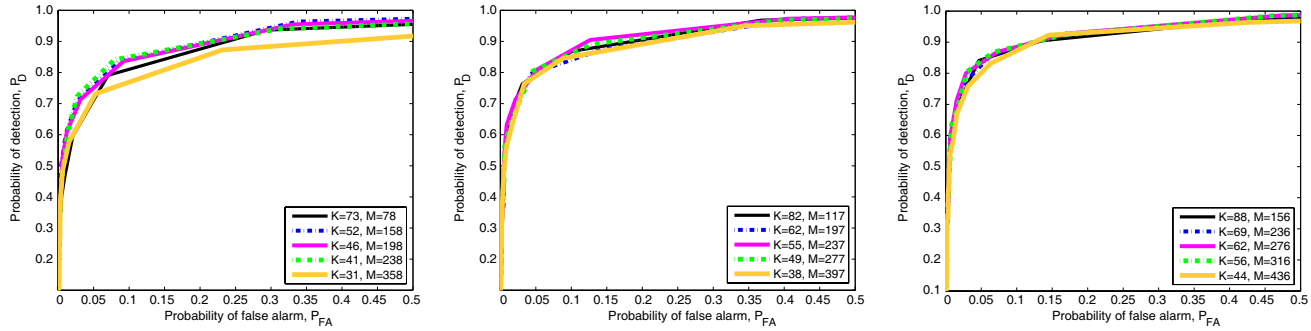


Fig. 5. The ground truth vision graph for the test image set. A dot at (i, j) indicates a vision graph edge between cameras i and j.



(a) (b) (c)

Fig. 6. ROC curves giving detection probability p_d vs. false alarm probability p_{fa}, when messages of length (a) 80 KB, (b) 100 KB, and (c) 120 KB are transmitted to establish the vision graph.

The user can select an appropriate point on the ROC curve based on application requirements on the performance of the predictor. Figure 6 shows the ROC curves for the 80 KB, 100 KB, and 120 KB cases for different combinations of M and K. By taking the upper envelope of each graph in Figure 6, we can obtain overall "best" ROC curves for each L, which are compared in Figure 7. In Figure 7, we also indicate the "ideal" ROC curve that is obtained by applying our algorithm using all features from each image and no compression. We can draw several conclusions from these graphs.

1) 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 expected, the performance improves with the message length.
2) Generally, neither extreme of making the number of features very large (the light solid lines in Figure 6) or the number of principal components very large (the dark solid lines in Figure 6) is optimal. The best detector performance is generally achieved at intermediate values of both parameters.
3) 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 7).

To calibrate the camera network, we chose the vision graph specified by the circle on the 120 KB curve in Figure 7, at which p_d =0.89 and p_{fa} =0.08. Then, each camera on one side of a vision graph edge communicated all of its features to the camera on the other side. This full information was used to re-estimate the epipolar geometry relating the camera pair and enabled many false edges to be detected and discarded. The resulting sparser, more accurate vision graph is denoted by the square in Figure 7, at

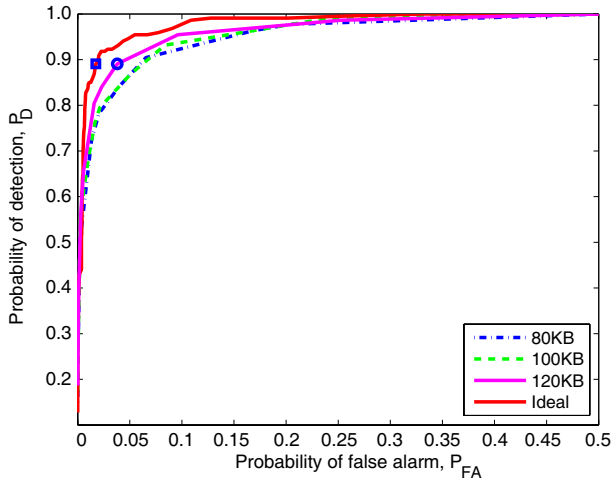
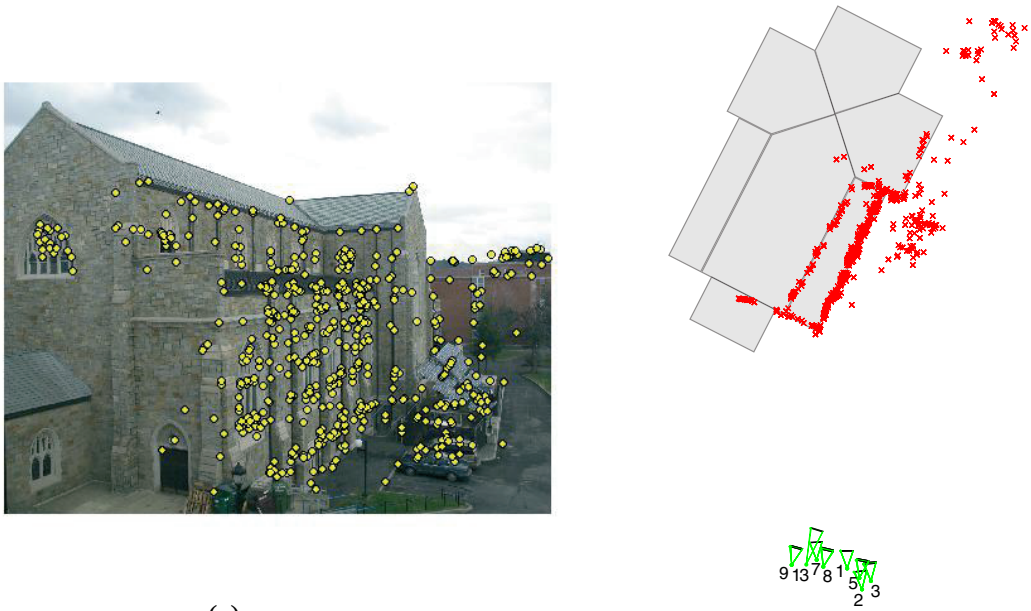


Fig. 7. Best achievable ROC curves for message lengths 80KB, 100KB and 120KB. These are obtained by taking the upper envelope of each curve in Figure 6 (and thus, each line segment corresponds to a different choice of M and K.) The "ideal" curve is generated by applying our algorithm using all features from each image and no compression.

which p_d =0.89 and p_{fa} =0.03. The correspondences along each vision graph edge provide the inputs u_{ij} required for the camera calibration algorithm, as described in Section V.

The camera calibration experiment was initially performed on the full set of 60 images. However, since the calibration algorithm has stricter requirements on image relationships than the vision graph estimation algorithm, not all 60 cameras in the network were ultimately calibrated due to several factors. Eight images were automatically removed from consideration due to insufficient initial overlap with other images, and seven additional images were eliminated by the RANSAC algorithm since a minimum number of inliers for metric reconstruction could not be found. Finally, five images were removed from consideration because a metric reconstruction could not be obtained (e.g. when the inlier feature points were almost entirely coplanar). Consequently, 40 cameras were ultimately calibrated.

The ground truth calibration for this collection of cameras is difficult to determine, since it would require a precise survey of multiple buildings and absolute 3D localization (both position and orientation) of each camera. However, we can evaluate the quality of reconstruction both quantitatively and qualitatively. 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. Since the entire scene consists of many buildings and cameras, visualizing the full network simultaneously is difficult. Figure 8 shows a subset of the distributed calibration result centered around a prominent church-like building in the scene (Figure 8a). To make it easier to perceive the reconstructed



(a) (b)

Fig. 8. Camera calibration results for a prominent building in the scene. (a) Original image 2, with detected feature points overlaid. (b) The 3D reconstruction of the corresponding scene points and several cameras obtained by the distributed calibration algorithm, seen from an overhead view, with building shape manually overlaid. Parallel and perpendicular building faces can be seen to be correct. Focal lengths have been exaggerated to show camera viewing angles. This is only a subset of the entire calibrated camera network.

structure, in Figure 8b we manually overlay a building outline from above to indicate the accurate position of a subset of the estimated points on the 3D structure. For example, the roof lines can be seen to be parallel to each other and perpendicular to the front and back wall. While this result was generated for visualization by registering each camera's structure to the same frame, each camera really only knows its location relative to its neighbors and reconstructed scene points.

VII. Conclusions And Future Work

We presented a new framework to determine image relationships in large networks of cameras where communication between cameras is constrained, as would be realistic in any wireless network setting. This is not a pure computer vision problem, but requires attention to and analysis of the underlying communication constraints to make the vision algorithm's implementation viable. We presented algorithms for each camera node to autonomously select a set of distinctive features in its image, compress them into a compact, fixed-length message, and establish a vision graph edge with another node upon receipt

of such a message. The ROC-curve analysis gives insight into how the number of features and amount of compression should be traded off to achieve desired levels of performance. We also showed how a distributed algorithm that passes messages along vision graph edges could be used to recover 3D structure and camera positions. Since many computer vision algorithms are currently not well-suited to decentralized, power-constrained implementations, there is potential for much further research in this area.

Our results made the assumption that the sensor 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, 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 its new feature set to the entire network every time it moves. However, it is undesirable that subtle motion should flood the camera network with broadcast messages, since the cameras could be moving frequently. While the information about each camera's motion needs to percolate through the entire network, only the region of the image that has changed would need to be broadcast to the network at large. In the case of gradual motion, the update message would be small and inexpensive to disseminate compared to an initialization broadcast. If the motion is severe, e.g. a camera is jolted so as to produce an entirely different perspective, the effect would be the same as if the camera had been newly initialized, since none of its vision graph links would be reliable. Hence, we imagine the transient broadcast messaging load on the network would be proportional to the magnitude of the camera dynamics.

It would also be interesting to combine the feature selection approach developed here with the training-data-based vector-quantization approach to feature clustering described by Sivic and Zissermann [35]. If the types of images expected to be captured during the deployment were known, the two techniques could be combined to cluster and select features that have been learned to be discriminative for the given environment.

Finally, it would be useful to devise a networking protocol well-suited to the correspondence application, which would depend on the MAC, network, and link-layer protocols, network organization, and channel conditions. Networking research on information dissemination [16], [17], node clustering [13], and node discovery/initialization [6] might be helpful to address this problem.

ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation, under the award IIS-0237516.

REFERENCES

[1] M. Andersson and D. Betsis. Point reconstruction from noisy images. *Journal of Mathematical Imaging and Vision*, 5:77–90, January 1995.

[2] S. Avidan, Y. Moses, and Y. Moses. Probabilistic multi-view correspondence in a distributed setting with no central server. In *Proceedings of the 8th European Conference on Computer Vision (ECCV)*, pages 428–441, 2004.

[3] A. Baumberg. Reliable feature matching across widely separated views. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 774–781, 2000.

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

[5] 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)*, pages 510–517, San Diego, CA., 2005.

[6] Z. Cai, M. Lu, and X. Wang. Distributed initialization algorithms for single-hop ad hoc networks with minislotted carrier sensing. *IEEE Transactions on Parallel and Distributed Systems*, 14(5):516–528, May 2003.

[7] S. Calderara, R. Vezzani, A. Prati, and R. Cucchiara. Entry edge of field of view for multi-camera tracking in distributed video surveillance. In *Proceedings of the IEEE International Conference on Advanced Video and Signal-Based Surveillance*, 2005.

[8] 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.

[9] D. Devarajan and R. Radke. Distributed metric calibration for large-scale camera networks. In *Proceedings of the First Workshop on Broadband Advanced Sensor Networks (BASENETS) 2004 (in conjunction with BroadNets 2004)*, San Jose, CA, October 2004.

[10] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley Interscience, 2000.

[11] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24:381–395, 1981.

[12] 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*, 3(3):209–226, September 1977.

[13] M. Gerla and J. Tsai. Multicluster, mobile, multimedia radio network. *Journal of Wireless Networks*, 1(3):255–265, 1995.

[14] C. Harris and M. Stephens. A combined corner and edge detector. In *Proceedings of the Fourth Alvey Vision Conference*, pages 147–151, Manchester, UK, 1988.

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

[16] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan. An application-specific protocol architecture for wireless microsensor networks. *IEEE Transaction on Wireless Communications*, 1(4):660–670, Oct. 2000.

[17] W. Heinzelman, J. Kulik, and H. Balakrishnan. Adaptive protocols for information dissemination in wireless sensor networks. In *Proceedings of the Fifth Annual ACM International Conference on Mobile Computing and Networking (MobiCom)*, pages 174–185, 1999. Seattle, Washington.

[18] D. F. Huber. Automatic 3D modeling using range images obtained from unknown viewpoints. In *Proceedings of the Third International Conference on 3D Digital Imaging and Modeling (3DIM)*, pages 153–160, May 2001.

[19] T. Kanade, P. Rander, and P. Narayanan. Virtualized reality: Constructing virtual worlds from real scenes. *IEEE Multimedia, Immersive Telepresence*, 4(1):34–47, January 1997.

[20] 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)*, pages 257–260, 2000. Barcelona, Spain.

[21] S. Khan and M. Shah. Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1355–1360, October 2003.

[22] T. Lindeberg. Detecting salient blob-like image structures and their scales with a scale-space primal sketch: a method for focus-of-attention. *International Journal of Computer Vision*, 11(3):283–318, 1994.

[23] D. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

[24] 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)*, pages 692–698, June 2004.

[25] K. Mikolajczyk and C. Schmid. Indexing based on scale invariant interest points. In *Procedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 525–531, Vancouver, Canada, 2001.

[26] K. Mikolajczyk and C. Schmid. Scale and affine invariant interest point detectors. *International Journal of Computer Vision*, 60(1):63–86, October 2004.

[27] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(10):1615–1630, 2005.

[28] 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)*, pages 90–95, 1998.

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

[30] 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)*, pages 103–119, 1998.

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

[32] C. Schmid and R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–534, 1997.

[33] 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)*, pages 587–597, Copenhagen, Denmark, 2002.

[34] C. Siva Ram Murthy and B. Manoj. *Ad Hoc Wireless Networks Architectures and Protocols*. Pearson PTR, 2004.

[35] J. Sivic and A. Zisserman. Video google: A text retrieval approach to object matching in videos. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1470–1477, 2003.

[36] I. Stamos and M. Leordeanu. Automated feature-based range registration of urban scenes of large scale. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 555–561, June 2003.

[37] P. Sturm and B. Triggs. A factorization based algorithm for multi-image projective structure and motion. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 709–720, 1996.

[38] S. Teller and M. Antone. Scalable extrinsic calibration of omni-directional image networks. *International Journal of Computer Vision*, 49(2):143–174, Sept.-Oct. 2002.

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