2008 Second ACM/IEEE International Conference on Distributed Smart Cameras

ONLINE DISTRIBUTED CALIBRATION OF A LARGE NETWORK OF WIRELESS CAMERAS USING DYNAMIC CLUSTERING

Henry Medeiros, Hidekazu Iwaki and Johnny Park

{hmedeiro, iwaki, jpark}@purdue.edu

ABSTRACT

We present a cluster-based distributed algorithm for calibrating large networks of wireless cameras. Due to the complex nature of sensing modality of a camera sensor, the work presented here differs significantly from the previous localization methods. Our system does not require any beacon nodes; it only utilizes object features of moving objects in the scene extracted from image sequences. The algorithm is fully distributed, and the localization estimates can be improved as more object features are acquired in the network. We show simulations of our system using a graphical simulator we developed specifically for wireless camera sensor networks. Early results indicate that our system is capable of localizing a large network of cameras in an energy-efficient way.

1. INTRODUCTION

Wireless sensor networks is an emerging technology that is envisioned to revolutionize the way how information can be gathered in different environments and how the information can be processed for a wide variety of applications. A typical sensor network consists of a large number of sensor nodes densely deployed in the field, each equipped with wireless communication, sensing and computing capabilities with a limited power resource. Each node gathers its surrounding information according to its sensing modality, and shares the data by communicating with other nodes in the network. Since the data collected by each node is location-specific, obtaining the locations of nodes is one of the fundamental problems in sensor networks. Consequently, there has been much effort in developing localization algorithms for wireless sensor networks (e.g., see [1, 2, 3, 4] and the references therein). These localization techniques, however, are not suited for camera sensors for two main reasons. First, the level of localization accuracy achieved is not sufficient for typical computer vision tasks. Even more importantly, these localization algorithms do not provide the orientation of a sensor, which is crucial for camera-based sensor networks. Although the existing localization algorithms may be used to obtain approximate positions of camera sensors, in order to obtain the precise positions and orientations of cameras that would be appropriate for basic computer vision tasks involving multiple cameras, an alternative calibration technique is needed.

Another challenging issue in calibrating a large network of wireless cameras is that the algorithm must take into account that each camera node has limited power resource. This implies that the energy efficiency of the algorithm is as important as the calibration accuracy – an entirely non-existent problem in wired camera network settings.

In this paper, we introduce four different approaches to the calibration of wireless camera networks, namely a centralized approach, a peer-to-peer offline approach, a peer-to-peer online approach, and finally a cluster-based online approach. We use our simulation system we designed specifically for wireless camera networks to analyze different tradeoffs between calibration accuracy, the amount of local memory used, and the number messages transmitted and received under various network settings. Finally, we show that our proposed cluster-based online calibration algorithm achieves calibration accuracy comparable to the centralized approach while the amount of local memory used in each camera node and the number of messages exchanged in the network are significantly less than the other calibration approaches.

2. PREVIOUS WORK

Calibration of multiple cameras is a well-studied problem in the computer vision community [5, 6, 7]. The majority of the works, however, does not take into account the issue of scalability and energy-efficiency where typically under 10, at most under 50 cameras, only have been considered. Thus, the calibration is typically performed by a single processor that collects the required image features of all the cameras, then performs a minimization of the calibration errors for the entire camera network.

Recently, however, there has been some work in developing localization algorithms specifically for camera-based sensor networks. Lee and Aghajan [8] proposed a localization technique that can estimate both the locations of a moving target and camera nodes. The method requires selecting two reference nodes to define the origin and the unit length, and the algorithm finds the positions and orientations of the other sensors with respect to the reference nodes. The technique was tested only in 2D space and only 5 cameras were

978-1-4244-2665-2/08/$25.00

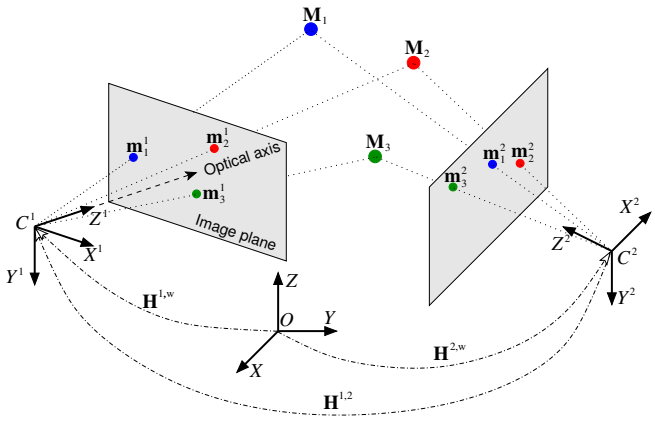


Fig. 1. Camera projection model

used for testing the algorithm. Mantzel *et al.* [9] proposed a distributed localization algorithm for camera sensors that iteratively refines the localization estimates. The algorithm assumes that image features required for localization have already been acquired and that the correspondences between the image features are known, thus the method cannot be applied in dynamic networks. Liu *et al.* [10] presented an automated calibration protocol for camera-based sensor networks. The proposed method utilizes a calibration device that is equipped with GPS and an LED. Using the global coordinates given by the calibration device and the image coordinates of the LED recorded in the camera, the camera's position and orientation can be computed. The main drawback of this approach is that the calibration device needs to be presented in front of each camera, which requires manual human assistance. Devarajan and Radke [11] proposed a distributed calibration algorithm for large camera networks. This work is similar to the distributed offline approach described in this paper, but the main difference is that the system assumes a message can be exchanged between any pairs of cameras. Finally, Funiak *et al.* [12] addressed the camera network calibration task by solving a simultaneous localization and tracking problem that estimates both the trajectory of the target object and poses of the cameras at the same time. However, the system assumes that all the cameras are ceiling mounted and considers only three extrinsic parameters (x, y, θ) of the camera.

3. BACKGROUND

In this section, we present some background material that is important to understand the work presented in this paper. We first describe the basic principles of camera projection model. Then we discuss how a pair of cameras can be related to each other through appropriate coordinate transformation.

3.1. Perspective Camera Model

Consider the image shown in Fig. 1. Let $OXYZ$ define the world coordinate system and $C^iX^iY^iZ^i$ the coordinate system of the i-th camera. Let $\mathbf{M}_j^w = [x_j^w, y_j^w, z_j^w, 1]^T$ be the homogeneous coordinate vector of the j-th object point with respect to the world coordinate system, and $\mathbf{m}_j^i = [u_j^i, v_j^i, 1]^T$ be the homogeneous image coordinate vector that represents the j-th object measured in the i-th camera. The basic image projection equation states that $\mathbf{m}_j^i$ is the projection of $\mathbf{M}_j^w$ up to an unknown scale factor s:

$$s\mathbf{m}_j^i = \mathbf{P}^i\mathbf{M}_j^w \tag{1}$$

where $\mathbf{P}^i$ is the 3×4 projection matrix of the i-th camera. We can rewrite Eq. (1) by further decomposing the projection matrix $\mathbf{P}^i$:

$$s\mathbf{m}_j^i = \mathbf{A}^i\left[\mathbf{R}^{i,w} \mid \mathbf{t}^{i,w}\right]\mathbf{M}_j^w \tag{2}$$

where $\mathbf{A}^i$ is a 3×3 upper triangular matrix that encodes the intrinsic parameters of camera i (e.g., focal length, principal point, etc.), and $\mathbf{R}^{i,w}$ and $\mathbf{t}^{i,w}$ are, respectively, a 3×3 rotation matrix and a translation vector that describe the 3D displacement from the world coordinate system to the i-th camera coordinate system.

A rotation matrix can be represented more compactly by a unit quaternion $\mathbf{q} = [q_w, q_x, q_y, q_z]^T$ where $q_w = \sqrt{1 - q_x^2 - q_y^2 - q_z^2}$. A rotation matrix $\mathbf{R}$ corresponding to a unit quaternion $\mathbf{q}$ is given by:

$$\mathbf{R}(\mathbf{q}) = \begin{bmatrix} 1-2q_y^2-2q_z^2 & 2(q_xq_y - q_zq_w) & 2(q_xq_z + q_yq_w) \\ 2(q_xq_y + q_zq_w) & 1-2q_x^2-2q_z^2 & 2(q_yq_z - q_xq_w) \\ 2(q_xq_z - q_yq_w) & 2(q_yq_z + q_xq_w) & 1-2q_x^2-2q_y^2 \end{bmatrix}$$

We will denote the extrinsic calibration parameters of a camera i as $\mathbf{p}^{w,i} = \begin{bmatrix} \mathbf{q}^{w,i} \\ \mathbf{t}^{w,i} \end{bmatrix}$ where $\mathbf{q}^{w,i}$ and $\mathbf{t}^{w,i}$ are, respectively, a quaternion vector and a translation vector that describe the transformation from the i-th camera coordinate system to the world coordinate system. Similarly, the relative transformation parameters from the j-th camera to the i-th camera is denoted as $\mathbf{p}^{i,j}$. A 4×4 homogeneous transformation matrix $\mathbf{H}$ corresponding to a calibration parameter vector $\mathbf{p}^{w,i}$ is given by:

$$\mathbf{H}(\mathbf{p}^{w,i}) = \begin{bmatrix} \mathbf{R}(\mathbf{q}^{w,i}) & \mathbf{t}^{w,i} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3}$$

Finally, for each camera parameter, say $\mathbf{p}^{i,j}$, we denote its corresponding covariance matrix as $\mathbf{\Sigma}^{i,j}$.

3.2. Pairwise Calibration

Finding the relative position and orientation between a pair of cameras is a well-studied problem in the computer vision community. The approach that fits well with our distributed

online algorithm is known as the estimation of *motion and structure*. Suppose there are n object points that are visible by two cameras 1 and 2. The *fundamental matrix* $\mathbf{F}$ between the two cameras is defined by the equation:

$$(\mathbf{m}_j^2)^T \mathbf{F} \mathbf{m}_j^1 = 0,\ j = [1, n] \tag{4}$$

Given at least eight matching image coordinates, the unknown fundamental matrix $\mathbf{F}$ can be solved [13]. The fundamental matrix along with each camera's intrinsic parameters (i.e., matrix $\mathbf{A}$ in Eq. (2)) can then be used to compute the relative position and orientation between the two cameras up to an unknown scale [14]. The advantage of the motion and structure method is that the 3D coordinates of object points do not need to be known — only corresponding image coordinates from two cameras. This indeed is an attractive feature that eliminates the need of global landmarks with known 3D locations or placing calibration objects at specific positions. The disadvantage is that the scale is arbitrary.

4. DISTRIBUTED CALIBRATION OF LARGE WIRELESS CAMERA NETWORKS

In this section we will describe two distributed algorithms for calibrating large wireless camera networks. We will first describe a peer-to-peer network approach to the distributed calibration, followed by a distributed calibration based on dynamic clustering of the camera nodes.

Before we dive into the algorithm description, let us lay out some notations and assumptions. Let $\mathcal{N}$ be the set of all camera nodes in the network. For each camera $i \in \mathcal{N}$, we define $\mathcal{S}^i$ to be a set of neighbors to which camera i can communicate directly (i.e., one-hop neighbor), and $\mathcal{V}^i$ to be a subset of these one-hop neighbors that also have overlapping viewing volume with camera i. We will call $\mathcal{S}^i$ as *communication neighbors* of camera i, and $\mathcal{V}^i$ as *vision neighbors* of camera i. Note that $\mathcal{V}^i \subseteq \mathcal{S}^i \subset \mathcal{N}$. For each camera i, its position and the orientation with respect to the world coordinate system is given by the camera parameter vector $\mathbf{p}^{w,i}$, and the relative position and orientation to a neighbor $k \in \mathcal{V}^i$ is given by $\mathbf{p}^{i,k}$. The goal here is to estimate a set of calibration parameters $\mathcal{P}^i = \left\{\mathbf{p}^{w,i}, \mathbf{p}^{i,k} \mid k \in \mathcal{V}^i\right\}$ for all $i \in \mathcal{N}$.

We assume that each camera in the network knows its own intrinsic camera parameters (e.g., focal length, principal points, distortion parameters, etc.). Although it is possible to estimate the intrinsic parameters using corresponding image points between cameras [15] and it would be possible to easily add this capability in the proposed system, we focus in this paper only on the estimation of the extrinsic camera parameters.

We also assume that the camera network is given the knowledge of the target objects used in the calibration. Specifically, we assume that each target object consists of multiple (at least two) distinctive features where the distance between the feature points are known. The distance information can be used to compute the correct scale in each pairwise calibration. In practice, we can devise an object attached with, for example, a blue LED and a red LED with a known distance between them.

4.1. Peer-To-Peer Distributed Calibration

A natural way to realize a distributed calibration of a large wireless camera network is to take a peer-to-peer network approach. We mean peer-to-peer in the sense that each camera node is responsible for collecting local measurements, sharing the measurements with its peers (i.t., communication neighbors), and computing the relative position and orientation of the neighbors. There are three main stages involved in distributed calibration: data collection, local calibration, and recursive estimation of calibration parameters.

4.1.1. Data Collection

Once a camera node is deployed and activated, it begins to search for moving objects in the scene. When a camera node detects a known target object in an image, a measurement tuple z^i of the target object is constructed. Without loss of generality, we will assume that all target objects have *two* distinctive features with a known distance between them. We can then define $\mathcal{Z}^i$ to be a set of measurement tuples obtained by camera i:

$$\mathcal{Z}^i = \left\{z_j^i \mid z_j^i = (\mathbf{m}_j^i, \mathbf{d}_j^i, t_j^i)\right\} \tag{5}$$

where z_j^i denotes the j-th measurement tuple, $\mathbf{m}_j^i = [u_1, v_1, u_2, v_2]^T$ is a measurement vector that contains the image coordinates of the two target features, $\mathbf{d}_j^i$ a target description vector that uniquely identifies the target object, and finally t_j^i the time at which the image was captured. Each time a measurement is generated, a message containing the acquired measurement is broadcast. Therefore, each camera in the network not only accumulates its local measurements $\mathcal{Z}^i$, but also the measurements $\mathcal{Z}^k$ received by its neighbors $k \in \mathcal{S}^i$.

4.1.2. Local Calibration

Assuming that each moving object can be identified by its description vector and that the image acquisition in the camera network is time-synchronized, two measurement tuples z_j^i and z_l^k acquired from two different cameras are corresponding measurements if $(\mathbf{d}_j^i, t_j^i) = (\mathbf{d}_l^k, t_l^k)$. A set of corresponding measurements between camera i and camera k is then defined as:

$$\mathcal{C}^{i,k} = \left\{(z_j^i, z_l^k) \mid z_j^i \in \mathcal{Z}^i, z_l^k \in \mathcal{Z}^k, (\mathbf{d}_j^i, t_j^i) = (\mathbf{d}_l^k, t_l^k)\right\} \tag{6}$$

When a sufficient number of measurements are accumulated, the number of corresponding measurements between

local measurements and each neighbor's measurements are computed. Then, for each neighbor with at least eight correspondences, a pairwise calibration can be performed to compute the relative position and orientation of the neighbor. Recall from the previous section that eight or more corresponding image points between a pair of cameras are required to compute the relative calibration parameters between two cameras up to an unknown scale. However, since we know the distance between two feature points of a target object, the correct scale for each pairwise calibration can be computed as follows. First, we compute the relative calibration parameters $\widehat{\mathbf{p}}^{i,k} = \left[\begin{array}{cc} (\mathbf{q}^{i,k})^T & (\widehat{\mathbf{t}}^{i,k})^T \end{array} \right]^T$ using a set of corresponding image coordinates in $\mathcal{C}^{i,k}$. Note the hat $\widehat{\ }$ implies an unknown scale. Then, based on these parameters, it is possible to obtain by stereo triangulation the 3D coordinates of each of the corresponding measurements in $\mathcal{C}^{i,k}$ at an arbitrary but consistent scale. Let $\widehat{\mathbf{M}_{\mathbf{1}j}}$ and $\widehat{\mathbf{M}_{\mathbf{2}j}}$ be the reconstructed 3D coordinates of the two target features corresponding to the j-th element of $\mathcal{C}^{i,k}$. Then, since the distance d between the two target features is known, it is possible to compute the scale factor s using:

$$s = \frac{1}{d} \left(\frac{1}{|\mathcal{C}^{i,k}|} \sum_{j=1}^{|\mathcal{C}^{i,k}|} \left\| \widehat{\mathbf{M}_{\mathbf{1}j}} - \widehat{\mathbf{M}_{\mathbf{2}j}} \right\| \right) \tag{7}$$

Then, the scale-corrected relative calibration parameter vector and the 3D coordinates are:

$$\mathbf{p}^{i,k} = \left[\begin{array}{c} \mathbf{q}^{i,k} \\ \left(\frac{1}{s}\right) \widehat{\mathbf{t}}^{i,k} \end{array} \right] \text{ and } \left[\begin{array}{c} \mathbf{M}_{\mathbf{1}j} \\ \mathbf{M}_{\mathbf{2}j} \end{array} \right] = \left(\frac{1}{s}\right) \left[\begin{array}{c} \widehat{\mathbf{M}_{\mathbf{1}j}} \\ \widehat{\mathbf{M}_{\mathbf{2}j}} \end{array} \right]$$

After each pairwise calibration, the covariance $\Sigma^{i,k}$ corresponding to the calibration parameter vector $\mathbf{p}^{i,k}$ can be estimated as follows. Each observation $\mathbf{m}_j^k$ by camera k can be mapped to a corresponding observation $\mathbf{m}_j^i$ in camera i by a function $\mathbf{m}_j^i = h^{i,k}(\mathbf{p}^{i,k}, \mathbf{m}_j^k)$. Now let the vector $\mathbf{x}^{i,k}$ be the stacked vector of n corresponding observations in cameras i and k, that is, $\mathbf{x}^{i,k} = (\mathbf{m}_1^k, ..., \mathbf{m}_n^k, \mathbf{m}_1^i, ..., \mathbf{m}_n^i)$. The stacked observation vector relates to the camera parameters by the function $\mathbf{x}^{i,k} = f^{i,k}(\mathbf{p}^{i,k}, \mathbf{m}_1^i, ..., \mathbf{m}_n^i)$. Then, the estimated observations and, consequently, the parameter vector $\mathbf{p}^{i,k}$ can be estimated based on these observations using an optimal criterion such as maximum likelihood assuming that the observations are perturbed by Gaussian noise. In this case, the covariance matrix of the vector $\left[\mathbf{p}^{i,k}, \mathbf{m}_1^k, ..., \mathbf{m}_n^k\right]$, that is, the arguments of $f^{i,k}(\cdot)$ is given by [16]:

$$\Sigma_{p,m}^{i,k} = \left[\begin{array}{cc} A^T \Sigma_x^{-1} A & A^T \Sigma_x^{-1} B \\ B^T \Sigma_x^{-1} A & B^T \Sigma_x^{-1} B \end{array} \right]^{\dagger} \tag{8}$$

where $\dagger$ denotes the pseudo-inverse, $A = \frac{\partial f^{i,k}(\cdot)}{\partial \mathbf{p}}$, $B = \frac{\partial f^{i,k}(\cdot)}{\partial \mathbf{m}_1^i, ..., \mathbf{m}_n^i}$, and Σ_x is the covariance matrix of $\mathbf{x}^{i,k}$. The top-left block of the matrix in Eq. (8) corresponds to the covariance of the relative position parameters $\Sigma^{i,k}$. As we will explain shortly in the next section, the calibration estimate vector $\mathbf{p}^{i,k}$ and its corresponding covariance matrix $\Sigma^{i,k}$ can be updated recursively if prior estimates are available.

Once we obtain approximate calibration parameters $\mathbf{p}^{i,k}$ for all neighbors $k \in \mathcal{V}^i$ as described above, one could carry out a nonlinear optimization scheme such as bundle adjustment [17] in order to optimize the camera parameters over all available data. Let h be a function that describes the reprojection of 3D coordinates $\mathbf{M}$ onto the image plane of a camera with calibration parameter $\mathbf{p}$, i.e., $\mathbf{m} = h(\mathbf{p}, \mathbf{M})$. The bundle adjustment essentially minimizes the reprojection error with respect to all 3D points and camera parameters, specifically:

$$\min_{\mathbf{p}^{i,k}, \mathbf{M}_j} \sum_{k=1}^{|\mathcal{V}^i|} \sum_{j=1}^{|\mathcal{C}^{i,k}|} \left\| h(\mathbf{p}^{i,k}, \mathbf{M}_j) - \mathbf{m}_j^k \right\| \tag{9}$$

by iteratively solving the weighted normal equations:

$$\mathbf{J}^T \boldsymbol{\Sigma}_{\mathbf{m}}^{-1} \mathbf{J} \delta = \mathbf{J}^T \boldsymbol{\Sigma}_{\mathbf{m}}^{-1} \epsilon \tag{10}$$

where $\mathbf{J}$ is the Jacobian matrix of h, $\boldsymbol{\Sigma}_{\mathbf{m}}$ the covariance matrix of the measurement vector, δ the sought update of the parameter vector in each iteration, and finally ϵ the reprojection error vector. The inverse covariance matrix of the calibration parameters can be obtained from the corresponding diagonal term of the matrix $\mathbf{J}^T \boldsymbol{\Sigma}_{\mathbf{m}}^{-1} \mathbf{J}$ at the end of the iteration. Although efficient implementation schemes for this large nonlinear optimization problem are available [18], it may still be computationally too expensive for the current generation of smart wireless cameras. Thus we will treat the bundle adjustment process as an optional procedure.

Algorithm 1 summarizes the local calibration procedure. Each camera i computes the relative position parameters and corresponding covariance matrix $(\mathbf{p}^{i,k}, \Sigma^{i,k})$ with respect to each of its neighbors k for which it has enough corresponding points, that is $\left|\mathcal{C}^{i,k}\right| > 8$. These initial parameter estimates are added to the set $\mathcal{P}^i$. When new estimates are computed, they are added to the temporary set $\mathcal{P}'$ so that they can be used to further refine the initial estimates via recursive estimation of camera parameters, described in the next section.

4.1.3. Recursive Estimation of Calibration Parameters

A local calibration at a camera node provides the calibration parameters for each of its neighbors. Since the local calibration is carried out independently at each node, two neighboring cameras i and k, where $i \in \mathcal{V}^k$ and $k \in \mathcal{V}^i$, each has its own estimate of the other, i.e., $\mathbf{p}^{i,k}$ computed at Camera i and $\mathbf{p}^{k,i}$ computed at Camera k. In an ideal case, they should be exactly the inverse transformation of each other, i.e., $\mathbf{H}(\mathbf{p}^{i,k}) = \mathbf{H}(\mathbf{p}^{k,i})^{-1}$, but in general this will not be the case due to measurement noise. Therefore, it is necessary to

Algorithm 1 Local calibration procedure.

```
Local-Calibration(i)
P' = ∅
for each k ∈ V^i
  Find C^{i,k} of Eq. (6)
  if (|C^{i,k}| > 8)
    Compute (p^{i,k}, Σ^{i,k})
    if (p^{i,k}, Σ^{i,k}) ∉ P^i
      add (p^{i,k}, Σ^{i,k}) to P^i
    else
      add (p^{i,k}, Σ^{i,k}) to P'
    end if
  end if
end for
if P' ≠ ∅
  Recursive-Estimation(i, P')
end if
// optional
Refine P^i by Bundle Adjustment
Remove used measurements
```

Algorithm 2 Recursive estimation of camera parameters.

```
Recursive-Estimation(senderID, P')
if (senderID = i)  //local update
  for each ((p'^{i,k}, Σ'^{i,k}) ∈ P')
    Update (p^{i,k}, Σ^{i,k}) using Eqs. (11-13)
  end for
else //received parameters from a neighbor
  for each (p'^{k,i}, Σ'^{k,i}) ∈ P'
    Compute (p'^{i,k}, Σ'^{i,k}) from H(p'^{k,i})^{-1}
    Update (p^{i,k}, Σ^{i,k}) using Eqs. (11-13)
  end for
  if (senderID < i)
    H^{w,i} = H(p'^{w,k})H(p'^{k,i})
    Compute (p^{w,i}, Σ^{w,i}) from H^{w,i}
      and replace it in P^i
  end if
end if
Broadcast m1 = (i, P^i)
```

integrate the estimates in the two cameras to obtain a more accurate estimate.

The need for integrating calibration estimates also arises in an *online* calibration approach where cameras in the network compute new calibration estimates as more measurements are acquired locally and more measurements are received from its neighbors throughout the calibration process. We employ the Weighted Recursive Least Squares technique to integrate a previous estimate $(\mathbf{p}(t-1), \Sigma(t-1))$ with a new estimate $(\mathbf{p}^{'}(t), \Sigma^{'}(t))$ into a more accurate estimate $(\mathbf{p}(t), \Sigma(t))$ by the following recursions:

$$K = \Sigma(t-1)\left(\Sigma(t-1) + \Sigma^{'}(t)\right)^{-1} \quad (11)$$

$$\mathbf{p}(t) = \mathbf{p}(t-1) + K\left(\mathbf{p}^{'}(t) - \mathbf{p}(t-1)\right) \quad (12)$$

$$\Sigma(t) = \Sigma(t-1) - K\Sigma(t-1) \quad (13)$$

It is important to note that, under Gaussian noise assumption, $\mathbf{p}(t)$ in Eq. (12) corresponds to the best unbiased estimator of the real camera parameters. The recursive update of calibration parameters is repeated whenever new calibration estimates from another local calibration are available or calibration estimates sent by a neighbor are received.

The procedures described so far provide each camera only the relative positions and orientations of its neighbors. In order to relate any locational information defined in a local coordinate system to the rest of the nodes in the network, it is inevitable to obtain a common coordinate system that is globally consistent throughout the network. One approach to obtain a globally common coordinate system is to use *reference index* [19]. Each node initially sets its reference index, denoted as w, to its own node ID. That is, $w = i$, thus $\mathbf{p}^{w,i} = 0$ ($\mathbf{H}(\mathbf{p}^{w,i}) = I$). Then, whenever a camera receives calibration parameters of a neighbor with a lower reference index, the camera changes the reference index and transforms the reference coordinate system to the coordinate system of the neighbor. In the end, the reference coordinate systems of all the nodes in the network will be in the coordinate system of the node with the lowest ID.

The complete procedure for the recursive estimation of camera parameters is shown in Algorithm 2. Figure 2 shows an example of one step of the recursive estimation of camera parameters in a simple network of 3 cameras. The left figure shows the status of the network at a given moment when camera 1 has already computed $\mathbf{p}^{0,1}$, its relative position parameters with respect to camera 0, and camera 2 has already computed $\mathbf{p}^{1,2}$, its relative position parameters with respect to camera 1. At that time, camera 1 has also defined its relative position to the origin to $\mathbf{p}^{w,1} = \mathbf{p}^{0,1}$. As camera 1 receives new measurements from camera 2, it computes $\mathbf{p}^{2,1}$, its relative position parameters with respect to camera 2 and broadcasts these new parameters along with its own relative position with respect to the origin $\mathbf{p}^{w,1}$. After camera 2 receives $\mathbf{p}^{2,1}$ and $\mathbf{p}^{w,1}$ from camera 1, it updates its own estimate $\mathbf{p}^{1,2}$ based on the inverse transformation $\mathbf{H}(\mathbf{p}^{2,1})^{-1}$ and Eqs. (11-12). After that, camera 2 verifies that the node ID of the sender is lower than its own and that it should replace its own global reference by that sent by camera 1. Therefore, it replaces its own global position parameters vector $\mathbf{p}^{w,2}$ by the parameter vector corresponding to the homogeneous transformation $\mathbf{H}(\mathbf{p}^{w,1})\mathbf{H}(\mathbf{p}^{1,2})$, and broadcasts its own updated estimates to its neighbors. The left figure in Figure 2 shows the status of the network at that moment.

Algorithm 3 shows the complete the peer-to- peer dis-

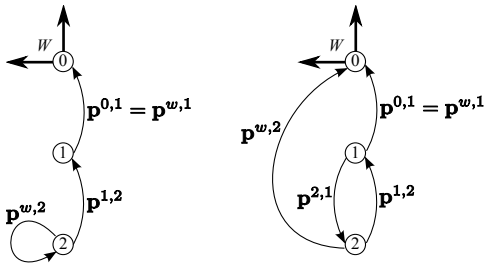


Fig. 2. Recursive update of camera parameters.

Algorithm 3 Peer-to-peer distributed calibration.

```
Peer-to-Peer-Calibration
while (1)
  if (new measurement available)
    Generate z_j^i = (m_j^i, d_j^i, t_j^i)
    Add z_j^i to Z^i
    Broadcast m2 = (i, z_j^i)
  end if
  if (Received m2 = (i_m2, z_m2))
    Add z_m2 to Z^{i_m2}
  end if
  if (enough measurements
        in Z^i and Z^{i_m2})
    Local_Calibration(i)
  end if
end while
```

tributed calibration procedure.

4.2. Cluster-based Distributed Calibration

The main drawback of the peer-to-peer calibration approach is that each node has to constantly listen for messages from its neighbors and compute its relative position with respect to them. As a consequence, the same measurements are received and processed by multiple nodes, thereby consuming unnecessary energy. To mitigate this problem, we propose to assign roles to the nodes using an event-driven clustering protocol. After a cluster is created, only the cluster head is responsible for receiving the measurements from the cluster members and computing their relative positions.

Our cluster-based calibration procedure is based on an event-driven clustering protocol that can be summarized as follows [20]. When an object with specific visual features is detected, cameras that detect this object create a cluster of cameras that can communicate in a single hop and elect a cluster head among themselves. If an object can be detected by cameras that cannot communicate in a single hop, multiple single-hop clusters are formed. If more than one object is detected, multiple clusters are formed based on the visual features of the objects. As the object moves, new cameras that detect it join the cluster, cameras that lose track of the target leave the cluster, and, when the cluster head loses track of the target, the role of cluster head is assigned to a different camera.

Algorithm 4 Cluster-based data collection.

```
Cluster_Based_Data_Collection()
while (cluster active)
  if (new measurement available)
    Generate z_j^i = (m_j^i, t_j^i)
    if (cluster head)
      Add z_j^i to Z^i
    else
      Send m2 = (i, z_j^i) to
        cluster head
    end if
  end if
  if (Received m2 = (i_m2, z_m2))
    Add z_m2 to Z^{i_m2}
  end if
end while
```

As the peer-to-peer approach, the cluster-based calibration procedure also consists of three main stages: data collection, local calibration, and recursive estimation of calibration parameters. Only data collection is carried out while the cluster is active. After the cluster head leaves the cluster and chooses another camera to become the new cluster head, local calibration takes place, and the former cluster head computes its relative positions with respect to its cluster members and broadcasts this information. Finally, the positions of the cameras with respect to a global reference are computed and their mutual relative position parameters are aggregated in the unification of calibration parameters stage.

During data collection, all the cameras that belong to a cluster acquire information about the target. That is, after a cluster is created, the members of the cluster as well as the cluster head compute the image coordinates $\mathbf{m}_j^i$ of the target. The cluster members transmit the corresponding measurement tuples $z_j^i = (\mathbf{m}_j^i, i)$ to the cluster head. Since the cluster is formed based on the target features, it is not necessary to include the target description vector $\mathbf{d}_j^i$ in the measurement tuples. The cluster head stores these measurements as well as the measurements acquired by itself until the target leaves its field of view and the role of cluster head is assigned to a different node. Algorithm 4 shows the data collection stage of the cluster-based calibration procedure.

After the cluster head leaves the cluster, it uses the measurements it acquired and the corresponding measurements received from the cluster members to compute its relative position parameters $\mathbf{p}^{i,k}$ with respect to each of its cluster members according to the local calibration procedure (Alg. 1) described in section 4.1.2. New estimates, estimates com-

Algorithm 5 Cluster-based camera calibration.

```
while (1)
  Cluster_Based_Data_Collection()
  if (enough measurements
        in Z^i and Z^{i_m1})
    Local_Calibration(i)
  end if
end while
```

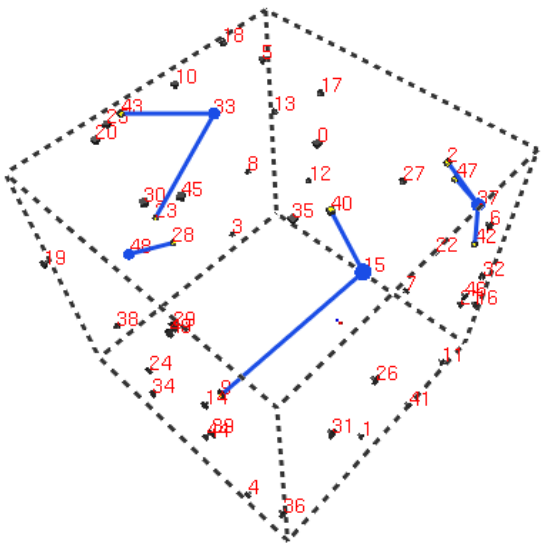


Fig. 4. Active clusters in the test environment.

puted by neighboring cameras, and the global reference are integrated according to the recursive update of calibration parameters described in section 4.1.3. Algorithm 5 presents the complete cluster-based calibration procedure.

5. EXPERIMENTS

We carried out simulations of our algorithm using a graphical simulation tool that we developed specifically for wireless camera sensor networks. Figure 3 shows the environment we used to carry out our simulations. The environment consists of a cube volume with the dimension of $20 \times 20 \times 20$ meters with 50 cameras randomly placed on the four side planes and the top plane. All cameras view roughly the opposite direction of the plane they are on. The red edges in Fig. 3(a) indicate communication connectivity among cameras. Fig. 3(b) is the same setting as in (a), but showing the viewing volumes of the cameras. Figure 4 shows the same setting with four active clusters, where the cluster heads are represented by the blue circles, and the cluster members are the cameras connected to the cluster heads by blue edges. The cameras are calibrated through the observation of a single target that moves at random inside the cube volume.

We compared the performance of the two approaches we present in this paper to that of two alternative approaches that carry out offline calibration. That is, estimation of the parameters only takes place after all the data is collected. The first alternative approach consists of computing the camera parameters in a centralized manner. That is, all the data collected by the cameras is initially transmitted to a base station. After data collection, the base station computes the estimated parameters of all the cameras of the network based on all the measurements. In the second alternative approach, which we call distributed offline calibration, all the data is initially collected by the nodes in the network and then processed locally by each node using the local calibration procedure and recursive estimation of the camera parameters.

5.1. Calibration Accuracy

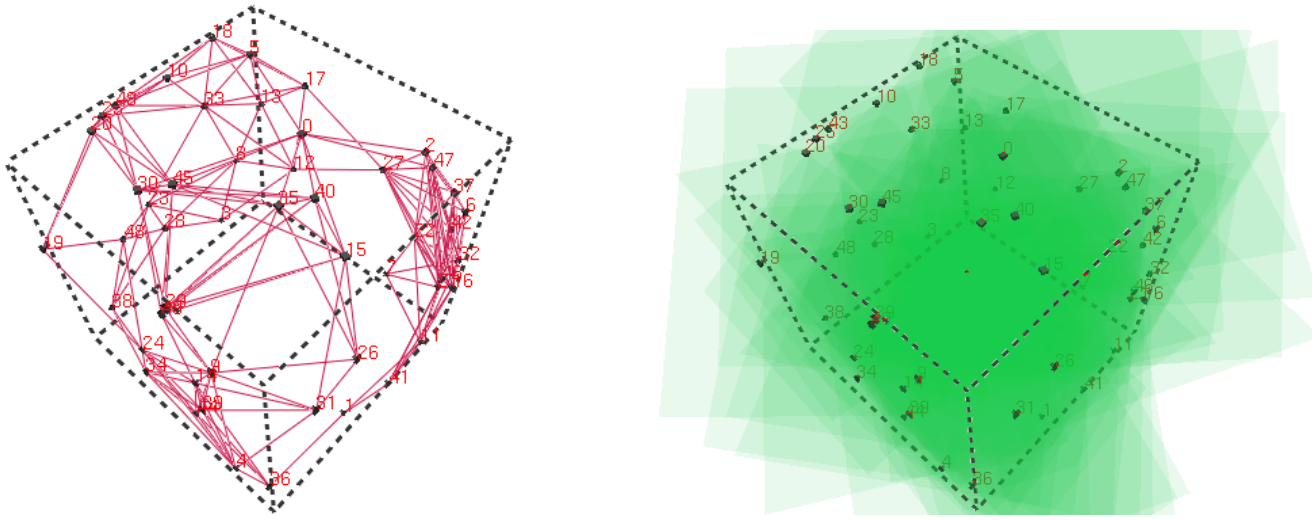
Evidently, the ultimate goal of our system is to accurately estimate the position of each camera in the network with respect to a global reference. Therefore, we measured the accuracy of the estimated global position of the cameras obtained by the four different approaches. Figures 5 and 6 show the root mean squared error with respect to the ground truth of the estimated translation $\mathbf{t}^{w,i}$ and rotation $\mathbf{R}(\mathbf{q}^{w,i})$ between each camera and the global reference as a function of simulation time. The figure shows that the total error decreases with time in all the approaches, i.e., as new measurements are acquired by the cameras and the estimated positions are updated, the error in the individual cameras decreases. The initial error increase corresponds to the moments when cameras that were not previously calibrated are initialized to a somewhat poor initial estimate. Eventually, all the cameras converge to the same average error. We can see that the accuracy achieved by the peer-to-peer approach and by the cluster-based approach are comparable to the accuracy obtained using the distributed offline or the centralized approaches. It is important to note that, in order to keep our implementation simple so that it could be ported to real wireless cameras, we have not employed the optional bundle adjustment step.

5.2. Number of Cameras Calibrated

Figure 7 shows the number of cameras calibrated as a function of simulation time for the cluster-based and the peer-to-peer calibration approaches. We can see that, since the cluster-based approach performs much less redundant computations, more simulation steps are necessary to calibrate all the cameras.

5.3. Number of Messages

We also measured the total number of messages exchanged by the cameras during calibration using the four approaches. To compare the number of messages exchanged by each of the



(a) (b)

Fig. 3. Test environment. 50 cameras placed on the side planes and on the top plane of a $20 \times 20 \times 20$ meters cube volume. **(a)** two nodes connected by an edge are able to communicate with each other; **(b)** The same setting as in (a), but showing viewing volumes of cameras.

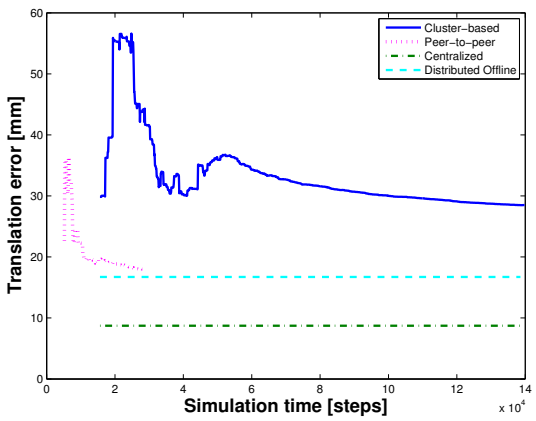


Fig. 5. Accuracy of the estimated translation of the cameras with respect to the global reference.

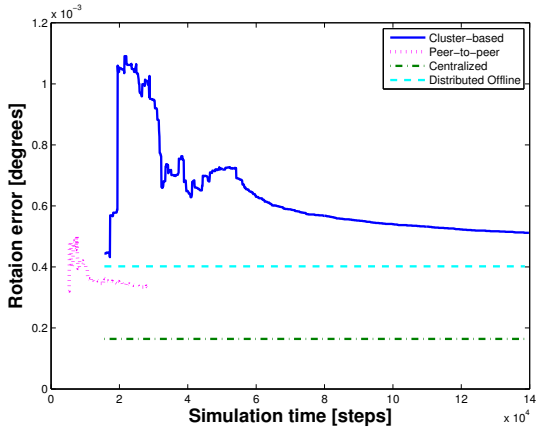


Fig. 6. Accuracy of the estimated rotation of the cameras with respect to the global reference.

approaches proposed, we moved the target using the same trajectory and performed the same number of simulation steps. The number of simulation steps was defined so that the average translation and rotation error would converge in all the proposed approaches. Figure 8 shows the average number of bits transmitted and received by the network after calibration was concluded. As we can see, in the centralized approach, since every measurement has to be transmitted to the base station, the number of messages exchanged is much higher than in the other three approaches. In the cluster-based approach, since the cluster members do not need to receive messages from their neighbors during data collection, they can turn off their radio receivers to avoid overhearing messages sent to the cluster head. As a consequence, there is a substantial reduction in the number of bits received compared to the peer-to-peer approach.

5.4. Memory Requirements

Since the amount of memory available in a wireless camera node is very constrained, for an algorithm to be feasible in practice it must require the storage of a limited amount of information. Figure 9 shows the maximum amount of memory used by each node of the network to store the acquired measurements in the cluster-based approach and the offline approaches. As expected, since in the centralized approach the

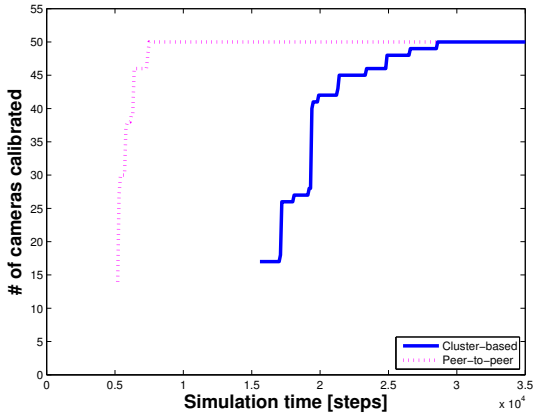


Fig. 7. Number of cameras calibrated with respect to the global reference.

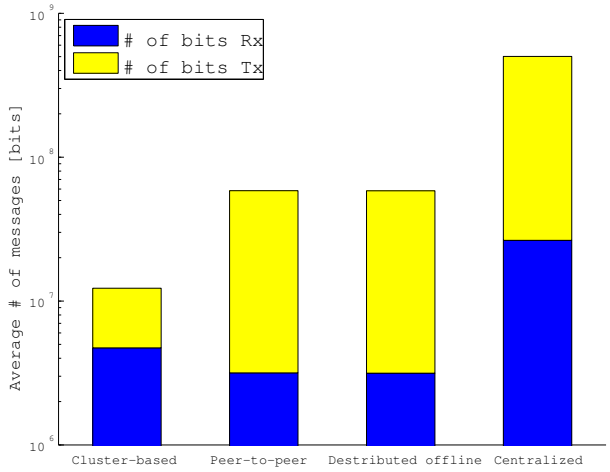


Fig. 8. Total number of messages received and transmitted.

base station must store all the measurements acquired by the network, it must have a very large amount of memory. In the distributed offline approach, since each node must store all the measurements acquired in its neighborhood during the entire process, each node should be provided a reasonable amount of memory. Although each node in the distributed offline approach requires less storage capacity than the base station in the centralized approach, we can see that the memory requirements of the cluster-based approach are much lower.

6. CONCLUSION AND FUTURE WORK

We presented a distributed online localization system specifically developed for a large network wireless cameras. Our system does not require any beacon nodes, but only utilizes object features of moving objects in the scene extracted from image sequences. The algorithm is fully distributed, and the

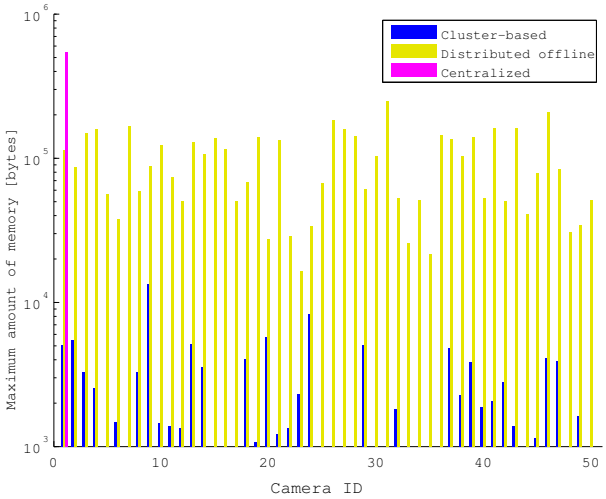


Fig. 9. Maximum amount of memory used by each node during calibration.

localization estimates can be improved as more object features are obtained. We showed simulations of our system using a graphical simulator we developed specifically for wireless camera sensor networks. Early results showed that our system is capable of localizing a large network of cameras accurately. The peer-to-peer approach, although more accurate than the cluster-based approach, requires that the nodes receive much more measurements from their neighbors, therefore spending much more energy.

We envision several directions for future work. First, the algorithm obviously needs to be tested in real wireless cameras such as Cyclops cameras [21]. Since we have employed only simple algorithms that do not require complex iterative solutions, we should be able to implement them efficiently in a real wireless camera. In fact, this project is already being carried out and the initial results are encouraging.

In addition to that, we plan on employing multiple objects in the calibration procedure since this should greatly reduce the time required to calibrate the entire network. In that case, we believe that the cluster-based approach should provide even greater energy savings. This is due to the fact that, since clusters are created to track a specific object, the features of the object do not need to be included in the measurement messages transmitted within a cluster. Suppose k objects are used to calibrate the network. In that case, if no clusters are formed, at least $\log_2 k$ additional bits are required in each message to identify the objects (assuming a very simplistic case in which a pair of objects can be distinguished by a single bit). Therefore, at least $8 \times log_2 k$ bits are required for each parameter estimation. However, clustering evidently imposes communication overhead, which is a function of the velocity at which the target moves and the camera arrangement. Therefore, it is important to make sure that, for a spe-

cific camera arrangement, the target speed is adjusted so that the total number of messages transmitted for cluster maintenance is smaller than the number of messages required to estimate the camera positions without employing clustering.

Another issue that must be explored in the future is the criterion for defining the global reference. Although the reference index approach is simple and effective, it may lead to much unnecessary communication in the network if the camera with lowest ID calibrates itself after a large number of cameras is already calibrated. We plan on using different approaches for defining the global reference that attempt to reduce unnecessary changes of reference and the consequent propagation of messages.

7. REFERENCES

[1] A. Savvides, C.-C. Han, and M. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Proc. International Conference on Mobile Computing and Networking*, pp. 166–179, 2001.

[2] C. Savarese, J. Rabaey, and J. Beutel, "Locationing in distributed ad-hoc wireless sensor networks," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 2037–2040, 2001.

[3] S. Čapkun, M. Hamdi, and J.-P. Hubaux, "GPS-free positioning in mobile ad hoc networks," *Cluster Computing*, vol. 5, no. 2, pp. 157–167, 2002.

[4] A. Galstyan, B. Krishnamachari, K. Lerman, and S. Pattem, "Distributed online localization in sensor networks using a moving target," in *Proc. International Symposium on Information Processing in Sensor Networks*, pp. 61–70, 2004.

[5] T. Svoboda, D. Martinec, and T. Pajdla, "A convenient multicamera self-calibration for virtual environments," *PRESENCE: Teleoperators and Virtual Environments*, vol. 14, pp. 407–422, August 2005.

[6] P. Baker and Y. Aloimonos, "Calibration of a multicamera network," in *Proc. Omnidirectional Vision and Camera Networks*, 2003.

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

[8] H. Lee and H. Aghajan, "Collaborative self-localization techniques for wireless image sensor networks," in *Proc. Asilomar Conference on Signals, Systems and Computers*, 2005.

[9] W. Mantzel, H. Choi, and R. Baraniuk, "Distributed camera network localization," in *Proc. Asilomar Conference on Signals, Systems and Computers*, 2004.

[10] X. Liu, P. Kulkarni, and P. Shenoy, "Snapshot: A Self-Calibration Protocol for Camera Sensor Networks," tech. rep., Department of Computer Science, University of Massachusetts, 2005.

[11] D. Devarajan and R. Radke, "Distributed metric calibration of large camera networks," in *Proc. International Conference on Broadband Networks*, 2004.

[12] S. Funiak, C. Guestrin, M. Paskin, and R. Sukthankar, "Distributed localization of networked cameras," in *International Conference on Information Processing in Sensor Networks (IPSN'06)*, pp. 34 – 42, 2006.

[13] R. Hartley, "In defense of the eight-point algorithm," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 6, pp. 580–593, 1997.

[14] J. Weng, T. Huang, and N. Ahuja, "Motion and structure from two perspective views: Algorithms, error analysis, and error estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 11, pp. 451–476, May 1989.

[15] M. Pollefeys, R. Koch, and L. V. Gool, "Self-calibration and metric reconstruction inspite of varying and unknown intrinsic camera parameters," *International Journal of Computer Vision*, vol. 32, no. 1, pp. 7–25, 1999.

[16] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[17] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, "Bundle Adjustment – A Modern Synthesis," in *Vision Algorithms: Theory and Practice*, Lecture Notes in Computer Science, pp. 153–177, Springer Berlin / Heidelberg, 2000.

[18] M. Lourakis and A. Argyros, "The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm," Tech. Rep. 340, Institute of Computer Science - FORTH, Heraklion, Crete, Greece, August 2004. Available from `http://www.ics.forth.gr/~lourakis/sba`.

[19] R. Iyengar and B. Sikdar, "Scalable and distributed GPS free positioning for sensor networks," in *Proc. IEEE International Conference on Communications*, pp. 338–342, 2003.

[20] H. Medeiros, J. Park, and A. C. Kak, "A Light-weight Event-driven Protocol for Sensor Clustering in Wireless Camera Networks," in *ACM/IEEE International Conference on Distributed Smart Cameras*, 2007.

[21] 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 *Proc. International Conference on Embedded Networked Sensor Systems*, pp. 192–204, 2005.