Distributed Calibration of Camera Sensor Networks

Ehsan Elhamifar
Center for Imaging Science, Clark Hall
Johns Hopkins University, Baltimore, MD 21218
Email: ehsan@cis.jhu.edu

René Vidal
Center for Imaging Science, Clark Hall
Johns Hopkins University, Baltimore, MD 21218
Email: rvidal@cis.jhu.edu

***Abstract*—We propose a distributed algorithm for calibrating the intrinsic and extrinsic parameters of a Camera Sensor Network (CSN). We assume that only one of the cameras is calibrated and that the network graph, *i.e.* the graph over which the cameras communicate, is connected. Each camera uses simple algorithms based on epipolar geometry to obtain its calibration matrix as well as its pose relative to a reference frame. A distributed consensus algorithm is derived to enforce a globally consistent solution for the recovered 3D structure across the entire network. We demonstrate the validity and effectiveness of our method through synthetic experiments.**

I. Introduction

Recent hardware innovations have produced low-power embedded computers (motes) equipped with small cameras that can self-organize into wireless networks. Such networks can be used in a variety of computer vision applications, such as 3D reconstruction of large environments, tracking of mobile targets, recognition of events and activities, etc.

In a camera sensor network, the cameras are often deployed in an unknown and unstructured 3D environment. Thus, an important pre-processing step for several computer vision applications is to compute the relative position and orientation of each camera with respect to a reference frame as well as the 3D structure of the observed scene. This problem, known as *structure from motion* (SfM), has been extensively studied in the computer vision community for the past few decades. However, most existing algorithms are not easily applicable to camera sensor networks, as we argue in the next paragraphs.

Prior work. The problem of camera calibration for a single camera has been widely studied in the computer vision literature [1], [2], [3], [4], [5]. Such methods typically require the user to show a known object (calibration rig) to the camera. The camera parameters are found by minimizing the reprojection error between the 2D projections of known 3D points in the object and their measured 2D projections in the image. Obviously, such semi-automatic methods do not scale well for large camera sensor networks, because they would require manual calibration of each camera. Moreover, since each camera would be calibrated independently from the others, the estimated poses may not be globally consistent.

On the other hand, several auto-calibration methods have been proposed [6], [7], [8], [9], [10], [5]. Such methods automatically match image points across several images and calibrate the cameras by solving nonlinear equations such as Kruppa's equations [11]. While these methods are very elegant, they suffer from the fact that the problem of solving Kruppa's equations is numerically ill-conditioned. Moreover, these methods assume that all the cameras have the same intrinsic parameters so they are not readily applicable to camera sensor networks, where each camera can have different intrinsic parameters.

Once the cameras' intrinsic and extrinsic parameters are known, the next problem is to recover the structure of the 3D scene. This *triangulation* problem can be solved in closed form from 2 views and requires nonlinear optimization in the case of multiple views [5]. However, solving the multiple view triangulation problem becomes difficult in a sensor network setup. This is because, while each pair of motes could easily compute the 3D structure of the scene via linear triangulation, the estimates from different pairs of motes may not be the same, especially in real situations where image data are noisy. Thus, there is a need for developing algorithms for integrating local SfM estimates in such a way that they converge to the global solution given by centralized SfM algorithms.

This has motivated several distributed localization algorithms in the camera sensor networks community such as [12], [13], [14], [15], [16], [17], [18], [19]. Semi-automatic camera network calibration methods use laser printed textures mounted on a board [12] or modulated LED emissions [13], [14]. Automatic methods perform local SfM via robust bundle adjustment [17] and integrate the information using belief propagation [18]. However, these algorithms integrate the information in an ad-hoc fashion and do not provide rigorous proofs of convergence to the centralized solution.

Paper contributions. This paper makes two important contributions. First, we show that the problem of automatically calibrating a large number of cameras in a sensor network can be solved in a distributed way by solving a set of linear equations. We show that this is possible under the mild assumption that only one of the cameras needs to be calibrated. Second, we propose a method based on distributed consensus algorithms [20], [21] for estimating the 3D structure of a scene in a distributed way. We show that all the motes compute the same 3D structure, even though they communicate with only a few neighbors in the network. This method requires that all the cameras observe the same scene and that the network graph over which the nodes communicate be connected.

Paper outline. The remainder of the paper is organized as follows. In Section II, we briefly review basic definitions

from graph theory and the consensus algorithms. Section III reviews the intrinsic and extrinsic calibration parameters of cameras and epipolar geometry methods for their estimation. In Section IV, we show how to recover the extrinsic and intrinsic calibration parameters of cameras in a sensor network by having only one calibrated camera. In Section V, we propose a method based on distributed consensus algorithm to get a common 3D structure in the network. Finally, in Section VI, we show the efficacy of our method using synthetic experiments.

II. Distributed Consensus Algorithms

We represent a network of n nodes by a graph $G = (V, E)$. The set $V = \{1, 2, \ldots, n\}$ represents the set of vertices in the graph and each $i \in V$ represents one node in the network. The set $E \subseteq V \times V$ represents the set of edges in the graph. An edge (i, j) belongs to E if nodes i and j can communicate with each other. We say that two nodes i and j are adjacent in G when there is an edge between them in the graph. The adjacency matrix of the graph $A \in \mathbb{R}^{n\times n}$ models this relation as $a_{ij} \neq 0$ when nodes i and j are adjacent and $a_{ij} = 0$ otherwise. The set of nodes adjacent to node i are represented by N_i. The degree of each node i denoted by d_i is the sum of the weights of the edges connected to node i, i.e. $d_i = \sum_{j\in N_i} a_{ij}$. We collect the degree of all nodes in a diagonal matrix $D = \text{diag}\{d_1, d_2, \cdots, d_n\}$. Finally, the Laplacian matrix of the graph is defined as $L = D - A$. One can show that when the graph is undirected, i.e., when the corresponding adjacency matrix is symmetric, the Laplacian matrix is always positive semi-definite and has a zero eigenvalue whose corresponding eigenvector has all the elements equal to 1, i.e., $L\,1 = 0$.

There are two main challenges in sensor networks. First, making a decision based on only few sensors' measurements is inherently unreliable. Thus one has to integrate information from all the sensors in order to make a good decision. Second, due to limited computational power and communication capabilities, each node cannot collect and store data from all the other nodes in the network in order to make a decision. Thus, there is a need for developing distributed algorithms in which every node gathers information only from its neighbors, yet all the nodes eventually reach the same decision.

There are several methods for integrating information across a sensor network. In this paper, we are particularly interested in distributed consensus algorithms. For the sake of simplicity, let us describe this method in the scalar case. That is, we assume that each node i measures a scalar quantity $x_i(0)$ (the argument would be similar for vectors and matrices). We say the nodes have reached consensus at time t when all the nodes have achieved a collective decision α, i.e.

$$x_1(t) = x_2(t) = \ldots = x_n(t) = \alpha. \tag{1}$$

There are a variety of distributed consensus algorithms ensuring the network will asymptotically reach consensus. The interested reader is referred to [21] for a comprehensive review of the algorithms. In this paper, we consider a popular iterative algorithm, each iteration of which has the following form:

$$x_i(t+1) = x_i(t) + \epsilon \sum_{j\in N_i} (x_j(t) - x_i(t)), \tag{2}$$

where $\epsilon < 1/\Delta_G$ and $\Delta_G = \max\{d_1, \ldots d_n\}$ is the maximum degree of the nodes. One can see from the above equation that, at every iteration, each node only gets information from its neighbors. It is shown in [21] that when the network graph is connected, all nodes will asymptotically reach the same value $\alpha = \frac{1}{n}\sum_{i=1}^{n} x_i(0)$. The speed of convergence is determined by the connectivity of the graph. More specifically, notice that one can write equation (2) in a compact form as:

$$x(t+1) = (I - \epsilon\, L)\, x(t) \tag{3}$$

where $x(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^\top$. The second smallest eigenvalue of the Laplacian L, known as the algebraic connectivity of the graph, will determine the speed of convergence of the algorithm. As the degree of each node increases, the second smallest eigenvalue will be larger and the convergence would be faster.

III. Epipolar Geometry and Camera Calibration

Assume we have N points $\{X^j \in \mathbb{R}^3\}_{j=1}^N$ in 3D space, where the vector X^j contains the coordinates of point j in the world reference frame. Let $(R, T) \in SO(3) \times \mathbb{R}^3$ be the pose of a camera, with respect to the world reference frame, where R is a rotation matrix belonging to $SO(3) = \{R \in \mathbb{R}^{3\times 3} : R^\top R = I, \det(R) = +1\}$ and T is the translation between the world and the camera frames. Then the projection of the point X^j onto the image plane of this camera has homogeneous coordinates $\tilde{x}^j \in \mathbb{R}^3$ satisfying the following equation:

$$\lambda^j \tilde{x}^j = KRX^j + KT. \tag{4}$$

In this equation, λ^j is the projective depth of a point j and is equal to the third coordinate of $KRX^j + KT$. The matrix $K \in \mathbb{R}^{3\times 3}$ is called the *intrinsic parameter matrix* or *camera calibration matrix* and is of the following form:

$$K = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{5}$$

The calibration matrix is constructed using the intrinsic parameters of the camera, namely, the position of the optical center (o_x, o_y), the size of the pixels (s_x, s_y), the skew factor s_θ and the focal length f. The rotation R and translation T describe the relative position and orientation of the camera frame with respect to the world reference frame. They are also called the *extrinsic calibration parameters* of the camera. Note that $\tilde{x}^j$ in equation (4) describes the image point in the pixel coordinates, while $x^j = K^{-1}\tilde{x}^j$ is the image point in the metric coordinates. The SfM problem refers to the problem of inferring both extrinsic and intrinsic parameters of the camera.

Now, assume that we have two cameras observing the same scene. Without loss of generality, we assume the world

reference frame is located at the center of the first camera, i.e., $(R_1, T_1) = (I, 0)$. The relation between the image points in the two cameras is then given as:

$$\lambda_2^j \tilde{x}_2^j = \lambda_1^j K_2 R K_1^{-1} \tilde{x}_1^j + K_2 T \tag{6}$$

where, for simplicity of notation, we used (R, T) rather than (R_2, T_2). One can eliminate the unknown scales λ_1^j and λ_2^j from (6) and get:

$$\tilde{x}_2^{j\top} F \tilde{x}_1^j = 0, \quad \text{where} \quad F = \widehat{K_2 T} K_2 R K_1^{-1}. \tag{7}$$

The matrix F is called the *fundamental matrix* between the two cameras and incorporates intrinsic and extrinsic calibration parameters. Here, $\widehat{u} \in so(3)$ denotes the mapping of $u \in \mathbb{R}^3$ to the space of skew-symmetric matrices so that $\widehat{u}\, u = 0$.

Note that equation (7) is a bilinear equation in $\tilde{x}_1^j$ and $\tilde{x}_2^j$. As a result, having enough number of point correspondences (at least 8) $\{\tilde{x}_1^j, \tilde{x}_2^j\}_{j=1}^N$ between the two cameras, one can reconstruct the fundamental matrix F up to a scale factor from the linear equation:

$$\begin{bmatrix} \tilde{x}_2^{1\top} \otimes \tilde{x}_1^{1\top} \\ \tilde{x}_2^{2\top} \otimes \tilde{x}_1^{2\top} \\ \vdots \\ \tilde{x}_2^{N\top} \otimes \tilde{x}_1^{N\top} \end{bmatrix} f = 0 \tag{8}$$

where $\otimes$ denotes the Kronecker product, and $f \in \mathbb{R}^9$ is obtained by stacking all rows of F into a vector.

When the intrinsic calibration parameters of the cameras, K_i, are known, one can use the metric coordinates $x_i^j = K_i^{-1} \tilde{x}_i^j$ in (6) to get the so called *epipolar constraint* as:

$$x_2^{j\top} E x_1^j = 0, \tag{9}$$

where $E = \widehat{T} R$ is called the *essential matrix*. It is known that, under generic conditions, one can recover the essential matrix using the 7-point algorithm ([22], [5]) and get the unique rotation R and translation T (up to a scale factor).

When the (intrinsic) calibration matrices are unknown, it is not as simple to recover intrinsic and extrinsic calibration parameters. In general, there are three main approaches to tackle the calibration problem for uncalibrated cameras:

1) *Calibration with a rig.* In this case one has access to the camera and an object with distinct points on it, and the coordinates of these points are known with high accuracy with respect to some reference frame.
2) *Calibration using a stratified approach.* In this case one assumes that the cameras have the same calibration matrix, i.e. $K_1 = K_2 = K$. First a projective reconstruction of the camera pose is achieved as $(KRK^{-1} + KTv^\top, KT)$ for some $v \in \mathbb{R}^3$. Then by finding a "plane at infinity" one can get v and upgrade the camera pose to an affine reconstruction (KRK^{-1}, KT). Finally, the calibration matrix K is obtained by solving a linear (Lyapunov) equation [22].
3) *Auto-calibration using Kruppa's equations.* This method addresses the calibration problem by solving Kruppa's equation $FYF^\top = \beta^2 \widehat{KT} Y \widehat{KT}^\top$, where $Y = KK^\top \in \mathbb{R}^{3\times 3}$ and $\beta \in \mathbb{R}$ are unknowns. These nonlinear equations are solved from three views (i.e. two fundamental matrices) to find the unknown matrix Y. Then the matrix K is obtained from the Cholesky factorization of Y. Although Kruppa's equation provides a nice algebraic relationship between the calibration matrix K and the fundamental matrix F, the unknown scale β requires to solve a set of nonlinear equations to get Y. Such equations are numerically unstable when the fundamental matrices are computed from noisy point correspondences.

IV. Distributed Calibration

As mentioned in the previous section, calibration of a camera is not an easy and straightforward task to do. The situation becomes worse in a camera sensor network where each camera possibly has a different calibration matrix. Looking back at the three methods of calibration, mentioned in the previous section, calibration with a rig does not seem to be a feasible approach, since it requires taking each one of the cameras separately and calibrating it with an object with known points coordinates. On the other hand, the stratified approach is a long process that requires finding a "plane at infinity" which can not be done automatically. Thus, the only interesting approach for calibrating a network of cameras is the auto-calibration approach which does not require any user interaction. However, as mentioned in the previous section, solving Kruppa's equation is not an straightforward task and requires solving a set of nonlinear equations.

To address these issues, we propose a method to automatically calibrate a network of cameras. We assume that (at least) one camera in the network is calibrated, the network graph is connected, and all cameras observe the same scene. This means we only need to calibrate one camera using a method such as calibration with a rig, and then the rest of the nodes in the network can automatically calibrate themselves by interacting only with their neighbors.

A. Calibration using one neighbor information

Without loss of generality, assume that node 1 represents the calibrated camera in the network. Each camera is associated with a flag that indicates whether it has been calibrated or not. At the beginning, only the flag of camera 1 is set to one and the value of the flag for the rest of the network is zero. As a result all neighbors of camera 1 in the network graph, would find a calibrated node in their adjacency. Consider an uncalibrated camera i adjacent to camera 1. We assume that the world reference frame is at the center of camera 1 and the relative pose of camera 1 with respect to camera i is denoted by (R_i, T_i). Therefore, the relation $X_1 = R_i X_i + T_i$ must hold for a point which has the coordinates X_1 and X_i with respect to the first and the i-th camera frames, respectively. Then, the relation between the projections of this point in the image planes of camera 1 and i can be written as:

$$\lambda_1 \tilde{x}_1 = \lambda_i K_1 R_i K_i^{-1} \tilde{x}_i + K_1 T_i. \tag{10}$$

Now, since the calibration matrix of camera 1 is known, we can replace $x_1 = K_1^{-1}\tilde{x}_1$ in the equation above and get

$$\lambda_1 x_1 = \lambda_i R_i K_i^{-1}\tilde{x}_i + T_i. \tag{11}$$

Then, by eliminating the unknown scales λ_1 and λ_i, we get the equation $x_1^\top F_i \tilde{x}_i = 0$ where

$$F_i = \widehat{T}_i R_i K_i^{-1} \tag{12}$$

is the fundamental matrix associated to camera 1 and i. One can immediately see that in this case, the two matrices are related by

$$E_i = F_i K_i. \tag{13}$$

One can recover the fundamental matrix F_i from a set of point correspondences $\{\tilde{x}_1^j, \tilde{x}_i^j\}_{j=1}^N$ using the 7-point algorithm [5] and then recover the translation $T_i^{'} = \gamma_i T_i$ (up to a scale factor γ_i), since by equation (12), T_i is in the left nullspace of F_i. Note that in order to compute F_i, node i needs to get the image points $\{\tilde{x}_1^j\}_{j=1}^N$ from node 1 (the calibrated node).

Having recovered $T_i^{'}$ and F_i, one particular canonical reconstruction of the projective camera matrices for camera i is given by $(\hat{T}_i^{'\top} F_i, T_i^{'})$, where $\hat{T}_i^{'\top} F_i = R_i K_i^{-1} + T_i^{'} v^\top$ for some $v \in \mathbb{R}^3$ [22]. If we let $Y_i = K_i K_i^\top$, it is easy to check that the following equation holds:

$$(\hat{T}_i^{'\top} F_i - T_i^{'} v^\top)\, Y_i\, (\hat{T}_i^{'\top} F_i - T_i^{'} v^\top)^\top = I. \tag{14}$$

Multiplying both sides of the above equation on left by $\widehat{T}_i^{'}$ and on right by $\widehat{T}_i^{'\top}$, we get:

$$(\widehat{T}_i^{'}\widehat{T}_i^{'\top} F_i)\, Y_i\, (F_i^\top \widehat{T}_i^{'}\widehat{T}_i^{'\top}) = \widehat{T}_i^{'}\widehat{T}_i^{'\top}. \tag{15}$$

One can show that $\widehat{T}_i^{'}\widehat{T}_i^{'\top} F_i = 1/\beta_i F_i$ for some $\beta_i \in \mathbb{R}$ and as a result equation (15) reduces to

$$F_i\, Y_i\, F_i^\top = \beta_i^2\, \widehat{T}_i^{'}\widehat{T}_i^{'\top}. \tag{16}$$

Notice that this is a linear equation in Y_i in contrast to the original Kruppa's equations which are nonlinear. As a result, one can linearly solve for $Y_i^{'}$ from $F_i\, Y_i^{'}\, F_i^\top = \widehat{T}_i^{'}\widehat{T}_i^{'\top}$ where $Y_i^{'} = 1/\beta_i^2\, Y_i$. Since $Y_i^{'}$ is still a positive definite matrix, we can get the calibration matrix $K_i^{'} = 1/\beta_i K_i$ using the Cholesky factorization. Since from (5), the last entry of the calibration matrix must be 1, we can eliminate the unknown scale β_i and get K_i by enforcing the last entry of $K_i^{'}$ to be 1.

Remark 1: According to Proposition 6.11 in [22], after eliminating the unknown scale in the general Kruppa's equation, one obtains at most two algebraically independent equations in the unknown parameters. With a similar analysis, we can show that equation (16) consists of at most two algebraically independent equations, allowing only two unknown variables in Y_i. As a result, one can recover the calibration matrix of each neighbor of node 1, when it has at most two unknown parameters. In the next subsection, we discuss a more general case where multiple calibrated nodes are available to begin with and discuss that under certain conditions on the network topology, we can recover the calibration matrices having more than two unknown parameters.

Having recovered the intrinsic calibration parameters K_i, for camera i, one can get the related essential matrix E_i from (13) as $E_i = F_i K_i$. Then, we can extract the pose information $(R_i, T_i^{'})$ from the essential matrix using the SVD of E_i. There would be generally 4 different solutions, but only one solution satisfies the positive depth constraint [22].

At this point, all the nodes adjacent to node 1 have calibrated themselves and obtained their relative pose with respect to node 1, so they set their calibration flags to 1. Then, neighbors of a newly calibrated camera i^* which detect a calibrated node in their adjacency, get the image points of node 1 from node i^* (this information is already available in node i^*). Then, they compute their fundamental matrices with respect to node 1, estimate their calibration matrices from (16), and finally extract their pose relative to the reference frame. By continuing this procedure, all cameras in the network calibrate themselves and obtain their relative poses with respect to camera 1, in at most $n-1$ steps.

B. Calibration using multiple neighbors information

While we showed how to calibrate a camera by having one calibrated neighbor, after the algorithm starts, it can happen that an uncalibrated camera has several calibrated neighbors. In this case, it would be preferable to use the information from all calibrated neighbors to obtain a more accurate estimate of the calibration parameters.

Looking back at the linear auto-calibration equation we derived in (16), for any two linearly independent vectors $\boldsymbol{w}_{i1}, \boldsymbol{w}_{i2} \perp T_i^{'}$, if we multiply both sides of the equation from left and right by $\boldsymbol{w}_{ij}^\top$ and $\boldsymbol{w}_{ij}$, respectively, we can get the unknown scale β_i^2 as:

$$\beta_i^2 = \frac{\boldsymbol{w}_{i1}^\top F_i Y_i F_i^\top \boldsymbol{w}_{i1}}{\boldsymbol{w}_{i1}^\top \widehat{T}_i^{'}\widehat{T}_i^{'\top}\boldsymbol{w}_{i1}} = \frac{\boldsymbol{w}_{i2}^\top F_i Y_i F_i^\top \boldsymbol{w}_{i2}}{\boldsymbol{w}_{i2}^\top \widehat{T}_i^{'}\widehat{T}_i^{'\top}\boldsymbol{w}_{i2}} = \frac{\boldsymbol{w}_{i1}^\top F_i Y_i F_i^\top \boldsymbol{w}_{i2}}{\boldsymbol{w}_{i1}^\top \widehat{T}_i^{'}\widehat{T}_i^{'\top}\boldsymbol{w}_{i2}}. \tag{17}$$

This gives two linear equations in the entries of the matrix Y_i. If we collect the 6 entries of Y_i (note that Y_i is symmetric) into a vector $\boldsymbol{y}_i \in \mathbb{R}^6$, we get a linear equation of the form

$$A_i \boldsymbol{y}_i = 0, \tag{18}$$

where $A_i \in \mathbb{R}^{2\times 6}$ is a matrix whose entries are a function of the elements of F_i, $T_i^{'}$. Notice that $\boldsymbol{w}_{i1}$ and $\boldsymbol{w}_{i2}$ can be chosen as two rows from $\widehat{T}_i$, thus the entries of A_i need not depend on the choice of $\boldsymbol{w}_{ij}$.

Now, assume that the uncalibrated node i detects $m \leq |N_i|$ calibrated nodes in its neighborhood. Then, it gets the image points $\{x_1^j\}_{j=1}^N$ of node 1, which are available at all the calibrated neighbors, to form the fundamental matrix F_i between itself and the reference frame. In addition, it gets the metric image coordinates $\{x_{i_1}^j, \ldots, x_{i_m}^j\}_{j=1}^N$ from the m calibrated nodes to form the fundamental matrices F_{i_k} between node i and i_k for $k \in \{1, \ldots, m\}$. Since the calibration matrix K_i for all these fundamental matrices is fixed, we would get $2(m+1)$

linear equations in $\boldsymbol{y}_i$ of the form (18), which together give

$$\begin{bmatrix} A_i \\ A_{i1} \\ \vdots \\ A_{im} \end{bmatrix} \boldsymbol{y}_i = 0. \tag{19}$$

As a result, we can get a better estimate of the calibration matrix of node i by using the additional information from all the calibrated neighbors. Similarly, node i receives the rotation matrices of the m calibrated neighbors with respect to the reference frame and estimates its rotation with respect to the reference frame and from the $m+1$ essential matrices $E_i, E_{i_1}, \ldots, E_{i_m}$ get $m+1$. Then a more accurate estimate of the rotation can be obtained by averaging these $m+1$ rotations in the manifold $SO(3)$ using the Karcher mean [23], [24].

Remark 2: The analysis in this part suggests to begin with more than one calibrated camera in the network and impose a certain network topology so that at the first step, the immediate (uncalibrated) neighbors of the calibrated cameras, have more than one calibrated node in their adjacency. This allows more freedom in the number of unknown parameters of the calibration matrix for each camera. However, as stated, it restricts the topology of the network to certain configurations.

V. Distributed 3D Structure Recovery

In the previous section, we showed how to calibrate a network of cameras having only one calibrated camera. As a result, every node i recovers its calibration matrix K_i and also its relative motion $(R_i, T_i^{'})$ with respect to the reference frame at node 1. In this section, we propose a method based on the consensus algorithm, such that all nodes in the network recover a common 3D structure by having each node exchange information only with its neighbors.

First, remember that the relation between the coordinates of 3D points in camera 1 and i is given by $X_i^j = R_i X_1^j + \gamma_i T_i^{'}$ for $j \in \{1, 2, \ldots, N\}$, where $(R_i, T_i^{'})$ are given from the camera calibration step. After projecting the points, $X_i^j = \lambda_i^j x_i^j$, and eliminating the unknown depth scale λ_i^j, one gets the following equation:

$$\hat{x}_i^j R_i X_1^j + \gamma_i \hat{x}_i^j T_i = 0, \tag{20}$$

where $x_i^j = K_i^{-1} \tilde{x}_i^j$ represents the metric coordinates of point j in the image plane of camera i. Since the scale of motion γ_i is fixed for all points in camera i, one can recover the 3D structure of the scene up to a scale factor from the following equation:

$$\begin{bmatrix} \hat{x}_i^1 R_i & 0 & \cdots & 0 & \hat{x}_i^1 T_i \\ 0 & \hat{x}_i^2 R_i & \cdots & 0 & \hat{x}_i^2 T_i \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \hat{x}_i^N R_i & \hat{x}_i^N T_i \end{bmatrix} \begin{bmatrix} X_1^1 \\ \vdots \\ X_1^N \\ \gamma_i \end{bmatrix} = 0. \tag{21}$$

A naive approach would be that every node solves (21) to find the 3D structure and the scale factor. However, there are two main problems with this approach. First, (21) is equivalent to finding a vector in the nullspace of a matrix, so a scaled version of a solution would also be a valid solution. As a result, each node recovers the 3D structure with a different scale. More importantly, when the data $\{\tilde{x}_i^j\}_{j=1}^N$ are noisy, the solution of (21) for each node would be different.

We now propose a distributed algorithm to aggregate information from all cameras in order to get a common solution for the whole network. Let for each node $i \in \{1, 2, \ldots, n\}$,

$$M_i \triangleq \begin{bmatrix} \hat{x}_i^1 R_i & 0 & \cdots & 0 \\ 0 & \hat{x}_i^2 R_i & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \hat{x}_i^N R_i \end{bmatrix} \quad \text{and} \quad P_i \triangleq \begin{bmatrix} \hat{x}_i^1 T_i \\ \vdots \\ \hat{x}_i^N T_i \end{bmatrix} \tag{22}$$

where $M_i \in \mathbb{R}^{3N \times 3N}$ and $P_i \in \mathbb{R}^{3N}$. Then using (21), the following equation must hold:

$$\begin{bmatrix} M_1 & 0 & 0 & \ldots & 0 \\ M_2 & P_2 & 0 & \ldots & 0 \\ M_3 & 0 & P_3 & \ldots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ M_n & 0 & 0 & \ldots & P_n \end{bmatrix} \begin{bmatrix} X_1^1 \\ \vdots \\ X_1^N \\ \gamma_2 \\ \vdots \\ \gamma_N \end{bmatrix} = 0. \tag{23}$$

If each node solves the above equation and enforces the norm of the solution to be 1, then all nodes get the same 3D structure. However, note that solving (23) at each node requires having the information from all nodes in the network which of course is not possible.

Now, we show how to solve (23) in a distributed way using the consensus algorithm described in Section II. Let $W \in \mathbb{R}^{3nN \times (3N+n-1)}$ denote the matrix on the left side of (23) with the SVD of $W = U \Sigma V^\top$. Then what we are interested in is to find the nullspace of W, i.e. the last column of V, in a distributed way. Let W_i denote the i-th block of $3\,N$ rows of W containing the information from node i, i.e. $W = \begin{bmatrix} W_1^\top, & \cdots, & W_n^\top \end{bmatrix}^\top$, then we have the following:

$$W^\top W = \sum_{i=1}^{n} W_i^\top W_i = V \Sigma^2 V^\top. \tag{24}$$

One can see from the above equation that in order to compute V at each node, we need to have $\sum_{i=1}^n W_i^\top W_i$ at every node.

Since $W_i^\top W_i$ is the information available at node i, nodes in the network can use the iterative consensus algorithm in (2) to compute $\frac{1}{n} \sum_{i=1}^n W_i^\top W_i = \frac{1}{n} V \Sigma^2 V^\top$. When the nodes reach consensus, each node can find the norm one vector in the nullspace of equation (23) by taking the last column of V. As a result, all nodes recover the same 3D structure in addition to all the scales γ_i.

VI. Experiments

In this section, we present synthetic experiments to illustrate the behavior of our method. We consider a nonplanar network of $n = 14$ cameras, where only the first camera is calibrated ($K_1 = I$). We assume that for each uncalibrated camera, the position of the optical center (o_x, o_y) is known and the skew

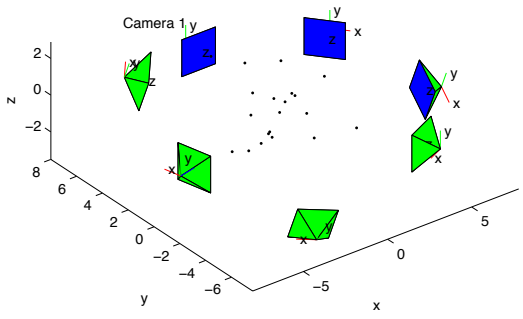

Fig. 1. Synthetic Camera Network Setup

factor s_θ is equal to zero. The unknown parameters fs_x and fs_y in (5) are chosen from Normal distributions with means of 400 and 500, and variances of 10 and 20, respectively. The cameras are roughly distributed in a circle of radius 8 times the average focal length of cameras and they are all facing toward the center of the scene. The structure of the scene is represented by $N = 20$ points randomly distributed in a cube with the length of each side to be 5.6 times the average focal length of cameras. We assume the network graph is 4-regular meaning that each node has 4 neighbors in the network. Figure 1 illustrates the camera network setup where only 7 cameras are shown.

We test our method under 4 different noise levels: when there is no noise in the image points, and when we add Gaussian noise with variance of 1, 2, and 3 pixels to the image points for an image of size 1000×1000 pixels.

We find the fundamental matrices by using the 7-point algorithm [5] and then get the calibration matrices from equation (16). Table I shows the average error of the estimated calibration matrices over 100 trials. The error is defined as

$$Err = \frac{||K_i - \tilde{K}_i||_F}{||K_i||_F} \times 100\%, \tag{25}$$

where K_i and $\tilde{K}_i$ are the original and the estimated calibration matrices, respectively. The results show that when there is no noise in the images, calibration matrices are estimated exactly with zero errors. However, the solution of Kruppas equation is sensitive to the error in the recovered scale. As a result, estimation accuracy decreases drastically by increasing the noise level. It would remain an open problem how to solve Kruppas equation in a robust way for the case of noisy image coordinates.

After finding the calibration matrix of each camera, we use (13) to get the essential matrices and extract the pose of the cameras. Table II shows the average and the variance of angles between the estimated and the ground-truth rotations over 100 trials in addition to the angle between the estimated and the ground-truth translations. Note that since translations are always recovered up to a scale factor, we use the angle between the normalized estimate and the ground-truth translations as the measure of error. The results indicate that the method performs well when the noise is small. However, the performance of estimation decreases for larger values of noise partly because of the error in the estimation of the true calibration matrices and partly as a direct result of having noise in the image points.

Finally, we use the iterative consensus algorithm in (3) to find the common 3D structure for $\epsilon = 0.02\Delta_G^{-1}$. Table III shows the average and the variance of angles between the estimated and the ground-truth 3D points. Again, since the structure is always recovered up to a scale factor, we used the angle between the estimated 3D structure (as a $3N$-dimensional vector) and the true one. Figure 2 shows how after only a few iterations all the nodes estimates of the 3D structure converge to the centralized solution of equation (23).

TABLE I
AVERAGE AND VARIANCE OF CALIBRATION ERRORS (%)

Noise	0 px	1 px	2 px	3 px
Average of Err	0.000	0.810	2.130	4.380
Variance of Err	0.000	0.030	0.080	0.210

TABLE II
AVERAGE AND VARIANCE OF ROTATION AND TRANSLATION ANGLE ERRORS (DEG.)

	Noise	0 px	1 px	2 px	3 px
Rotation Error	Average	0.000	0.947	1.978	4.364
	Variance	0.000	0.142	0.321	1.055
Translation Error	Average	0.000	0.998	1.830	4.498
	Variance	0.000	0.125	0.464	1.073

TABLE III
AVERAGE AND VARIANCE OF 3D STRUCTURE ANGLE ERRORS (DEG.)

Noise	0 px	1 px	2 px	3 px
Average	0.000	0.989	2.097	4.835
Variance	0.000	0.109	0.570	1.196

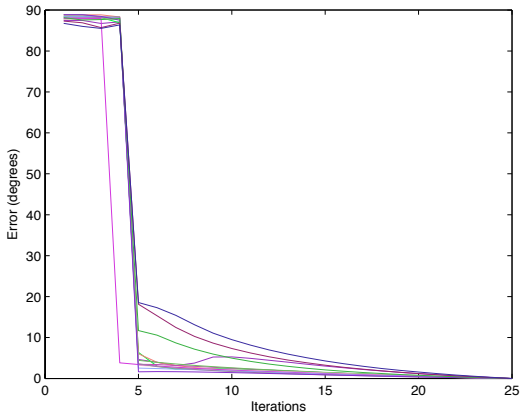

Fig. 2. Structure estimation error between centralized and distributed solution at each node in the network. Each curve indicates convergence at each node.

VII. Conclusion

In this paper, we showed how to calibrate a network of cameras by having only one calibrated camera. We proposed a method based on auto-calibration and epipolar geometry to calibrate both the intrinsic and the extrinsic parameters of all the cameras in a distributed way. Finally, we proposed a distributed consensus algorithm to find a common 3D structure of the scene in all the cameras. Future work includes finding the pose parameters from a global cost function in a distributed way as well as exploring robust methods for estimating the parameters in a distributed fashion.

Acknowledgment

This work was partially supported by startup funds from JHU and by grants NSF CAREER ISS-0447739, NSF CNS-0834470, and ONR N00014-05-10836.

References

[1] R. Y. Tsai, "An efficient and accurate camera calibration technique for 3D machine vision," in *IEEE Conf. on Computer Vision and Pattern Recognition*, ser. IEEE Publ.86CH2290-5. IEEE, 1986, pp. 364–374.

[2] Z. Zhang, "A flexible new technique for camera calibration," *Microsoft Technical Report MSR-TR-98-71*, 1998.

[3] Z. Zhang, Q.-T. Luong, and O. Faugeras, "Motion of an uncalibrated stereo rig: self-calibration and metric reconstruction." *IEEE Transactions on Robotics and Automation*, vol. 12, no. 1, pp. 103–13, 1996.

[4] P. Sturm, "Critical motion sequences for monocular self-calibration and uncalibrated Euclidean reconstruction," in *IEEE Conf. on Computer Vision and Pattern Recognition*. IEEE Comput. Soc. Press, 1997, pp. 1100–1105.

[5] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge, 2004.

[6] B. Triggs, "Autocalibration and the absolute quadric," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 1997.

[7] ——, "Autocalibration from planar scenes," in *IEEE Conf. on Computer Vision and Pattern Recognition*, 1998.

[8] M. Pollefeys, R. Koch, and L. V. Gool, "Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters," in *IEEE Int. Conf. on Computer Vision*, 1998.

[9] R. I. Hartley, "Self-calibration from multiple views with a rotating camera," in *European Conf. on Computer Vision*, 1994, pp. 471–8.

[10] Q.-T. Luong and O. Faugeras, "Self-calibration of a moving camera from point correspondences and fundamental matrices." *Int. Journal of Computer Vision*, vol. 22, no. 3, pp. 261–89, 1997.

[11] E. Kruppa, "Zur ermittlung eines objecktes aus zwei perspektiven mit innerer orientierung." *Sitz.-Ber.Akad.Wiss., Math.Naturw., Kl.Abt.IIa, 122:1939-1948*, 1913.

[12] P. Baker and Y. Aloimonos, "Calibration of a multicamera network," in *CVPR Workshop on Omnidirectional Vision and Camera Networks*, 2000, pp. 134–141.

[13] A. Barton-Sweeney, D. Lymberopoulos, and A. Savvides, "Sensor localization and camera calibration in distributed camera sensor networks," in *International Conference on Broadband Communications, Networks and Systems*, 2006, pp. 1–10.

[14] R. Farrell, R. Garcia, D. Lucarelli, A. Terzis, and I.-J. Wang, "Localization in multi-modal sensor networks," in *Third International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, 2007.

[15] S. Sinha, M. Pollefeys, and L. McMillan, "Camera network calibration from dynamic silhouettes," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, 2004, pp. 195–202.

[16] S. Sinha and M. Pollefeys, "Synchronization and calibration of a camera network for 3D event reconstruction from live video," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2005, p. 1196.

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

[18] D. Devarajan and R. Radke, "Calibrating distributed camera networks using belief propagation," *EURASIP Journal of Applied Signal Processing*, pp. 221–221, 2007.

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

[20] R. Olfati-Saber and R. Murray, "Consensus problems in networks of agents with switching topology and time-delays," *IEEE Transactions on Automatic Control*, vol. 49, no. 3, pp. 1520–1533, 2004.

[21] R. Olfati-Saber, J. Fax, and R. Murray, "Consensus and cooperation in networked multi-agent systems," *Proceedings of the IEEE*, vol. 95, no. 1, pp. 215–233, 2007.

[22] Y. Ma, S. Soatto, J. Košecká, and S. Sastry, *An Invitation to 3D Vision: From Images to Geometric Models*. Springer Verlag, 2003.

[23] H. Karcher, "Riemannian center of mass and mollifier smoothing," *Communications on Pure and Applied Mathematics*, vol. 30, no. 5, pp. 509–541, 1977.

[24] J. Manton, "A globally convergent numerical algorithm for computing the centre of mass on compact Lie groups," in *International Conference on Automation, Robotics, Control and Vision*, vol. 3, 2004, pp. 2211–2216.