Distributed Localization of Networked Cameras

Stanislav Funiak
Carnegie Mellon

Carlos Guestrin
Carnegie Mellon

Mark Paskin
Stanford University

Rahul Sukthankar
Intel Research

ABSTRACT

Camera networks are perhaps the most common type of sensor network and are deployed in a variety of real-world applications including surveillance, intelligent environments and scientific remote monitoring. A key problem in deploying a network of cameras is calibration, i.e., determining the location and orientation of each sensor so that observations in an image can be mapped to locations in the real world. This paper proposes a fully distributed approach for camera network calibration. The cameras collaborate to track an object that moves through the environment and reason probabilistically about which camera poses are consistent with the observed images. This reasoning employs sophisticated techniques for handling the difficult nonlinearities imposed by projective transformations, as well as the dense correlations that arise between distant cameras. Our method requires minimal overlap of the cameras' fields of view and makes very few assumptions about the motion of the object. In contrast to existing approaches, which are centralized, our distributed algorithm scales easily to very large camera networks. We evaluate the system on a real camera network with 25 nodes as well as simulated camera networks of up to 50 cameras and demonstrate that our approach performs well even when communication is lossy.

Categories and Subject Descriptors: G.3 Probability and Statistics: Miscellaneous; I.4.1 Digitization and Image Capture, Camera calibration.

General Terms: algorithms, experimentation.

Keywords: sensor networks, graphical models.

1. INTRODUCTION

Camera networks are perhaps the most common type of sensor network. These networks are ubiquitous in a variety of real-world applications including surveillance, intelligent environments and scientific remote monitoring. In most applications, camera network data is only useful if we know from where the images were captured, i.e., the real world location of the cameras. Manually measuring the **pose** (location and orientation) of all cameras in the network is a very tedious and time consuming task. In this paper, we present a distributed method for solving this **calibration** task automatically by using information provided by the actual cameras in the network.

IPSN'06, April 19–21, 2006, Nashville, Tennessee, USA.

Suppose that a moving object is seen in the field of view of a camera and, a few moments later, the same object is observed by another camera; if we knew the trajectory of this object, we could infer information about the relative position of the two cameras. Unfortunately, without an independent localization system like GPS, the trajectory is unknown. However, if we knew the poses of the cameras, we could infer the trajectory of the object. Therefore, we can address the camera network calibration task by solving a **simultaneous localization and tracking** (SLAT) problem, where we estimate both the trajectory of the object and the poses of the cameras. An effective solution of the SLAT problem leads to a very simple camera network deployment procedure: cameras are placed throughout the environment at unknown locations, then, as an object (e.g., a person) moves throughout the environment following an unknown trajectory, the network automatically calibrates itself.

Cameras provide noisy observations about possible locations of the moving object, and there may be times when the object is not visible by any camera. Thus, we formulate SLAT as a *probabilistic inference* task, where we maintain a joint distribution over possible object locations and poses of all cameras, given the images collected by the network. The object location and the poses of the cameras are continuous variables in this model. Unfortunately, representing general distributions over continuous variables is a very challenging task, and most representations lead to intractable inference. If the distribution can be represented as a Gaussian, however, the inference task can be solved by simple matrix operations. Unfortunately, the camera calibration problem has nonlinearities (e.g., due to periodicity in the angles) that cannot directly be represented by Gaussian distributions. This paper presents a novel approach, **relative over-parameterization** (ROP) of the camera pose, that enables us to represent the complex distributions in the SLAT problem effectively using a single Gaussian. Even with our ROP representation, however, we still need to incorporate nonlinear information obtained from the camera network. We show that the standard procedure for integrating this nonlinear information leads to inaccurate SLAT solutions. We address this problem by proposing a novel **conditional hybrid linearization** procedure that isolates and addresses the main source of nonlinearity, which is uncertainty in the camera angle. The combination of our ROP representation with this linearization procedure leads to very precise SLAT solutions using a simple Kalman filter, even with minimal or no overlap of the cameras' fields of view. A strong advantage of this approach is that it provides an explicit representation of the **uncertainty** in the estimate of camera poses. By representing uncertainty, we have a direct measure of the quality of the solution, indicating when the calibration procedure can be stopped, and what parts of the network need more information to improve their calibration, poten-

tially enabling an **active** control of the path of the object that optimizes the quality of the solution.

Often, we cannot expect a camera network to be able to upload all images to a central location, e.g., in ad-hoc deployments, when the network is large, when the nodes are resource constrained, or in situations where a single point of failure could jeopardize the entire system, such as in an emergency response system. To address these situations, we present a distributed solution to the SLAT problem that builds on our recent work on **distributed probabilistic inference** [3, 9, 11]. Our approach only uses local communication between nearby camera nodes, and guarantees that the solution obtained will be exactly the same as if we were to download all of its images to a central location. Our approach is **online**, so that at any time, each camera can locally obtain an estimate of its pose given the images observed thus far by all of the cameras. The messages communicated by our algorithm are compact summaries of observations made by large sets of cameras, so nodes never need to transmit images, significantly reducing the communication cost. Furthermore, our approach provides strong guarantees with respect to node failures and lossy communication: even when a subset of the camera nodes fail, our algorithm is guaranteed to provide a principled approximation of the solution obtained by the remaining cameras.

The main contributions of this paper are:

1. A demonstration of the viability of distributed, precise and efficient calibration of a large network of cameras by simply tracking a moving object.
2. Novel representation and linearization procedures yielding effective solutions to SLAT using simple Gaussians, despite the problem's complexity and nonlinearity.
3. A scalable and distributed algorithm for the SLAT problem that is guaranteed to converge to the same solution as the centralized approach, providing robustness guarantees with respect to lossy communication and node failures.
4. An experimental validation of the approach on several large simulated scenarios and on real data from a network of 25 cameras, demonstrating minimal estimation error.

2. SIMULTANEOUS LOCALIZATION AND TRACKING IN CAMERA NETWORKS

2.1 Problem formulation

Our goal is to recover the poses of the cameras in a camera network. In general, the pose of a camera can be represented by six parameters: three position parameters x, y, z, and three angles (e.g., roll, pitch, yaw). This paper focuses on recovering three of these parameters: the (x, y) location of the camera and an angle θ, i.e., the rotation around the z-axis; the remaining parameters (such as tilt and height) are assumed to be known. We call (x, y, θ) the **absolute parameterization** of a camera's pose. This parameterization can represent a wide range of camera poses, including downward-facing cameras attached to a ceiling and wall-mounted cameras at known heights.

The cameras estimate their pose parameter by tracking a moving object. We assume that this object maintains an (approximately) known height throughout its motion, so that its location can be characterized by (x, y) coordinates. We assume very little about the motion of this object except smoothness—the object can stop, change direction, or change speed. For example, the moving object could be a visually distinct marker carried by a person. The images observed by the cameras are governed by perspective projection, and are therefore highly nonlinear both in the camera's pose parameters and the object's location.

2.2 Assumptions on the camera network

We assume a general camera network model where each camera node has (limited) resources for computation and communication, and synchronized internal clocks that allow the nodes to share a common notion of time. We assume that the camera nodes communicate using a multiple access channel that does not guarantee perfect communication; messages can be lost, and interference can partition the network. This communication model is general enough to accommodate a wide range of camera nodes from Crossbow motes with on-board cameras to wireless webcams.

2.3 Related work

SLAT is related to a problem in mobile robotics called **simultaneous localization and mapping** (SLAM) [10]. In SLAM, a mobile robot observes landmarks and from these observations, its odometry, and its control signals, the robot must jointly estimate its location and the positions of the landmarks. We can view SLAT as a SLAM problem where the cameras play the role of landmarks, and the moving object plays the role of the robot; the key difference is that in SLAT, the cameras observe the object; in SLAM, the robot observes the landmarks. In some ways, SLAT is easier than SLAM: in SLAT there is no data association problem, since there is only a single object; in SLAM, there are many landmarks and the robot must reason about which is associated with each observation. In other ways, SLAT is more difficult than SLAM; in SLAM there is significant information about the motion of the robot (from its odometry and controls), whereas in SLAT we know little about the object's dynamics. Another feature which makes SLAT more challenging is that there are many variables that represent angles—and thus interact nonlinearly—whereas in SLAM there is typically only one angular variable, the orientation of the robot.

Aspects of SLAT are also related to work in computer vision in **multiple camera tracking and calibration** [5,15], which as largely focused on overlapping camera configurations, and **structure from motion** (SFM) [8,12,14]. Given a sequence of images of a static scene captured by a moving camera, the goal of SFM is to recover the 3D geometry of the scene and the trajectory of the camera motion, typically by using correspondences between feature points. There are two key differences between SLAT and SFM. In SLAT, the positions from which images are obtained are not related by smooth motion of the camera, but by the layout of the camera network; this means that the overlap between different images is typically much smaller. Additionally, in SFM geometric information is extracted from large sets of feature point correspondences; in SLAT, only a single point is tracked—the moving object.

In the sensor networks community, there has been a large body of work on localizing nodes from pairwise distance estimates (c.f. [4] for one such approach, and [18] for an interesting analysis of this problem). The assumptions of SLAT are weaker, since we localize nodes by simply tracking an external, uncontrolled object. A related approach that solves the camera localization problem distributedly but relies on feature point correspondences is presented in [7]. Perhaps the closest work in the sensor network community is that on *passive localization*, where sensors attempt to localize themselves using sound events of unknown origin [16].

A recent proposal by Rahimi *et al.* to address the SLAT problem uses an offline optimization algorithm [13]. This approach is based upon a probabilistic model that is similar to ours, but rather than computing a complete posterior distribution over camera poses, they compute the most likely trajectory and the most likely pose for each camera with the Newton-Rhapson method. In Sec. 6, we show that our approach and that of Rahimi *et al.* provide solutions of comparable quality. However, the approach of Rahimi *et al.* is offline, centralized and does not provide an explicit representation of the uncertainty in the solution. On the other hand, our algorithm is online, distributed and provides uncertainty estimates that can be used for active control.

3. DYNAMIC PROBABILISTIC SLAT

We model the SLAT problem using a linear dynamical system [13]. The variables of this system are the location and velocity of the object at each time step, M_t, and for each camera i, the pose of the camera C_i. The motion of the object is modeled using a Brownian motion:

$$M_t = \begin{bmatrix} M_t^x \\ M_t^y \\ M_t^{\dot{x}} \\ M_t^{\dot{y}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} M_{t-1}^x \\ M_{t-1}^y \\ M_{t-1}^{\dot{x}} \\ M_{t-1}^{\dot{y}} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ \epsilon \\ \epsilon \end{bmatrix}, \tag{1}$$

where (M_t^x, M_t^y) is the object's position, $(M_t^{\dot{x}}, M_t^{\dot{y}})$ is its velocity, and ϵ is a white noise variable giving additive noise in velocities. This motion model assumes little about the motion of the object except smoothness. It is a linear-Gaussian model, because M_t is a linear function of M_t and some additive Gaussian noise. This means that we can represent the **motion model** $p(M_t \mid M_{t-1})$ using a compact parametric form [2].

When the object appears in the image of camera i, an observation is generated which is represented by a point, $O = (O^x, O^y)$, in the image coordinates of that camera. This observation depends upon the object's state M_t and the camera's pose C_i via

$$\begin{bmatrix} O^x \\ O^y \end{bmatrix} = g(M_t, C_i) + \begin{bmatrix} \delta_{t,i}^x \\ \delta_{t,i}^y \end{bmatrix}, \tag{2}$$

where g is the (non-linear) projective transformation for camera i and δ are white noise variables with a small standard deviation (e.g., 3 pixels). For instance, when an overhead (downward-facing) camera with known focal length, f, located at (C^x, C^y) rotated at an angle of θ observes an object located at (M^x, M^y) and a known height offset h, the observation is given by

$$\begin{bmatrix} O^x \\ O^y \end{bmatrix} = \frac{f}{h} R_\theta \begin{bmatrix} M^x - C^x \\ M^y - C^y \end{bmatrix} + \begin{bmatrix} \delta^x \\ \delta^y \end{bmatrix},$$

where R_θ represents a clockwise rotation by θ. This measurement equation is not linear-Gaussian; therefore, we cannot represent the **observation model** $p(O \mid C_i, M_t)$ exactly using linear-Gaussian parameters. Our approach, described below, is to use linearization to find a good linear-Gaussian approximation to the observation model.

To complete our definition of the probability model, we must specify the prior distribution over the object location at the first time step, $p(M_1)$, and the poses of the cameras $p(C_i)$. Our observations give us only relative information, so any translation or rotation of the coordinate frame is equally reasonable. To resolve the coordinate system, we initialize the prior of the first camera that observes the object to a point mass at the origin and set its orientation to zero. The remaining priors (over the object location and the other cameras' parameters), are "uniform", represented by a Gaussian with a large variance [2].

At each time step, we get some set of object observations $\mathbf{o}_t$. The **belief state** at time t is defined to be

$$p(M_t, \mathbf{C} \mid \mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_{t-1}). \tag{3}$$

This is the posterior distribution over the current state of the system (i.e., the object state and all cameras' poses), given all observations made so far. A **filtering** algorithm iteratively computes the belief state at time t using the previous belief state, the probability model, and the observations at time $t-1$. For example, given our linear Gaussian assumptions, the belief state can be represented by a mean vector μ_t and a covariance matrix Σ_t, and it can be computed using the Kalman filter (c.f. [6]). The mean vector represents the most likely values for the state variables, and in particular, it represents the most likely poses for the cameras.

The filtering update can be viewed in terms of a three step process—a view that will be useful to us later:

Estimation. In this step, we condition on the observations of the current time step by computing

$$p(M_t, \mathbf{C} \mid \mathbf{o}_{1:t}) \propto p(M_t, \mathbf{C} \mid \mathbf{o}_{1:t-1}) \cdot p(\mathbf{o}_t \mid M_t, \mathbf{C}), \tag{4}$$

$$= p(M_t, \mathbf{C} \mid \mathbf{o}_{1:t-1}) \prod_i p(o_t^i \mid M_t, C_{\omega(i)}). \tag{5}$$

The first term on the r.h.s of Eq. (4) is the previous belief state, and the second term is the likelihood of the current observations. In Eq. (5), we have used the assumption that observations are independent given the location of the object and the pose of the camera that made the observation; this allows us to decompose the likelihood into a product of likelihoods, one per object observation: o_t^i is the i^{th} observation, and $C_{\omega(i)}$ is the pose of the camera that received the observation. Note that each observation depends upon the location of the object and the pose of the observing camera—not upon the joint state vector. To summarize: estimation is accomplished by multiplying into the belief state a likelihood for each observation (and then renormalizing).

Prediction. In this step we augment the belief state with the new object state variable by computing

$$p(M_{t+1}, M_t, \mathbf{C} \mid \mathbf{o}_{1:t}) = p(M_t, \mathbf{C} \mid \mathbf{o}_{1:t}) \cdot p(M_{t+1} \mid M_t) \tag{6}$$

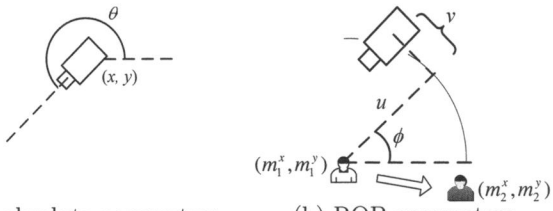
The first term on the r.h.s is the result of estimation, and the second term is the object's motion model from Eq. (1).

Roll-up. In this step we marginalize out the object's state variable from the current time step by computing

$$p(M_{t+1}, \mathbf{C} \mid \mathbf{o}_{1:t}) = \int p(M_{t+1}, M_t = m_t, \mathbf{C} \mid \mathbf{o}_{1:t}) \mathrm{d}m_t. \tag{7}$$

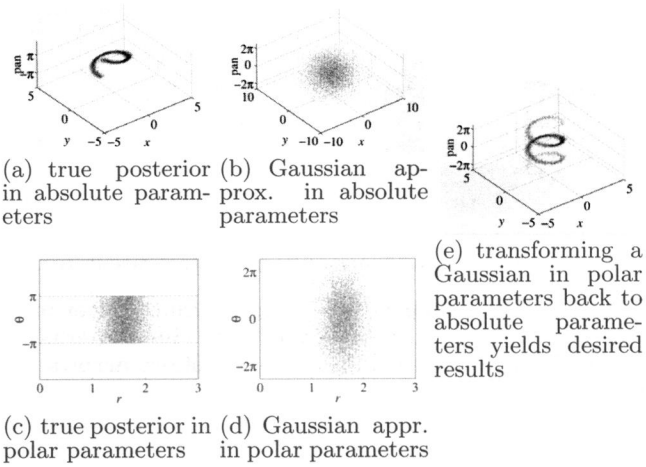
The first term on the r.h.s is the result of prediction, and the l.h.s. is the belief state at the next time step.

In our setting, where the joint distribution is (approximated by) a multivariate Gaussian distribution, the multiplications and marginalizations required by the filtering updates can be implemented algebraically using simple matrix operations, as described in [2].



(a) absolute parameters (b) ROP parameters

Figure 1: Two parameterizations that represent the pose of a camera. (a) Standard parameterization in terms of camera center (x, y) and orientation θ. (b) ROP parameterization that expressess the camera pose as a composition of a translation and a rotation about a hypothetical location (m^x, m^y) of the object when first observed. At $t = 1$, the camera makes its first observation of the tracked object (person), and represents its distribution in terms of m_1^x, m_1^y, u, v, and ϕ. (m_1^x, m_1^y) is the unknown location of the person at $t = 1$, ϕ is the camera orientation, u represents the distance of the camera's image plane from the person, and v the lateral offset (if $v = 0$, the camera would observe the person head on). If we vary ϕ from $-\pi$ to π, the camera traces a circle around (m_1^x, m_1^y). (m_1^x, m_1^y) remains a part of the camera's belief state even after the object has moved to a different location (m_2^x, m_2^y) at the next time step.



(a) true posterior in absolute parameters (b) Gaussian approx. in absolute parameters (e) transforming a Gaussian in polar parameters back to absolute parameters yields desired results

(c) true posterior in polar parameters (d) Gaussian appr. in polar parameters

Figure 2: The folly of Gaussian representations in absolute parameters, and the improvement obtained with relative parameters. (a) the posterior distribution of a camera's pose in absolute parameter space, given that it has observed the object at the origin. The distribution forms a spiral in the (x, y, θ) space and a ring in the (x, y) space. (b) the best Gaussian approximation to this posterior; note the bad approximation. In contrast, when expressed in polar coordinates (c), the posterior can be effectively approximated with a single Gaussian (d). By transforming this distribution back to the absolute parameters (e), we can verify that we obtained an accurate approximation of the true posterior in (a).

4. ADDRESSING NON-GAUSSIANITY

Thus far, we have referred to the pose of camera i abstractly as C_i. One possible representation for this pose uses the absolute parameters (x, y, θ). This parameterization is illustrated in Fig. 1(a).

Suppose that a camera with an unknown pose observes the object at a known position. Given the heights of the object and camera are known, we can estimate the distance from the camera to the object using a simple inverse projection. Unfortunately, we cannot recover the orientation of the camera θ, as the camera could be anywhere in a ring around the object's location. Fig. 2(a) illustrates this phenomenon by visualizing the true posterior distribution over possible camera poses in absolute coordinates given an observation of a object with known location. This ring-like distribution is highly non-Gaussian, and if we tried to approximate it with a Gaussian, the problem structure would be lost, as shown in Fig. 2(b).[1] Because of this, applying the Kalman filter to solving the SLAT problem fails when the camera poses are represented in absolute parameters, see Fig. 3(b).

4.1 Relative over-parameterization

One approach for representing such ring-like distributions is to use a mixture of Gaussians [4]. Unfortunately, computations with mixtures of Gaussians are significantly more costly, losing the simplicity of the Kalman filter approach; typically an exponential number of mixture components are required to represent the pose of multiple cameras simultaneously. We now present a novel, simple reparameterization of the problem that allows us to represent these complex distributions with a single Gaussian.

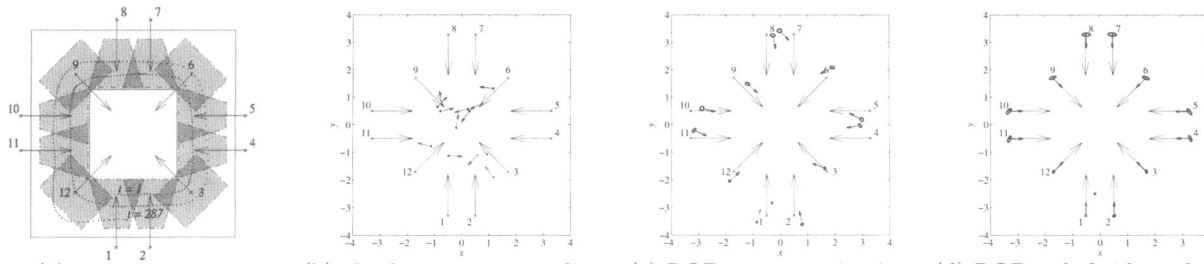
Our reparameterization is based on a simple intuition. Fig. 2(b) shows that a Gaussian in absolute parameters cannot represent the ring structure of the position variables in Fig. 2(a). Nevertheless, this structure can be represented well with a Gaussian in *polar coordinates* (r, ϕ), Fig. 2(c), where the origin corresponds to the observed object's true location, r is the distance to the camera's position, and the angle ϕ describes both the orientation of the camera and its orientation with respect to the object, since the camera must be looking inward toward the object. A Gaussian with a small variance for r would provide a good approximation, see Fig. 2(d). Comparing Fig. 2(e) to the exact posterior in Fig. 2(a), we see that we have obtained a sensible approximation to the true distribution of the camera pose.

This intuition has two problems that we must correct. First, the object location—the origin of our polar coordinate system—is not known with certainty when the object is observed; to remedy this, we can add its position (m^x, m^y) position to the pose variables, so that they too can be estimated from observations. Second, the camera does not necessarily observe the object head–on, but at an angle according to the orientation of the camera. To correct this, we can substitute for the radius r a pair of parameters, u and v, which describe the distance from the object to its projection on the camera's image plane (u), and the distance from this projection to the camera's center (v). Thus, our **relative over-parameterization** (ROP) of a camera's pose is given by (m^x, m^y, u, v, ϕ), as illustrated in Fig. 1(b).

Using the ROP representation in our Kalman filter requires two small changes. First, the observation model Eq. (2)

[1]Placing Gaussian distributions over angular variables requires some care because of periodicity. In our convention, a Gaussian-distributed angle Θ with mean μ and variance σ^2 is distributed so that for all $-\pi \leq \theta_0 \leq \theta_1 < \pi$,

$$\Pr\{\theta_0 \leq \Theta \leq \theta_1\} = \sum_{k=-\infty}^{\infty} \int_{\theta_0+2k\pi}^{\theta_1+2k\pi} \mathcal{N}(\theta; \mu, \sigma^2) \mathrm{d}\theta.$$



(a) tower scenario (b) absolute + accurate lin. (c) ROP parameterization (d) ROP + hybrid cond. lin.

Figure 3: The performance of the Kalman Filter on a simulated network of cameras. The long arrows indicate the true location and orientation of the cameras, the ellipses (often small) are 95% confidence intervals in the position estimates, and the two short arrows (often overlapping) are the 95% confidence intervals in the estimate of orientation. (a) Eight side-facing and four overhead cameras are placed around a hallway. The dark-shaded regions represent the overlapping fields of the view. The dotted line shows the location of the object at each time step. (b) Nonlinearities give poor results when camera poses are represented as (x, y, θ), even with accurate linearization of the observations. (c) The ROP representation of camera poses improves results. (d) Combining ROP with the hybrid conditional linearization techniques gives excellent results.

must be expressed in terms of (m^x, m^y, u, v, ϕ), which requires the projection operation $g(M, C)$ to first perform a transformation of the ROP camera pose C into absolute coordinates, (x, y, θ), and then apply the standard projection. Second, the prior over camera poses must also be converted to the ROP representation. This prior is similar to the one in absolute coordinates: If camera i first observes the object at time t, the prior over the coordinates (u, v, ϕ) is uniform, and the prior over (m^x, m^y) is defined such that these variables are exactly equal to the (unknown) object position (M_t^x, M_t^y). This conversion preserves the strong correlations among the cameras that arise when the person is seen by multiple cameras at once: the cameras will be correlated through their values of (m^x, m^y). Using our new ROP representation, we are able to obtain very precise pose estimates for a large camera network using only a Kalman filter with a single Gaussian, as shown in Fig. 3(d).

4.2 Hybrid conditional linearization

Recall that our observation model in Eq. (2) includes a projection operation that is highly nonlinear. This makes it impossible to directly apply the Kalman filter to SLAT, because its linear-Gaussian assumptions are violated. The standard approach to applying the Kalman filter to nonlinear systems is to adopt a strategy for **linearization**, which chooses a linear approximation to the observation model in the region that has highest probability according the prior distribution. The most sophisticated techniques are based on numerical integration, including Gaussian Quadrature and Exact Monomials [6, §6], which include as a special case the *Unscented Kalman filter* (UKF) [17].[2] In this section, we briefly describe this approach to linearization and two new enhancements we developed for SLAT, *hybrid linearization* and *hybrid conditional linearization*, which dramatically improve the linearization quality.

Suppose that we have a Gaussian approximation of the belief state $p(M_t, \mathbf{C} \mid \mathbf{o}_{1:t-1})$. We wish to condition this belief state on a nonlinear observation o^i made by camera i. It is sufficient for us to consider the simpler problem of computing a Gaussian approximation to $p(M_t, C_i \mid o^i)$ from $p(M_t, C_i)$; this allows us to focus on the state of the object and the camera making the observation.[3] In Gaussian Quadrature techniques like the UKF, we compute this Gaussian approximation by first approximating $p(M_t, C_i, O^i)$, and then instantiating the observation $O^i = o^i$ (using exact Gaussian conditioning). The joint distribution $p(M_t, C_i, O^i)$ is approximated as follows. First, we select some small number of **integration points** in (M_t, C_i) space to characterize the prior $p(M_t, C_i)$; these typically include the mean and points along a confidence ellipse to characterize the uncertainty in the prior. Then we evaluate the nonlinear observation function $g(M_t, C_i)$ for each integration point to compute their images. The desired Gaussian approximation to the joint $p(M_t, C_i, O^i)$ is then computed by estimating its mean and covariance from the integration points and their images. In general, these techniques provide formal guarantees when the function f is a polynomial of bounded degree.

Unfortunately, as shown in Fig. 3(c), the Gaussian Quadrature approach does not provide effective linearization in the SLAT problem. The main cause of this problem is the periodicity of the angle ϕ, which represents the orientation of the camera. This periodicity in the projection function cannot be approximated well by a polynomial of bounded degree.

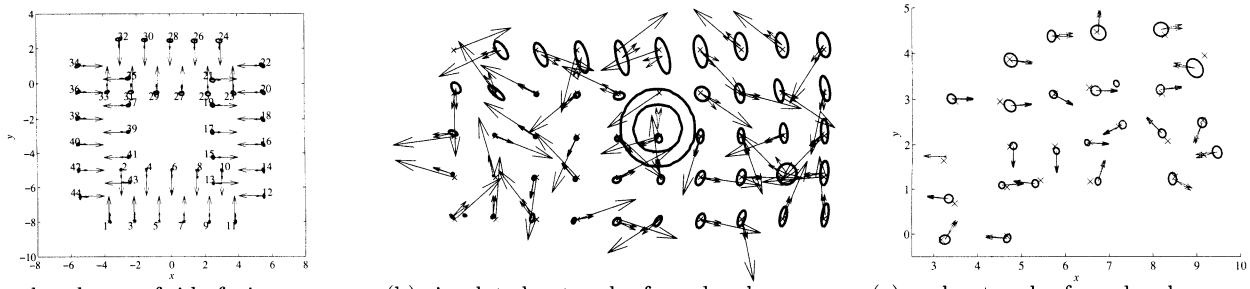
We address this problem with an approach we call **hybrid linearization**. By fixing the value of the angle ϕ in the projection operator, the periodicity problem is eliminated and our integration problem can be approximated well with Gaussian Quadrature methods. Building on this idea, we redefine our integration problem by selecting a number of integration points ϕ_j for the angle, and then we use Gaussian Quadrature to compute a Gaussian approximation of $p(M_t, C_i, O^i \mid \phi_j)$ for each ϕ_j.[4] (Note that ϕ_j is a component of the camera pose C_i.) The approximation of our joint distribution is then given by:

$$p(M_t, C_i, O^i) \approx \frac{\sum_j p(\phi_j) p(M_t, C_i, O^i \mid \phi_j)}{\sum_j p(\phi_j)}. \qquad (8)$$

[2]The more common *extended Kalman filter* uses a first-order Taylor expansion for linearization; Gaussian Quadrature methods are far more powerful.

[3]This follows from the fact that, based on our independence assumptions, $p(M_t, \mathbf{C} \mid \mathbf{o}_{1:t-1}, o^i) = p(M_t, C_i \mid o^i) \cdot p(\mathbf{C}_{\setminus i} \mid \mathbf{o}_{1:t-1})$, where $\mathbf{C}_{\setminus i}$ is the vector of poses for all cameras except for i.

[4]The integration points for ϕ_j are evenly spaced in the interval $[\bar{\phi} - \alpha, \bar{\phi} + \alpha]$, where $\bar{\phi}$ is the prior mean over the angle, and $\alpha = \min(3\sigma_\phi, \pi)$, where σ_ϕ is the prior standard deviation over the angle.



(a) simulated net. of side-facing cameras (b) simulated network of overhead cameras (c) real network of overhead cameras

Figure 4: Additional results from the centralized algorithm. Figures (a) and (b) demonstrate very good results on simulated networks with many cameras. In (b), where the cameras are overhead, the estimates are more uncertain because the object is observed less frequently. (c) shows the results of our algorithm when run on a real camera network of twenty-five cameras.

In this equation, each $p(M_t, C_i, O^i \mid \phi_j)$ is a Gaussian, making $p(M_t, C_i, O^i)$ a mixture of Gaussians. In order to approximate this mixture as a single Gaussian, we use a standard approach that finds the optimal Gaussian approximation by moment matching [6, §3.3].

The Gaussian Quadrature method described thus far improves the quality of the SLAT results, but these results are still not satisfactory. The reason for this is the mixture of Gaussians in Eq. (8) is often a complex, multi-modal distribution that cannot be approximated well by a single Gaussian. On the other hand, the distribution *after* the observation is instantiated is more focused and better approximated by a single Gaussian. This leads us to our **hybrid conditional linearization** technique, where the observation is instantiated in each mixture component:

$$p(M_t, C_i, o^i) \approx \frac{\sum_j p(\phi_j) p(M_t, C_i, o^i \mid \phi_j)}{\sum_j p(\phi_j)}. \qquad (9)$$

This mixture is typically much closer to a Gaussian than the one in Eq. (8); thus, when we approximate it by a Gaussian as above, the approximation is precise. As before, Gaussian quadrature methods and exact conditioning can be used to approximate the summands in Eq. (9) when the prior over C_i, M_t is sufficiently focused. When a camera makes its first observation, its pose prior is completely "uniform", which does not allow us to select integration points. Instead, we use the inverse projection function $g^{-1}(M_t, o^i, \delta^i, \phi_j)$ to approximate $p(M_t, C_i, o^i \mid \phi_j)$ as a marginal of

$$p(C_i, M_t, o^i, \delta^i \mid \phi_j) \propto \qquad (10)$$
$$p(C_i \mid M_t, o^i, \delta^i, \phi_j) p(M_t \mid o^i, \delta^i, \phi_j) p(\delta^i \mid o^i, \phi_j).$$

Here, $p(C_i \mid M_t, o^i, \delta^i, \phi_j)$ is deterministic and corresponds exactly to the inverse projection function. Furthermore, $p(M_t \mid \phi_j, o^i, \delta^i) = p(M_t)$, because the prior distribution of C_i is uninformative, and we approximate $p(\delta^i \mid \phi_j, o^i)$ as a normal distribution $\mathcal{N}(0, \sigma_\delta^2 I)$. We have found that the resulting hybrid conditional linearization yields excellent results for SLAT problems, as shown in Fig. 3(d).

5. EXPERIMENTAL RESULTS

In addition to the smaller tower scenario in Fig. 3(d), we evaluated our approach on several other larger simulated scenarios. We include two sample results here, Fig. 4(a) and Fig. 4(b). We omit the results for the absolute parameterization, because it performed very poorly in these larger scenarios. These results are best visualized with videos, we refer the reader to `http://www.cs.cmu.edu/~sfuniak/slat`.

The scenario in Fig. 4(a) includes 44 side facing cameras, tilted down about 35°, arranged along both walls in a square corridor. The object circles the loop twice. Note that all of the camera position and orientation estimates are within the estimated 95% confidence intervals. Thus, we are effectively representing both the estimate and the uncertainty.

The scenario in Fig. 4(b) consists of 50 downward-facing cameras, and the object circles the space. Again, we see that our estimates are almost all within the 95% confidence interval. Interestingly, camera 16 in the center only sees the object once; its posterior distribution should be ring-like. Since our ROP parameterization can capture such a structure, we see that the 95% confidence interval for this camera is in fact a ring.

We have also evaluated our approach on a real network of twenty-five overhead cameras. Here, a toy remote-controlled car was driven around a room carrying a colored marker, and a standard image processing algorithm was used to extract the center of the marker. Fig. 4(c) illustrates the solution we obtain. We see that our approach generates excellent pose estimates with real data.

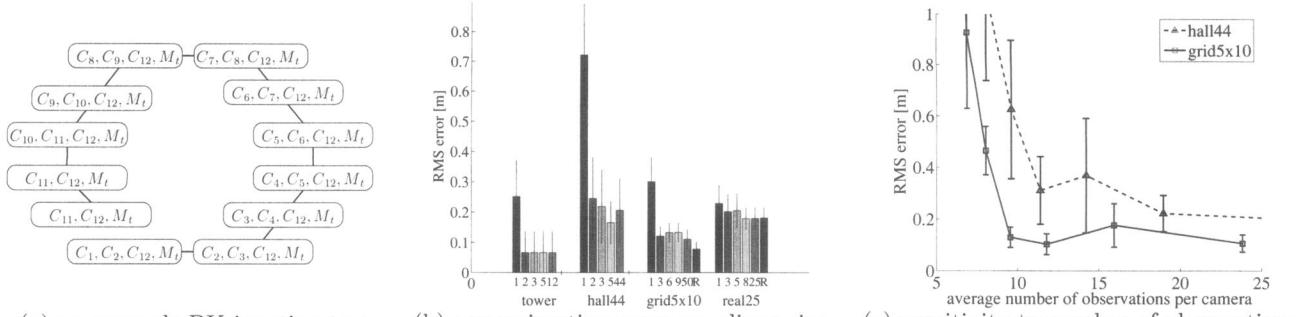
6. SCALING UP TO LARGE NETWORKS

The model we have developed so far would not scale well to very large camera networks for two reasons. First, the representation of the belief state (in terms of a mean vector and covariance matrix) requires space that is quadratic in the number of cameras (since correlations between all pairs of camera poses are maintained). In addition, these correlations must be updated with each observation, which makes the filter update a quadratic-time algorithm. This problem also occurs in the SLAM domain [10].

In this section, we give a brief overview of an approximation strategy that circumvents these difficulties and facilitates the distributed inference algorithm described below. This approximation strategy is an instance of the Boyen & Koller (BK) algorithm [1]. Rather than representing the belief state as a monolithic probability distribution over all state variables, the BK algorithm uses an approximation built out of marginals of that joint distribution:

$$\tilde{p}(M_t, \mathbf{C} \mid \mathbf{o}_{1:t-1}) = \frac{\prod_{\mathbf{C}_i} \tilde{p}(M_t, \mathbf{C}_i \mid \mathbf{o}_{1:t-1})}{\prod_{\mathbf{C}_{ij}} \tilde{p}(M_t, \mathbf{C}_{ij} \mid \mathbf{o}_{1:t-1})}, \qquad (11)$$

where $\mathbf{C}_i$ are subsets of the camera pose variables $\mathbf{C}$ and $\mathbf{C}_{ij} = \mathbf{C}_i \cap \mathbf{C}_j$. For the approximation to be well-defined, these subsets must be defined in terms of a data structure called a **junction tree** [2]. An example junction tree for



(a) an example BK junction tree (b) approximation error vs. clique size (c) sensitivity to number of observations

Figure 5: Using the Boyen & Koller (BK) algorithm. (a) An example BK junction tree for the problem in Fig. 3(a);the cliques are shown (but the separators are not). (b) Approximation error vs. clique size. The horizontal axis shows the four scenarios discussed in the previous sections. The bars correspond to solutions with increasing numbers of variables per clique. The fifth bar in each group corresponds to the solution of a Kalman Filter. We see that often, a very simple approximation suffices to obtain accurate results. The last bars in the `grid5x10` and `real25` examples (labeled 'R') show the performance of the algorithm [13], running on our data set with the same choice of model parameters. (c) Approximation error vs. number of observations.

the SLAT problem of Fig. 3(a) is shown in Fig. 5(a). Each node of the junction tree is annotated with a set of variables called a **clique**, and each edge is associated with another set of variables called its **separator**; each separator is the intersection of the two incident cliques. Junction trees satisfy a constraint called the **running intersection property**: if a variable is in two cliques, then it must also be in all cliques on the unique path between them. In our BK approximation Eq. (11), the cliques are $\{M_t\} \cup \mathbf{C}_i$ and the separators are $\{M_t\} \cup \mathbf{C}_{ij}$; thus, the object state variable M_t is in every clique and separator.

The intuition behind the BK approximation is that to avoid the cost of maintaining dependency information between all variables, we can instead maintain dependencies between small, overlapping subsets of variables. How, should we select these cliques to get the best approximation? If we examine the exact solution computed with the Kalman Filter, we notice that the strongest dependencies are among the variables of the cameras with overlapping views. This provides a good heuristic for choosing the BK approximation: we select the cliques to cover sets of cameras that are near one another, as in Fig. 5(a).[5] Furthermore, since the object state is strongly coupled to the camera poses (via the observations), we add M_t to all cliques and separators in the junction tree. In a real camera deployment, wireless radio range could be used to determine the couplings among the cameras. Alternatively, a technique called Thin junction tree filtering (TJTF) [10] could be used to automatically and adaptively select the approximation structure of the BK algorithm, without topological information.

The filter updates, estimation, prediction, and roll-up, can be implemented efficiently using the BK representation. When performing filtering, for each clique, we must compute the new clique marginal $\tilde{p}(M_{t+1}, \mathbf{C}_i \mid \mathbf{o}_{1:t})$ using the previous belief state, Eq. (11), the observation likelihoods from time step t, and the motion model for the object. This can be accomplished in a two-step process: first, the estimation phase is performed by multiplying the likelihoods into the belief state Eq. (11), as in Eq. (4), and then using an efficient dynamic programming algorithm in which each pair of adjacent cliques in the junction tree exchange information [2].

[5]Our cliques are selected using gross, imprecise topological information about the environment, not the precise locations of the cameras, which are determined by our approach.

The result of this process is a belief state of the form

$$\tilde{p}(M_t, \mathbf{C} \mid \mathbf{o}_{1:t}) = \frac{\prod_{\mathbf{C}_i} \tilde{p}(M_t, \mathbf{C}_i \mid \mathbf{o}_{1:t})}{\prod_{\mathbf{C}_{ij}} \tilde{p}(M_t, \mathbf{C}_{ij} \mid \mathbf{o}_{1:t})}, \tag{12}$$

i.e., all marginals have been conditioned on the new observation o_t. The prediction and roll-up phases of filtering can now be implemented very efficiently: for each clique (and separator) marginal, we independently multiply in the object motion model and marginalize out the old object state:

$$\begin{aligned} &\tilde{p}(M_{t+1}, \mathbf{C}_i \mid \mathbf{o}_{1:t}) \\ &= \int p(M_{t+1} \mid M_t = m_t) \tilde{p}(M_t = m_t, \mathbf{C}_i \mid \mathbf{o}_{1:t}) \mathrm{d}m_t. \end{aligned} \tag{13}$$

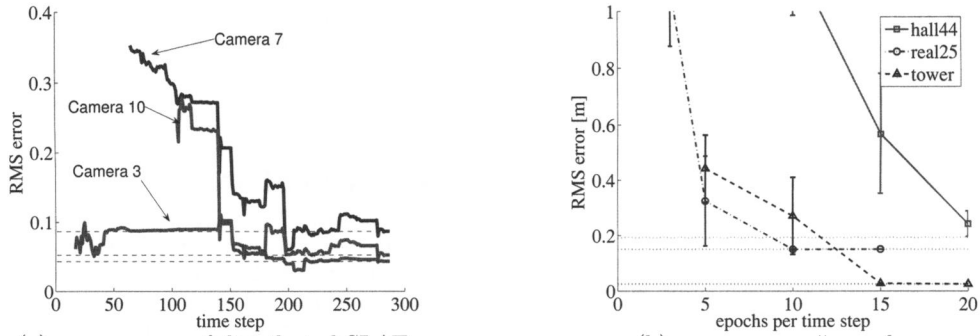
In the SLAT problem, we have found that the BK approximation is excellent, and yields almost no approximation error. In Fig. 5(b), we show the result of solving several SLAT scenarios using a BK approximation, for different clique sizes. We also compared the performance of our approach to the calibration algorithm in [13] on our scenarios with overhead cameras (at the time of writing, the implementation of their algorithm did not support side-facing cameras). We see that, in these scenarios, our online approach and the offline optimization approach of Rahimi *et al.* provide solutions of comparable quality.

Fig. 5(c) shows the performance of our algorithm as we vary the number of observations made by the cameras. With roughly 10-12 observations per camera, the algorithm obtains accurate estimates. Only a few of these observations are made in the regions where the camera views overlap.

7. DISTRIBUTED FILTERING

In the previous section we described the BK filtering algorithm, which can be used to decompose the belief state into smaller pieces while introducing minimal approximation error. In this section, we give an overview of our distributed SLAT approach, which uses the BK representation to perform robust, distributed filtering. For further details on the algorithm, refer to the companion technical report [3].

Our approach builds on a general architecture for robust, distributed inference in sensor networks [9], which can be applied to many types of inference problems, including regression (or function fitting) problems, probabilistic inference problems, and control problems. In this architecture,



(a) convergence of distributed SLAT (b) accuracy v. # epochs

Figure 6: (a) RMS of estimated poses versus number of time steps, in the tower scenario, using our distributed algorithm; (b) RMS of the resulting solution as the number of epochs per time step increases. Horizontal lines indicate the quality of the corresponding centralized solution.

each node of the network starts with local pieces of information, which may come from its local measurements and from a global model. Then, the nodes coordinate to build a spanning tree overlay and form a distributed data structure called a **network junction tree**, where each camera node is associated with a subset of the variables in the inference problem. Inference is then performed with a message passing algorithm in which adjacent nodes in the spanning tree exchange information about their common variables. This architecture can solve inference problems robustly in the face of lost messages and failed nodes, and the algorithm can recover from interference that partitions the network [9].

Distributed probabilistic inference problems can be solved using this architecture in combination with the **robust message passing algorithm** described in [11]. This algorithm provides robustness properties beyond those guaranteed by the architecture. Not only does the network compute the right answer after message passing has converged, but even before convergence, or when nodes fail, each node in the network can compute a principled approximation to the correct answer. In our SLAT setting, this guarantee corresponds to a principled approximation of a node's posterior distribution, given the images observed by the cameras that are within communication range. This guarantee ensures that even when communication is very unstable, making global coordination impossible, nodes can collaborate with other nearby nodes to obtain informative approximate solutions.

Unfortunately, we cannot directly apply robust message passing to our SLAT inference problem. The robust message passing algorithm is designed to solve *static* inference problems, where the random variables do not change over time. In the SLAT problem, however, we must solve a *dynamic* inference problem—a filtering problem—because the object's location M_t changes over time. In this section, we describe a new algorithm for performing *robust, distributed filtering* which is based upon both the architecture of [9] and the robust message passing algorithm of [11].

Recall from Sec. 6 that the BK filtering algorithm represents the belief state $\tilde{p}(M_t, \mathbf{C} \mid \mathbf{o}_{1:t-1})$ using a set of clique marginals, and that these cliques are defined in terms of a junction tree like that in Fig. 5(a). We will call this junction tree the **external junction tree** to distinguish it from the network junction tree described above, which is built by the camera nodes as the distributed algorithm runs. Thus, while the purpose of the network junction tree is to pass messages about the variables assigned to each node, the external junction tree represents the conditional dependencies among the variables. In our algorithm, each node in the network maintains a portion of this belief state, i.e., a subset of the external junction tree's clique marginals. In each filtering update, the nodes collaborate to update their clique marginals with each other's observations, and to reflect the evolution of the process—in the case of SLAT, the motion of the object. After the distributed filtering update has converged, the network has implemented an exact BK update, and each node has advanced each of its clique marginals from $\tilde{p}(M_t, \mathbf{C}_i \mid \mathbf{o}_{1:t-1})$ to $\tilde{p}(M_{t+1}, \mathbf{C}_i \mid \mathbf{o}_{1:t})$. At this point, the process repeats, to once again update the belief state with new observations. At any time during the distributed filtering algorithm, each node has immediate access to marginals of the joint belief state; in SLAT, these marginals enable each camera to estimate its own pose, for example.

Initialization of the distributed algorithm. To initialize the distributed filtering algorithm, each clique of the external junction tree—which is a set of camera pose variables, plus the initial object state M_1—is distributed to one or more nodes of the camera network. Each camera is given at least one clique that contains its pose variable, and each clique is given to at least one camera. These cliques may be distributed redundantly to increase robustness in the face of node failures, just as in the robust message passing algorithm [11]. Using these cliques, the cameras initialize the clique priors of the first belief state, which are uniform. In addition to receiving these cliques, each camera node also receives the motion model of the object of Eq. (1), so that it can perform the prediction step of the filtering algorithm.

The estimation phase. In the estimation phase of the filtering algorithm, the nodes must collaborate to condition their clique marginals on the images observed in the current time step. That is, the camera nodes start with a distributed representation of the BK belief state Eq. (11), and they must collaborate so as to obtain a distributed representation of Eq. (12). This is a challenging problem, but one that has already been solved: this type of inference is exactly the sort of inference performed by the robust message passing algorithm, where we wish to compute a posterior from a prior and some observed evidence. In estimation, the belief state at the current time step plays the role of the prior, and the goal is to compute Eq. (12), which is the posterior, conditioned on the current observations. We can therefore run the robust message passing algorithm without change to implement the estimation phase of the filtering update.

The prediction and roll–up phases. In comparison to the estimation phase, the prediction and roll–up phases are very simple, because they require no coordination. In order to reflect the motion of the object, each camera node independently applies the update in Eq. (13) to each one of its local cliques. The old clique marginals are then replaced by the new clique marginals, yielding a new distributed representation of the belief state in Eq. (11).

Aligning inconsistent beliefs. The distributed algorithm requires us to face issues that do not arise in static inference problems. In dynamic settings, communication latencies or failures may prevent information from reaching all parts of the network in a timely fashion. If the estimation process fails to converge before the end of the time step, the robust message passing algorithm guarantees that each node has a principled approximation, but these approximations may not be consistent with each other. For example, nodes may not agree on the posterior distribution of the object state M_t, because they have not had access to the same observations. In this case, performing the prediction and roll–up computations will propagate these inconsistencies into the next belief state. This inconsistency is theoretically troublesome because it becomes impossible to characterize the belief state of the network as a whole. To correct such inconsistencies, our distributed filtering algorithm must **align** the beliefs at different nodes, so that they are consistent with some joint distribution. The key idea behind our alignment algorithm is to note that if every pair of adjacent nodes in the network junction tree agree on the distribution of the variables they share, then all nodes are consistent with a global distribution. Our alignment algorithm guarantees that if a single filtering step successfully converges, then all inconsistencies in the belief state due to communication failures in the past are resolved, and the network represents a well-defined joint distribution over the current state of the process. A full description of the algorithm is presented in the companion technical report [3].

8. DISTRIBUTED SLAT EXPERIMENTS

Since do not have a wireless camera network at this time, we evaluated our distributed algorithm using the event-based distributed-systems simulator described in [11] to obtain a qualitative evaluation of our distributed SLAT approach. We simulated link qualities using an exponentially-decaying function of the squared distance between nodes, where nearby cameras (about 1 meter apart) had about 20% packet loss.

Our first experiment applies our distributed SLAT algorithm to the tower scenario. Fig. 6(a) shows that our distributed algorithm converges to the same solution as the centralized one. Note that the convergence curve is different for different cameras, since their estimate is uninformative until they first observe the object. Interestingly, in this figure, we can clearly see a "loop-closing" effect [10] after about 150 time steps: the first camera to observe the object is certain about its location; when the object returns to the field of view of this camera, its position becomes more certain, and the estimates of all cameras become more accurate.

To illustrate the effect of information propagation throughout the network, in Fig. 6(b), we evaluate the quality of the final solution of the distributed algorithm as a function of the number of epochs in each time step. In each epoch, each node attempts to send any new messages it has queued up; about 30% of messages are lost due to lossy communication. These results show that with about 15–20 epochs per time step, our distributed algorithm converges to the same solution as the centralized one. For the real data experiment, the algorithm converges much quicker: due to the small size of the network, fewer messages are needed to propagate the information around the network.

9. CONCLUSION

This paper has demonstrated that large camera networks can be automatically calibrated by tracking a moving object. We presented two techniques, *relative over-parameterization* and *hybrid conditional linearization*, that enable an efficient Kalman filter solution to the SLAT problem, in spite of its complexity and nonlinearity. Our approach obtains the estimate of a camera's pose, as well as the uncertainty in the estimate. We demonstrated that the BK algorithm gives an excellent approximation to the Kalman filter solution, and we used the BK representation as the basis of a scalable *distributed filtering algorithm* that solves the SLAT problem robustly; even in the presence of communication failures, the distributed algorithm converges to good SLAT solutions.

Acknowledgments. This research was supported by grant CNS-0428738 NSF ITR: Synthetic Reality; S. Funiak was supported by the Intel Research Scholar Program. Thanks to A. Rahimi for his implementation of [13] and to J. Huang and S. Schlosser for their help with the camera testbed.

10. REFERENCES

[1] X. Boyen and D. Koller. Tractable inference for complex stochastic processes. In *Proc. UAI*, 1998.
[2] R. Cowell, P. Dawid, S. Lauritzen, and D. Spiegelhalter. *Probabilistic Networks and Expert Systems*. Springer, New York, NY, 1999.
[3] S. Funiak, C. Guestrin, M. Paskin, and R. Sukthankar. Robust probabilistic filtering in distributed systems. Technical Report CMU-CALD-05-111, Carnegie Mellon Univ., 2005.
[4] A. Ihler, J. Fisher, R. Moses, and A. Willsky. Nonparametric belief propagation for self-calibration in sensor networks. In *Proc. IPSN*, 2004.
[5] S. Khan and M. Shah. Consistent labeling of tracked objects in multiple cameras with overlapping fields of view. *IEEE PAMI*, 25(10), 2003.
[6] U. Lerner. *Hybrid Bayesian Networks for Reasoning about Complex Systems*. PhD thesis, Stanford, 2002.
[7] W. Mantzel, H. Choi, and R. Baraniuk. Distributed Camera Network Localization. In *Asilomar Conference on Signals, Systems, and Computers*, 2004.
[8] D. Nistér. *Automatic dense reconstruction from uncalibrated video*. PhD thesis, Royal Inst. of Tech., 2001.
[9] M. Paskin, C. Guestrin, and J. McFadden. A robust architecture for inference in sensor networks. In *Proc. IPSN*, 2005.
[10] M. A. Paskin. Thin junction tree filters for simultaneous localization and mapping. In *Proc. IJCAI*, 2003.
[11] M. A. Paskin and C. E. Guestrin. Robust probabilistic inference in distributed systems. In *Proc. UAI*, 2004.
[12] M. Pollefeys, L. Van Gool, M. Vergauwen, F. Verbiest, K. Cornelis, J. Tops, and R. Koch. Visual modeling with a hand-held camera. *IJCV*, 59(3), 2004.
[13] A. Rahimi, B. Dunagan, and T. Darrell. Simultaneous calibration and tracking with a network of non-overlapping sensors. In *CVPR*, 2004.
[14] S. Soatto and P. Perona. Reducing "structure from motion": A general framework for dynamic vision part 1: Modeling. *IEEE PAMI*, 20(9), 1998.
[15] C. Stauffer and K. Tieu. Automated multi-camera planar tracking correspondence modeling. In *CVPR*, 2003.
[16] S. Thrun. Affine structure from sound. In *NIPS*, 2005.
[17] E. A. Wan and R. van der Merwe. The unscented Kalman filter for nonlinear estimation. In *Proc. Adaptive Sys. for Signal Proc., Comm. and Control*, 2000.
[18] K. Whitehouse, C. Karlof, A. Woo, F. Jiang, and D. Culler. The effects of ranging noise on multihop localization: an empirical study. In *Proc. IPSN*, 2005.