Snapshot: A Self-Calibration Protocol for Camera Sensor Networks

Xiaotao Liu, Purushottam Kulkarni and Prashant Shenoy
Department of Computer Science
University of Massachusetts, Amherst
Amherst, MA 01003
Email: {xiaotaol, purukulk, shenoy}@cs.umass.edu

Abstract— **A camera sensor network is a wireless network of cameras that are designed for ad-hoc deployment. The camera sensors in such a network need to be properly calibrated by determining their location, orientation, and range. This paper presents *Snapshot*, an automated calibration protocol that is explicitly designed and optimized for camera sensor networks. *Snapshot* uses the inherent imaging abilities of the cameras themselves for calibration. Further, unlike some vision-based techniques that require tens of reference points for calibration, *Snapshot* can determine the location and orientation of a camera sensor using only four reference points. Our techniques are based on principles from optics and geometry and are designed to work with low-fidelity, low-power camera sensors that are typical in sensor networks. As experimental evaluation of our prototype implementation shows that *Snapshot* yields an error of 1-2.5 degrees when determining the camera orientation and 5-10cm when determining the camera location. We show that this is a tolerable error in practice since a *Snapshot*-calibrated sensor network can track moving objects to within 11cm of their actual locations. Finally, our measurements indicate that *Snapshot* can calibrate a camera sensor within 20 seconds, enabling it to calibrate a sensor network containing tens of cameras within minutes.**

I. Introduction

A. Motivation

Recent advances in embedded systems technologies have made the design of camera sensor networks a reality. A camera sensor network is an ad-hoc wireless network of low-power imaging sensors (cameras) that are connected to networked embedded controllers. Today, available camera sensors range from tiny, low-power cameras such as CMUcams and Cyclops to "cell-phone-class" cameras and from inexpensive web-cams to high-resolution pan-tilt-zoom cameras.

Typical applications of camera sensor networks include active monitoring of remote environments and surveillance tasks such as object detection, recognition, and tracking. These applications involve acquisition of images and video from multiple camera sensors and real-time processing of this data for recognition, tracking, and camera control. Video acquisition and processing involves interaction and coordination between multiple cameras, for instance, to hand-off tracking responsibilities for a moving object from one camera to another. Precise calibration of camera sensors is a necessary pre-requisite for such interactions and coordination. Calibration of a camera sensor network involves determining the location, orientation, and range of each camera sensor in three dimensional space as well as the overlap and spatial relationships between nearby cameras.

Automated camera calibration is well studied in the computer vision community [20], [21], [22], [24]. Many of these techniques are based on the classical Tsai method—they require a user to specify reference points on a grid whose true locations are known in the physical world and use the projection of these points on the camera image plane to determine camera parameters. However, such vision-based calibration techniques may not be directly applicable to camera sensor networks for the following reasons. First, the vision-based systems tend to use high-resolution cameras as well as high-end workstations for image and video processing; consequently, calibration techniques can leverage the availability of high-resolution images and abundance of processing power. Neither assumption is true in sensor networks. Such networks may employ low-power, low-fidelity cameras such as the CMUcam [16] or Cyclops [12] that have coarse-grain imaging capabilities; at best, a mix of low-end and a few high-end cameras can be assumed for such environments. Further, the cameras may be connected to nodes such as the Crossbow Motes [13] or Intel Stargates [18] that have one or two orders of magnitude less computational resources than PC-class workstations. Calibration techniques for camera sensor networks need to work well with low-resolution cameras and should be computationally efficient.

Second, vision-based calibration techniques have been designed to work with a single camera or a small group of cameras. In contrast, a camera sensor network may comprise tens or hundreds of cameras and calibration techniques will need to scale to these larger environments. Further, camera sensor networks are designed for ad-hoc deployment, for instance, in environments with disasters such as fires or floods. Since quick deployment is crucial in such environments, it is essential to keep the time required for calibrating the system to a minimum. Thus, calibration techniques need to be scalable and designed for quick deployment.

Third, vision-based camera calibration techniques are designed to determine both intrinsic parameters (e.g., focal length, lens distortion, principal point) and extrinsic parameters (e.g., location and orientation) of a camera. Due to the large number of unknowns, the calibration process typically involves many tens of measurements of reference points and is computationally intensive. In contrast, calibrating a camera sensor network involves only determining external parameters such as camera location and orientation, and may be amenable to simpler, more efficient techniques that are better suited to resource-constrained sensor platforms.

Automated localization techniques are a well-studied problem in the sensor community and a slew of techniques have been proposed. Localization techniques employ beacons (e.g., IR [1], ultrasound [2], RF [3]) and use sophisticated triangulation techniques to determine the location of a node. Most of these technique have been designed for general-purpose sensor networks, rather than camera sensor networks in particular. Nevertheless, they can be employed during calibration, since determining the node location is one of the tasks performed during calibration. However, localization techniques are by themselves not sufficient for calibration. Cameras are *directional* sensors and camera calibration also involves determining other parameters such as the orientation of the camera (where a camera is pointing) as well as its range (what it can see). In addition, calibration is also used to determine overlap between neighboring cameras. Consequently, calibration is a harder problem than pure localization.

The design of an automated calibration technique that is cost-effective and yet scalable, efficient, and quickly deployable is the subject matter of this paper.

B. Research Contributions

In this paper, we propose *Snapshot* a novel wireless protocol for calibrating camera sensor networks. *Snapshot* advances prior work in vision-based calibration and sensor localization in important ways. Unlike vision-based techniques that require tens of reference points for calibration and impose restrictions on the placement of these points in space, *Snapshot* requires only four reference points to calibrate each camera sensor and allows these points to be randomly chosen without restrictions. Both properties are crucial for sensor networks, since fewer reference points and fewer restrictions enable faster calibration and reduce the computational overhead for subsequent processing. Further, unlike sensor localization techniques that depend on wireless beacons, *Snapshot* does not require any specialized positioning equipment on the sensor nodes. Instead, it leverages the inherent picture-taking abilities of the cameras and the onboard processing on the sensor nodes to calibrate each node. Our results show *Snapshot* yields accuracies that are comparable those obtained by using positioning devices such as ultrasound-based Cricket on each node.

Our techniques can be instantiated into a simple, quick and easy-to-use wireless calibration protocol—a wireless calibration device is used to define reference points for each camera sensor, which then uses principles from geometry, optics and elementary machine vision to calibrate itself. When more than four reference points are available, a sensor can use median filter and maximum likelihood estimation techniques to improve the accuracy of its estimates.

We have implemented *Snapshot* on a testbed of CMU-cam sensors connected to wireless Stargate nodes. We have conducted a detailed experimental evaluation of *Snapshot* using our prototype implementation. Our experiments yield the following key results:

1) *Feasibility:* By comparing the calibration accuracies of low and high-resolution cameras, we show that it is feasible to calibrate low-resolution cameras such as CMU-cams without a significant loss in accuracy.
2) *Accuracy:* We show that *Snapshot* can localize a camera to within few centimeters of its actual location and determine its orientation with a median error of 1.3–2.5 degrees. More importantly, our experiments indicate that this level of accuracy is sufficient for tasks such as object tracking. We show that a system calibrated with *Snapshot* can localize an external object to within 11 centimeters of its actual location, which is adequate for most tracking scenarios.
3) *Efficiency:* We show that the *Snapshot* algorithm can be implemented on Stargate nodes and have running times in the order of a few seconds.
4) *Scalability:* We show that *Snapshot* can calibrate a camera sensor in about 20 seconds on current hardware; Since a human needs to only specify a few reference points using the wireless calibration device—a process that takes a few seconds per sensor—*Snapshot* can scale to networks containing tens of camera sensors.

The rest of this paper is structured as follows. Section II presents the Background and Problem Formulation of our problem. Sections III, IV and V present the design of *Snapshot* design, its instantiation into a protocol and an application. The implementation of *Snapshot* is described in Section VI and the experimental evaluation in Section VII. Section VIII describes related work and Section IX presents our conclusions.

II. Background and Problem Formulation

A camera sensor network is defined to be a wireless network of camera sensors, each connected to an embedded controller. A typical realization of a camera sensor node consists of a low-power camera such as the CMUcam [16] or Cyclops [12] connected to an embedded sensor platform such as the Berkeley Mote [13] or the Intel Stargate [18]. The sensor platform consist of a programmable microprocessor, memory, and a wireless interface for communication. Not all cameras in the system are homogeneous; in general, a small number of higher resolution cameras may be deployed to assist the low-fidelity cameras in performing their tasks. Thus, a mix of camera sensors, each with different capabilities may be expected.

Consider an ad-hoc deployment of N heterogeneous camera sensor nodes in an environment. An ad-hoc deployment implies that cameras are quickly placed without *a priori* planning. Given such an ad-hoc deployment, the location, orientation and the range of each camera sensor needs to be automatically determined. The goal of our work is to design a wireless protocol to automatically derive these parameters for each camera node. Specifically, the calibration protocol needs to determine the (x, y, z) coordinates of each camera, which is defined as the coordinates of the center of the camera lens. The protocol also needs to determine the camera orientation along the three axes, namely the *pan*, *tilt* and *roll* of the camera. The pan angle α is defined to be the camera rotation along the Z axis, the tilt angle β is the rotation along the X axis, and the roll angle γ is the rotation along the Y axis. Finally, the protocol needs to determine the field of view of each camera (i.e., what it can see) and use the field of view to determine the degree of overlap in neighboring cameras (i.e., the common areas visible to both cameras).

Our work assumes that the focal length f of camera lens is known to the calibration protocol. This is a reasonable assumption since lens parameters are typically published in the camera specifications by the manufacturer or they can be estimated offline for each camera prior to deployment [20], [21]. Further, sensor nodes are assumed to lack specialized positioning devices such as GPS receivers. Instead, our goal is to devise a protocol that exploits the inherent imaging abilities of each camera and the onboard processing on each sensor node to determine the above calibration parameters.

III. Snapshot Design

The basic *Snapshot* protocol involves taking pictures of a small randomly-placed calibration device. To calibrate each camera sensor, at least four pictures of the device are necessary, and no three positions of the device must lie along a straight line. Each position of the calibration device serves as a reference point; the coordinates of each reference point are assumed to be known and can be automatically determined by equipping the calibration device with a locationing sensor (e.g., GPS or ultra-sound Cricket receiver). In what follows, we describe how *Snapshot* uses the pictures of the calibration device together with the coordinates of each position to estimate the camera location, orientation, and range. We also discuss how the estimates can be refined when additional reference points are available.

A. Camera Location Estimation

We begin with the intuition behind our camera location estimation approach. Consider a camera sensor C whose coordinates need to be determined. Suppose that four reference points R_1, R_2, R_3 and R_4 are given along with their coordinates for determining the camera location. No assumption is made about the placement of these points in the three dimensional space, except that these points be in visual range of the camera and that no three of them lie along a straight line. Consider the first two reference points R_1 and R_2 as shown in Figure 1. Suppose that point objects placed at R_1 and R_2 project an image of P_1 and P_2, respectively, in the camera's image plane as shown in Figure 1. Further, let θ_1 be the angle incident by the the reference points on the camera. Since θ_1 is also the angle incident by P_1 and P_2 on the camera lens, we assume that it can be computed using elementary optics (as discussed later). Given θ_1, R_1 and R_2, the problem of finding the camera location reduces to finding a point in

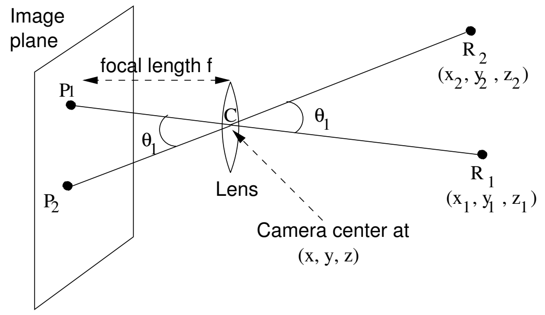
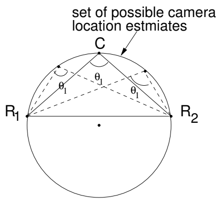
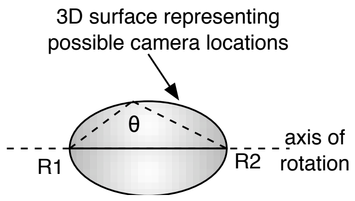


Fig. 1. Projection of reference points on the image plane through the lens.



(a) Arc depicting possible solutions in two dimensions.



(b) Football-like surface of possible solutions in three dimensions.

Fig. 2. Geometric representation of possible camera locations.

space where R_1 and R_2 impose an angle of θ_1. With only two reference points, there are infinitely many points where R_1 and R_2 impose an angle of θ_1. To see why, consider Figure 2(a) that depicts the problem in two dimensions. Given R_1 and R_2, the set of possible camera locations lies on the arc R_1CR_2 of a circle such that R_1R_2 is a chord of the circle and θ_1 is the angle incident by this chord on the circle. From elementary geometry, it is known that a chord of a circle inscribes a constant angle on any point on the corresponding arc. Since we have chosen the circle such that chord R_1R_2 inscribes an angle of θ_1 on it, the camera can lie on any point on the arc R_1CR_2. This intuition can be generalized to three dimensions by rotating the arc R_1CR_2 in space with the chord R_1R_2 as the axis (see Figure 2(b)) . Doing so yields a three dimensional surface of possible camera locations. The nature of the surface depends on the value of θ_1: the surface is shaped like a football when $\theta_1 > 90°$, is a sphere when $\theta_1 = 90°$, and a double crown when $\theta_1 < 90°$. It is clear that the camera can lie on any point of this surface.

Next, consider the third reference point R_3. If we consider reference points R_1 and R_3, we obtain another surface that consists of all camera possible locations such that R_1R_3 impose a known angle θ_2 on all points of this surface. Since the camera must lie on both surfaces, it follows that the set of possible locations is given by the intersection of these two surfaces. The intersection of two surfaces is a closed curve and the set of possible camera locations is reduced to any point on this curve.

Finally, if we consider the pair of reference points R_2 and R_3, we obtain a third surface of all possible camera locations. The intersection of the first surface and the third yields a second curve of possible camera locations. It follows that the

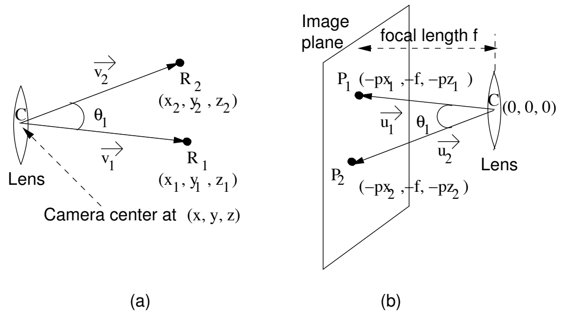


Fig. 3. Vector representation of reference points and their projections.

camera location lies on the intersection of these two curves, and two curves can intersect in multiple points. The number of possible camera locations can be reduced further to at most 4 by introducing the fourth reference point R_4.

Although 4 reference points give us up to 4 possible camera locations, we observe that, in reality, only one of these locations can generate the same projections as R_1, R_2, R_3, and R_4 on the image plane. Using elementary optics, it is easy to eliminate the false solutions and determine the true and unique location of the camera.

With this intuition, we now present the details of our technique. Consider a camera C placed at coordinates (x, y, z), and four reference points $R_1, ..., R_4$ with coordinates $(x_1, y_1, z_1) \dots (x_4, y_4, z_4)$. with each of these reference point defines a vector. For instance, as shown in Figure 3(a), the line joining C and R_1 defines a vector $\overrightarrow{CR_1}$, denoted by $\vec{v_1}$. The components of v_1 are given by

$$\vec{v_1} = \overrightarrow{CR_1} = \{x_1 - x, y_1 - y, z_1 - z\}$$

Similarly, the vector joining points C and R_i, denoted by $\vec{v_i}$, is given as

$$\vec{v_i} = \overrightarrow{CR_i} = \{x_i - x, y_i - y, z_i - z\} \quad 1 \leq i \leq 4$$

As shown in Figure 3(a), let θ_1 denote the angle between vectors $\vec{v_1}$ and $\vec{v_2}$. The dot product of vectors $\vec{v_1}$ and $\vec{v_2}$ is given as

$$\vec{v_1} \cdot \vec{v_2} = |\vec{v_1}||\vec{v_2}| \cos \theta_1 \tag{1}$$

By definition of the dot product,

$$\vec{v_1} \cdot \vec{v_2} = (x_1 - x)(x_2 - x) + (y_1 - y)(y_2 - y) + (z_1 - z)(z_2 - z) \tag{2}$$

The magnitude of vector $\vec{v_1}$ is given as

$$|\vec{v_1}| = \sqrt{(x_1 - x)^2 + (y_1 - y)^2 + (z_1 - z)^2}$$

The magnitude of $\vec{v_2}$ is defined similarly. Substituting these values into Equation 2,we get

$$\cos(\theta_1) = \frac{(x_1 - x)(x_2 - x) + (y_1 - y)(y_2 - y) + (z_1 - z)(z_2 - z)}{|\vec{v_1}| \cdot |\vec{v_2}|} \tag{3}$$

Let θ_2, through θ_6 denote the angles between vectors $\vec{v_1}$ and $\vec{v_3}$, $\vec{v_1}$ and $\vec{v_4}$, $\vec{v_2}$ and $\vec{v_3}$, $\vec{v_2}$ and $\vec{v_4}$ and $\vec{v_3}$ and $\vec{v_4}$ respectively. Similar expressions can be derived for θ_2, $\theta_3, \dots$ θ_6.

The angles θ_1 through θ_6 can be computed using elementary optics and vision, as discussed next. Given these angles and the coordinates of the four reference points, the above expressions yield six quadratic equations with three unknowns: x,y, and z. A non-linear solver can be used to numerically solve for these unknowns.

Estimating θ_1 through θ_6: We now present a technique to compute the angle between any two vectors $\vec{v_i}$ and $\vec{v_j}$. Consider any two reference points R_1 and R_2 as shown in Figure 3 (a). Figure 3 (b) shows the projection of these points through the camera lens onto the image plane. The image plane in a digital camera consists of a CMOS sensor that takes a picture of the camera view. Let P_1 and P_2 denote the projections of the reference points on the image plane as shown in the Figure 3(b), and let f denote the focal length of the lens. For simplicity, we define all points with respect to the camera's coordinate system: the center of the lens is assumed to be the origin in this coordinate system. Since the image plane is at a distance f from the lens, all points on the image plane are at a distance f from the origin. By taking a picture of the reference points, the coordinates of P_1 and P_2 can be determined. These are simply the pixel coordinates where the reference points project their image on the CMOS sensor; these pixels can be located in the image using a simple vision-based object recognition technique.[1] Let the resulting coordinates of P_1 and P_2 be $(-px_1, -f, -pz_1)$ and $(-px_2, -f, -pz_2)$ respectively. We define vectors $\vec{u_1}$ and $\vec{u_2}$ as lines joining the camera (i.e., the origin C) to the points P_1 and P_2. Then, the angle θ_1 between the two vectors $\vec{u_1}$ and $\vec{u_2}$ can be determined by taking the dot product of them.

$$\cos(\theta_1) = \frac{\vec{u_1} \cdot \vec{u_2}}{|\vec{u_1}||\vec{u_2}|}$$

The inverse cosine transform yields θ_1, which is also the angle incident by the original reference points on the camera.

Using the above technique to estimate θ_1–θ_6, we can then solve our six quadratic equations using a non-linear optimization algorithm [6] to estimate the camera location.

B. Camera Orientation Estimation

We now describe the technique employed by *Snapshot* to determine the camera's orientation along the three axes. We assume that the camera location has already been estimated using the technique in the previous section. Given the camera location (x, y, z), our technique uses three reference points to determine the pan, tilt, and roll of the camera. Intuitively, given the camera location, we need to align the camera in space so that the three reference points project an image at the same location as the pictures takes by the camera. Put another way, consider a ray of light emanating from each reference point. The camera needs to be aligned so that each ray of light pierces the image plane at the same pixel where the image of that reference point is located. One reference point is sufficient to determine the pan and tilt of the camera using this technique

[1]In *Snapshot* the calibration device contains a colored LED and the vision-based recognizer must locate this LED in the corresponding image.

and three reference point are sufficient to uniquely determine all three parameters: pan, tilt and roll. Our technique uses the actual coordinates of three reference points and the pixel coordinates of their corresponding images to determine the unknown rotation matrix R that represents the pan, tilt and roll of the camera.

Assume that the camera is positioned at coordinates (x, y, z) and that the camera has a a pan of α degrees, a tilt of β degrees, a roll of γ degrees The pan, tilt and roll rotations can be represented as matrices, and can be used to calculate locations of points in the camera's coordinate space. The composite matrix for the pan, tilt and roll rotations of the camera that results in its orientation is given by

$$\begin{aligned}\mathbf{R} &= \begin{bmatrix} \cos(\gamma) & 0 & \sin(\gamma) \\ 0 & 1 & 0 \\ -\sin(\gamma) & 0 & \cos(\gamma) \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\beta) & \sin(\beta) \\ 0 & -\sin(\beta) & \cos(\beta) \end{bmatrix} \times \\ &\quad \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 0 \end{bmatrix} \\ &= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad (4)\end{aligned}$$

If an object is located at (x_i, y_i, z_i) in the world coordinates, the object's location in the camera coordinates $(x_i^{'}, y_i^{'}, z_i^{'})$ can be computed via Equation 5.

$$\begin{bmatrix} x_i^{'} \\ y_i^{'} \\ z_i^{'} \end{bmatrix} = \mathbf{R} \times \begin{bmatrix} x_i - x \\ y_i - y \\ z_i - z \end{bmatrix} \quad (5)$$

where the composite rotation matrix $\mathbf{R}$ is given by Equation 4.

Intuitively, we can construct and solve a set of linear equations (see Equation 6) where (x_1, y_1, z_1), (x_2, y_2, z_2), and (x_3, y_3, z_3) are the world coordinates of 3 reference points, and $(x_1^{'}, y_1^{'}, z_1^{'})$, $(x_2^{'}, y_2^{'}, z_2^{'})$, and $(x_3^{'}, y_3^{'}, z_3^{'})$ are the corresponding camera coordinates to estimate $\mathbf{R}$, and then estimate α, β, and γ from $\mathbf{R}$. It is easy to see that as these three reference points are not co-linear, the matrix $\begin{bmatrix} x_1 - x & y_1 - y & z_1 - z \\ x_2 - x & y_2 - y & z_2 - z \\ x_3 - x & y_3 - y & z_3 - z \end{bmatrix}$ is a non-singular matrix, and hence, the three sets of linear equations in Equation 6 have unique solution for $\mathbf{R}^T$.

$$\begin{bmatrix} x_1 - x & y_1 - y & z_1 - z \\ x_2 - x & y_2 - y & z_2 - z \\ x_3 - x & y_3 - y & z_3 - z \end{bmatrix} \times \mathbf{R}^T = \begin{bmatrix} x_1^{'} & y_1^{'} & z_1^{'} \\ x_2^{'} & y_2^{'} & z_2^{'} \\ x_3^{'} & y_3^{'} & z_3^{'} \end{bmatrix} \quad (6)$$

As shown in Figure 4, an object's location in the camera coordinates and the projection of the object on the image plane have the following relation:

$$\begin{bmatrix} x_i^{'} \\ y_i^{'} \\ z_i^{'} \end{bmatrix} = \frac{D_i}{D_p} \times \begin{bmatrix} px_i \\ f \\ pz_i \end{bmatrix} \quad (7)$$

where:

$D_i = \sqrt{(x_i - x)^2 + (y_i - y)^2 + (z_i - z)^2}$ and
$D_p = \sqrt{px_i^2 + f^2 + pz_i^2}$

D_i and D_p represent the magnitude of the object to camera center vector and the projection on image plane to camera center vector respectively.

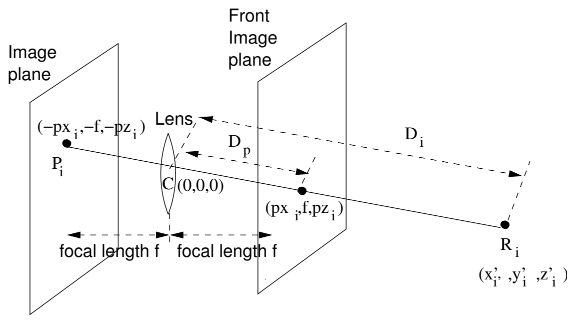


Fig. 4. Relationship between object location and its projection.

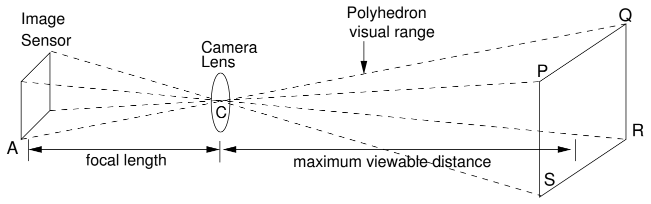


Fig. 5. Polyhedron representing the visual range of the camera.

Therefore, we can compute the location of an object in the camera coordinate system using Equation 7, given the camera location and focal length, and the object location and its projection. The actual location of each reference point and its location in the camera coordinates can then be used in Equation 6 to determine the rotation matrix R. Given R, we we can obtain pan α, tilt β, and roll γ using Equation 4 as follows:

$$\begin{aligned}\alpha &= \begin{cases} \arctan(\frac{r_{21}}{r_{22}}) - 180^\circ & \text{if } \frac{r_{21}}{\cos(\beta)} < 0 \text{ and } \frac{r_{22}}{\cos(\beta)} < 0 \\ \arctan(\frac{r_{21}}{r_{22}}) + 180^\circ & \text{if } \frac{r_{21}}{\cos(\beta)} >= 0 \text{ and } \frac{r_{22}}{\cos(\beta)} < 0 \\ \arctan(\frac{r_{21}}{r_{22}}) & \text{otherwise} \end{cases} \\ \beta &= \arcsin(r_{23}) \qquad (8) \\ \gamma &= \begin{cases} \arctan(\frac{r_{13}}{r_{33}}) - 180^\circ & \text{if } \frac{r_{13}}{\cos(\beta)} < 0 \text{ and } \frac{r_{33}}{\cos(\beta)} < 0 \\ \arctan(\frac{r_{13}}{r_{33}}) + 180^\circ & \text{if } \frac{r_{13}}{\cos(\beta)} >= 0 \text{ and } \frac{r_{33}}{\cos(\beta)} < 0 \\ \arctan(\frac{r_{13}}{r_{33}}) & \text{otherwise} \end{cases}\end{aligned}$$

Eliminating False Solutions: Recall from Section III-A that our six quadratic equations yields up to four possible solutions for the camera location. Only one of these solution is the true camera location. To eliminate false solutions, we compute the pan, tilt and roll for each computed location using three reference points. The fourth reference point is then used to eliminate false solutions as follows: for each computed location and orientation, we project the fourth reference point onto the camera's image plane. The projected coordinates are then matched to the actual pixel coordinates of the reference point in the image. The projected coordinates will match the pixel coordinates only for the true camera location. Thus, the three false solutions can be eliminated by picking the solution with the smallest re-projection error. The chosen solution is always guaranteed to be the correct camera location.

C. Determining Visual Range and Overlap

Once the location and orientation of each camera have been determined, the next task is to determine the visual range of each camera and the overlap of viewable regions between neighboring cameras. The overlap between cameras is an indication of the redundancy in sensor coverage in the environment. Overlapping cameras can also be used to localize and track moving objects in the environment.

The visual range of a camera can be approximated as a polyhedron as shown in Figure 5. The apex of the polyhedron is the location of the camera C (also the lens center) and height of the pyramid is the maximum viewable distance of the camera. An object in the volume of the polyhedron is in the visual range of the camera.

Although a camera can view infinitely distant objects, such objects will appear as point objects in any picture taken by the camera and are not useful for tasks such as object detection and recognition. Thus, it is necessary to artificially restrict the viewable range of the camera; the maximum viewable distance is determined in an application-specific manner and depends on the sizes of the objects being monitored (the larger the object, the greater is the maximum viewable distance of each camera). Assuming that this distance is determined offline, *Snapshot* can then precisely determine the polyhedron that encompasses the viewable range of the camera (assuming no obstacles such as walls are present to cut off this polyhedron).

Assume that the camera location (x, y, z) is given. We also assume that the size of the camera CMOS sensor is known (specifications for digital cameras typically specify the size of the internal CMOS sensor). Since the CMOS sensor is placed at a focal length distance from the lens, the coordinates of the four corners of the sensor can be determined relative to the camera location (x, y, z). As shown in Figure 5, the polyhedron is fully defined by specifying vectors $\overrightarrow{CP}$, $\overrightarrow{CQ}$, $\overrightarrow{CR}$ and $\overrightarrow{CS}$ which constitute its four edges. Further, $\overrightarrow{CP} = \frac{d}{f} \cdot \overrightarrow{AC}$, where AC is the line segment joining the edge of the CMOS sensor to the center of the lens, and d is the maximum viewable distance of the camera. Since the coordinates of points A and C are known, the vector $\overrightarrow{AC}$ is known, and $\overrightarrow{CP}$ can then be determined. The four edges of the polyhedron can be determined in this fashion.

To determine if two cameras overlap, we need to determine if their corresponding polyhedrons intersect (the intersection indicates the region in space viewable from both cameras). To determine if two polyhedrons intersect, we consider each surface of the first polyhedron and determine if one of the edges on the other polyhedron intersects this surface. For instance, does the line segment CP intersect any of the four surfaces of the other polyhedron? If any edge intersects a surface of the other polyhedron, then the two cameras have overlapping viewable regions. The intersection of a line segment with a plane can be easily represented in vector algebra using vector cross and dot products [11] and we omit specific details due to space constraints.

D. Iterative Refinement of Estimates

While *Snapshot* requires only four reference points to calibrate a camera sensor, the estimates of the camera location and orientation can be improved if additional reference points are available. Suppose that n reference points, $n \geq 4$, are available for a particular sensor node. Then $\binom{n}{4}$ unique subsets of four reference points can be constructed from these n points. For each subset of four points, we can compute the location and orientation of the camera using the techniques outlined in the previous sections. This yields $\binom{n}{4}$ different estimates of the camera location and orientation. These estimates can be refined to obtain the final solution using one of three methods:

Least Square Method: This technique picks one solution from the $\binom{n}{4}$ solutions that most accurately reflects the camera location and orientation. To do so, the technique uses each computed camera location and orientation to re-project all reference points on the camera image plane and chooses the solution that yields the minimum error between the projected coordinates and the actual coordinates in the image. The solution that yields the minimum error is one that minimizes the following expression

$$\sum_{i=1}^{n} || \frac{f}{y_i'} \times \begin{bmatrix} x_i' \\ y_i' \\ z_i' \end{bmatrix} - P_i ||^2. \qquad (9)$$

where $\begin{bmatrix} x_i' \\ y_i' \\ z_i' \end{bmatrix}$ is the location of reference point i in camera coordinates according to Equation 5, and $P_i = \begin{bmatrix} px_i \\ f \\ pz_i \end{bmatrix}$ is the real projection of reference point i.

Median Filter Method: This method simply takes the median of each estimated parameter, namely x, y, z, pan α, tilt β, and roll γ. These median values are then chosen as the final estimates of each parameter. Note that while the least squares method picks one of the $\binom{n}{4}$ initial solutions as the final solution, the median filter method can yield a final solution that is different from all $\binom{n}{4}$ initial solutions (since the median of each parameter is computed independently, the final solution need not correspond to any of the initial solutions). The median filter method is simple and cost-effective, and it performs well when n is large.

Maximum Likelihood Estimation: The MLE method [9] uses the initial estimates as its initial guess and searches through the state space to choose a solution that minimizes an error term. We choose the same error function as the least squares method: the search should yield a solution that yields the least error when projecting the reference points on the camera image plane.

Minimizing Equation 9 by searching through the parameter state space is a non-linear minimization problem, that can be solved numerically using the Levenberg-Marquardt algorithm. The algorithm requires an initial guess of $\mathbf{R}$ and (x, y, z): our estimates from *Snapshot* can be used as this initial guess. Note that, MLE is computationally more expensive that the median filter method or the least squares method. While its advantage diminishes when n is large, it can yield better accuracy when n is small.

Choosing between these methods involves a speed versus accuracy tradeoff. In general, the first two methods are more suitable if calibration speed is more important. The MLE method should be chosen when calibration accuracy is more important or if n is small.

IV. A Self-Calibration Protocol

In this section, we describe how the estimation techniques presented in the previous section can be instantiated into a simple wireless protocol for automatically calibrating each camera sensor. Our protocol assumes that each sensor node has a wireless interface that enables wireless communication to and from the camera. The calibration process involves the use of a wireless calibration device which is a piece of hardware that performs the following tasks. First, the device is used to define the reference points during calibration—the location of the device defines a reference point, whose coordinates are automatically determined by equipping the device with a positioning sensor (e.g., ultrasound-based Cricket). Second, the device also also serves as a point object for pictures taken by the camera sensors. To ensure that the device can be automatically detected in an image by vision processing algorithms, we equip the device with a bright LED sensor (which then serves as the point object in an image). Third, the devices serves as a "wireless remote" for taking pictures during the calibration phase. The devices is equipped with a switch that triggers a broadcast packet on the wireless channel. The packet contains the coordinates of the device at that instant and includes a image capture command that triggers a snapshot at all camera sensors in its wireless range.

Given such a device, the protocol works as follows. A human assists the calibration process by walking around with the calibration device. The protocol involves holding the device at randomly location points and initiating the trigger. The trigger broadcast a packet to all cameras in the range with a command to take a picture (if the sensor node is asleep, the trigger first wakes up a node using a wakeup-on-wireless algorithm). The broadcast packet also includes the coordinates of the current position of the device. Each camera then processes the picture to determine if the LED of the calibration device is visible to it. If so, the pixel coordinates of the device and the transmitted coordinates of the reference point are recorded. Otherwise the camera simply waits for the next trigger. When at least four reference points become available, the sensor node then processes this data to determine the location, orientation and range of the camera. These parameters are then broadcast so that neighboring cameras can subsequently use them for determining the amount of overlap between cameras. Once a camera calibrates itself, a visual cue is provided by turning on an LED on the sensor node so that the human assistant can move on to other sensors.

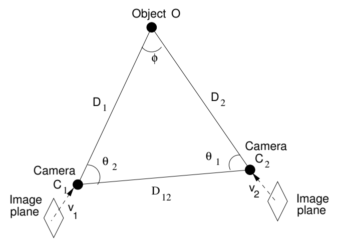


Fig. 6. Object localization using two cameras.

V. An Object Localization and Tracking Application

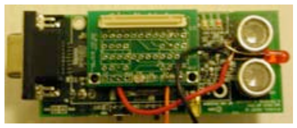
In general, the accuracy desired from the calibration phase depends on the application that will subsequently use this calibrated sensor network. To determine how calibration errors impact application accuracy, we consider a simple object localization and tracking example. This scenario assumes that the calibrated sensor network is used to detect external objects and track them as they move through the environment. Tracking is performed by continuously computing the coordinates of the moving object. A camera sensor network can employ triangulation techniques to determine the location of an object—if an object is simultaneously visible from at least two cameras, and if the locations and orientations of these cameras are known, then the location of the object can be calculated by taking pictures of the object and using its pixel coordinates to compute its actual location.

To see how this is done, consider Figure 6 that depicts an object O that simultaneously visible in cameras C_1 and C_2. Since both cameras are looking at the same object, the lines connecting the center of the cameras to the object, should intersect at the object O. Since the locations of each camera is known, a triangle C_1OC_2 can be constructed as shown in the figure. Let D_1 and D_2 denote the distance between the object and the two cameras, respectively, and let D_{12} denote the distance between the two cameras. Note that D_{12} can be computed as the Euclidean distance between the coordinates C_1 and C_2, while D_1 and D_2 are unknown quantities. Let θ_1, θ_2 and ϕ denote the internal angles of the triangle as shown in the figure. Then the Sine theorem for a triangle from elementary trigonometry states that

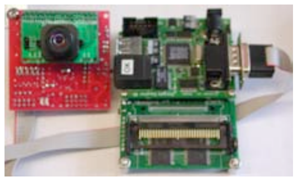
$$\frac{D_1}{\sin(\theta_1)} = \frac{D_2}{\sin(\theta_2)} = \frac{D_{12}}{\sin(\phi)} \tag{10}$$

The angles θ_1 and θ_2 can be computed by taking pictures of the object and using its pixel coordinates as follows. Suppose that the object projects an image at pixel coordinates $(-px_1, -pz_1)$ at camera C_1, Let f_1 denote the focal length of camera C_1. Then projection vector $\vec{v_1} = (px_1, f, pz_1)$ is the vector joining the pixel coordinates to the center of the lens and this vector lies along the direction of the object from the camera center. If $\vec{v}$ is the vector along the direction of line connected the two cameras, the the angle θ_1 can be calculated using the vector dot product:

$$\vec{v}.\vec{v_1} = |\vec{v}| \times |\vec{v_1}| \times \cos(\theta_1) \tag{11}$$



(a) Calibration Device



(b) CMUcam+Stargate

Fig. 7. *Snapshot* hardware components.

The angle θ_2 can be computed similarly and the angle ϕ is next determined as $(180 - \theta_1 - \theta_2)$.

Given θ_1, θ_2 and ϕ and the distance between two cameras D_{12}, the values of D_1 and D_2 can be computed using the Sine theorem as stated above.

Given the distance of the object from the cameras (as given by D_1 and D_2) and the direction along which the object lies (as defined by the projection vectors $\vec{v_1}$ and $\vec{v_2}$), the object location can be easily computed. Note that the orientation matrices of the cameras must also be accounted for when determining the world coordinates of the object using each camera. In practice, due to calibration errors, the object location as estimated by the two cameras are not identical. We calculate the mid–point of the two estimates as the location of the object.

Thus, two overlapping cameras can coordinate with one another to triangulate the location of an external object. We will use this object localization application in our experimental evaluation to quantify the impact of calibration errors on the application tracking error.

VI. Snapshot Implementation

This section describes the prototype implementation of *Snapshot*.

A. Hardware Components

The *Snapshot* wireless calibration device is a Mote sensor node equipped with a Cricket ultrasound receiver (see Figure 7(a)). We assume that the environment is equipped with Cricket reference beacons, which are used by a Cricket receiver to compute the location coordinates of the Mote during calibration [14]. We also enhance the Mote by equipping it with a LED that turns itself on during calibration. Empirical results have shown that Cricket-based positioning has an error of few centimeters and our experiments quantify the impact of this error on calibration accuracy.

We use two types of camera sensors in our experiments: the CMUcam vision sensor [5] and a Sony webcam. The CMUcam comprises of a OV6620 Omnivision CMOS camera and a SX52 micro–controller and has a resolution of 176x255. In contrast, the Sony webcam has a higher resolution of 640x480. We use the high resolution webcam to quantify the loss in accuracy when calibrating low-resolution cameras such as the CMUcam. Although beyond the scope of the current paper, our ongoing work focuses on calibrating a second low-resolution camera sensor, namely the Agilent Cyclops [12].

All camera sensors are connected to Intel Stargates [18] (see Figure 7(b)), which is a PDA-class sensor platform and s equipped with a 400MHz XScale processor. Each Stargate also has a Crossbow Mote [13] connected to it for wireless communication with our Mote-based calibration device.

Finally, we use a digital compass, Sparton 3003D [8], to quantify the orientation error during calibration. The compass has resolution of 0.1 degrees and accuracy of 0.3 degrees.

B. Software Architecture

Our Mote-based calibration device runs TinyOS [19] with the Cricket toolkit. The *Snapshot* software on the Mote is simple: each human-initiated trigger causes the Mote to determine its coordinates using Cricket and these coordinates are embedded in an “image-capture” trigger packet that is broadcast to all nodes. using the wireless radio.

Camera Calibration Tasks: Each Stargate runs the Linux operating system. Every time a trigger packet is received from the calibration device, the Stargate sends a set of commands over the serial cable to capture an image from the CMUcam. The image is processed using a vision-based recognition algorithm; our current prototype uses background subtraction and a connected components algorithm [15] to detect the presence of the calibration device LED. If the device is found, the pixel coordinates of the LED and the Cricket coordinates of the Mote are stored as a new reference point. Otherwise the image is ignored.

Once four reference points become available, the Stargate estimates the location, orientation and range of the camera A non–linear solver based on the interior–reflective Newton method [6], [7] is used to obtain the location estimate of the camera. Since the technique yields multiple solutions for the camera location, we compute the camera orientation for each location and then filter out false solutions by re-projecting the four reference point and comparing it to the actual coordinates. The solution that results in the least error between the computed projection and the real projection is chosen as the final estimate. The estimates can be further refined using as discussed in Section III-D if more than four reference points become available.

Object Localization and Tracking: Finally, we implement our object localization and tracking (described in Section V) application on the Stargates. If an object is simultaneously viewed by two cameras, the cameras exchange their parameters, location and orientation, and the objects projection coordinates on its image place. This information is used by each camera to localize the object and estimate its location. Continuous localization can be used at each node to track an object of interest.

VII. Experimental Evaluation

In this section we present a detailed experimental evaluation of the *Snapshot* protocol. Specifically, we evaluate the accuracy of *Snapshot* to estimate the camera parameters of location and orientation and also study some of the parameters that can affect its accuracy. We also evaluate the accuracy of the localization application which uses parameters determined by *Snapshot* and study the runtime scalability of *Snapshot*.

A. Experimental Setup

The setup to evaluate the accuracy and sensitivity to system parameters of *Snapshot* consisted of placing the two types of cameras, CMUcam and the Sony MotionEye webcam, at several locations. To simplify accurate location measurements we marked a grid to place the position sensor objects. Each camera took several pictures to estimate the parameters. The difference between the estimated parameter value and the actual value is reported as the measurement error. The Cricket sensors on the objects received beacons from a set of pre–calibrated Cricket sensor nodes placed on the ceiling of a room. The digital compass was attached to the two cameras in order to measure the exact orientation angles.

B. Camera Location Estimation Accuracy

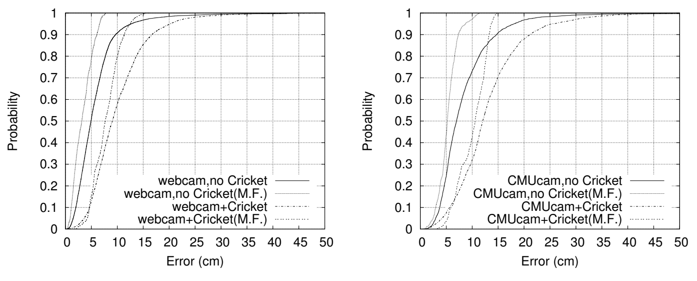
To evaluate *Snapshot*'s performance with camera location estimation, we place tens of reference points in the space, and take pictures of these reference points at different locations and orientations. We measure the location of these reference points by hand (referred as without Cricket) which can be considered as the object's real location and by Cricket [14] (referred as with Cricket) where we observed a 2–5cm error.

For each picture, we take all the combinations of any four reference points in view (not any 3 points in the same line), and estimate camera's location accordingly. We consider the distance between the estimated camera's location and the real camera's location as the location estimation error.

As shown in Figure 8(a), our results show: (i) the median errors using webcam without Cricket and with Cricket are $4.93cm$ and $9.05cm$, respectively; (ii) the lower quartile and higher quartile errors without Cricket are $3.14cm$ and $7.13cm$; (iii) the lower quartile and higher quartile errors with Cricket are $6.33cm$ and $12.79cm$; (iv) the median filter (referred as M.F. in figures) improves the median error to $3.16cm$ and $7.68cm$ for without Cricket and with Cricket, respectively.

Figure 8(b) shows: (i) the median errors using CMUcam without Cricket and with Cricket are $6.98cm$ and $12.01cm$, respectively; (ii) the lower quartile and higher quartile errors without Cricket are $5.03cm$ and $10.38cm$; (iii) the lower quartile and higher quartile errors with Cricket are $8.76cm$ and $15.97cm$; (iv) the median filter improves the median error to $5.21cm$ and $10.58cm$ for without Cricket and with Cricket, respectively.

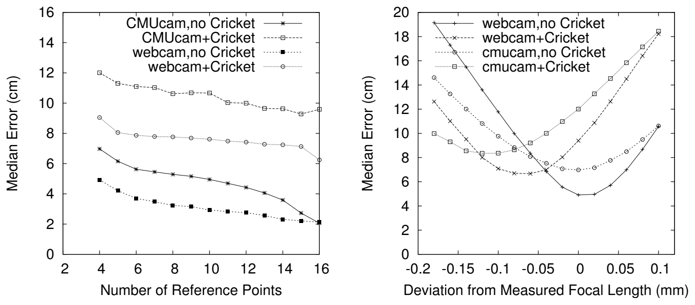
1) Effect of Iteration on Estimation Error: As our protocol proceeds, the number of available reference points increases. As a result, the number of combinations of any four reference points also increases, and we have more location estimations available for the median filter. Consequently, we can eliminate tails and outliers better. In this section, we study the effect of the iterations of our protocol's runs on camera location estimation error by plotting the median versus the number of available reference points.



(a) Webcam

(b) CMUcam

Fig. 8. Empirical CDF of error in estimation of camera location.



(a) Varying #reference points

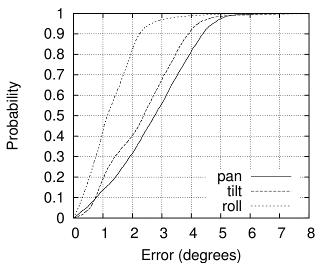
(b) Sensitivity to focal length

Fig. 9. Effect of number of reference points and focal length on location estimation error.

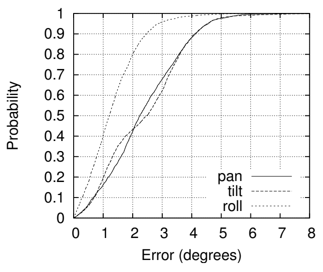
Figure 9(a) shows: (i) the median errors using webcam drop from $4.93cm$ to $2.13cm$ and from $9.05cm$ to $6.25cm$ as the number of reference points varies from 4 to 16 for without Cricket and with Cricket, respectively; (ii) the median errors using webcam drop from $6.98cm$ to $2.07cm$ and from $12.01cm$ to $9.59cm$ as the number of reference points varies from 4 to 16 for without Cricket and with Cricket, respectively. The difference in the location estimation errors (with and without Cricket) are due to the position error estimates of the Cricket and also due errors in values of camera intrinsic parameters.

C. Sensitivity to Lens Focal Length

Several intrinsic camera parameters can affect the estimation of the extrinsic parameters. To illustrate, we study the effect the focal length estimate on the estimated location of the camera. Figure 9(b) plots the median location estimation error of a camera with changing focal length. The X axis plots the deviations from the real focal length and show that even with small deviations the error in the estimate changes significantly. When the focal length is off by 0.1mm the error difference between a CMUcam with Cricket and without Cricket is 8cm, as compared to 5cm at the exact focal length. A similar trend,

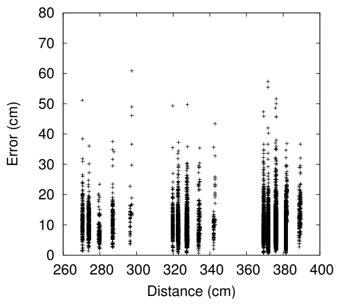


(a) CDF without Cricket

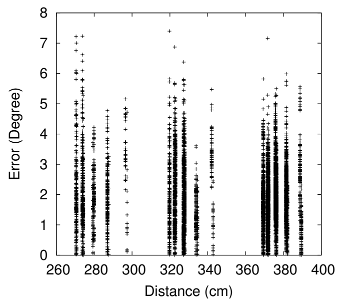


(b) CDF with Cricket

Fig. 10. Empirical CDF of error in estimating pan, tilt and roll orientations, with the CMUcam camera.



(a) Location Error



(b) Pan Error

Fig. 11. Effect of reference point distance from camera on estimated parameters for the webcam.

is seen with the webcam, where the error difference increase with deviation in focal length value from the real value. From the figure, both cameras without using Cricket, have minimum error at the real focal length. The minimum is shifted in the case of using Cricket, due to introduction of positioning errors.

The experiment demonstrates the sensitivity of calibrating the extrinsic parameters to a intrinsic parameter, the focal length. Thus, the intrinsic parameters should be accurately measured a priori to estimate the extrinsic parameters.

D. Camera Orientation Estimation Error

Next, we evaluate *Snapshot*'s accuracy with estimation of camera orientation parameters. We used the two cameras, the CMUcam and the Sony MotionEye webcam, to capture images of reference points at different locations and different orientations of the camera. We used estimated location of the camera based on exact locations on reference points and Cricket–reported locations of reference points to estimate the orientation parameters of the camera. The orientation of the camera was computed using the estimated camera location. We compared the estimated orientation angles with the measured angles to calculate error. Figure 10(a) shows the CDF of the error estimates of the pan, tilt and roll orientations respectively using the CMUcam camera. Figure 10(b) show the CDF of the error of the three orientations using Cricket for location estimation. The cumulative error plots follow the same trends for each of the orientation angles. The median roll orientation error using Cricket and without Cricket for camera location estimations is 1.2 degrees. In both cases, the 95th percentile error is less than 5 degrees for the pan and tilt orientation and less than 3 degrees for the roll orientation. The slight discrepancies in the error measurement of the two cases is due to the use the digital compass to measure the orientation of the camera.

Thus, we conclude the Cricket sensor's positioning errors do not add significant errors in estimation of camera orientation parameters. In our experiments, we find that a median location estimation error of 11cm does not affect the orientation estimation significantly.

E. Effect of distance of reference points

Since the measurement error of a point's projection can be a function of its distance from the camera, we study the effect of this distance on the estimated camera parameters. We plot the error in camera location and camera orientation estimation using Cricket with varying distance of the reference location. Since, the camera location estimation requires four reference points, we choose the maximum distance from the camera for the plot. As shown in Figure 11(a), the location error using the webcam is uniformly distributed in terms of distance from the camera. A similar trend is observed in the pan rotation estimates of the camera as seen in Figure 11(b). These results imply that the error in estimating the parameters is not sensitive to the distance of reference points measured in our experiments.

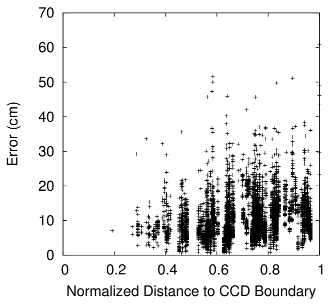
Other cameras and other rotations (tilt and roll) have similar results, and thus we omit those results in the interest of size limit, and refer readers to our technical report for more details.

F. Effect of Projection Location

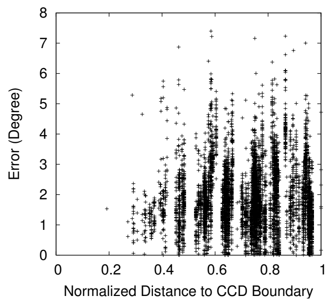
In this work, we do not estimate the internal parameters of the camera (e.g.: lens distortion and scale factor) which have an impact on the object's projection on the image plane, and thus can affect the accuracy of camera calibration. In general, the lens distortion has the largest impact on the object's projection on the image plane, and the closer the projection to the CCD boundary, the larger the impact.

In order to show this impact, we plot the error in camera location and camera orientation estimation using Cricket with varying distance of the projections to the CCD boundary in Figure 12. For the reason of easier understanding, we normalized the distance of the projections to the CCD boundary into the range $[0, 1]$ with 0 at the center of CCD, and 1 right on the boundary. Since, our camera calibration technique requires four reference points, we choose the reference points closest to the CCD boundary.

As shown in Figure 12, the location and pan orientation errors of the webcam are larger for reference points located



(a) Location Error



(b) Pan Error

Fig. 12. Effect of projection distance to CCD boundary on estimated parameters for the webcam.

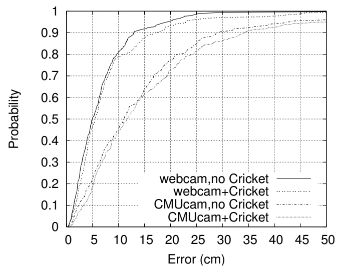


Fig. 13. Empirical CDF of error in estimation of object's location.

closer to the CCD boundary. A similar trend has been observed for the tilt and roll error, and for the CMUcam.

G. Object Localization

In this section, we study the performance of object localization using *Snapshot*. We use *Snapshot* to estimate camera locations and their orientations, and then in turn use the calibrated parameters to triangulate an object via the technique described in Section V. Similar to Section VII-B, we use the empirical CDF of object's location estimation error to measure the performance.

Our results (see Figure 13) show that: (i) object localization using webcams achieve median errors of $4.94cm$ and $5.45cm$ without Cricket and with Cricket, respectively; (ii) object localization using CMUcams achieve median errors of $11.10cm$ and $11.73cm$ without Cricket and with Cricket, respectively; (iii) object localization without using Cricket outperforms object localization using Cricket for all cameras; (iv) object localization using webcam outperforms object localization using CMUcam irrespective of using the Cricket.

H. Runtime Scalability

Using our prototype implementation of we measure the runtime of the *Snapshot* protocol. Table I reports the runtime of different tasks of the *Snapshot* calibration protocol executing on the Intel Stargate platform with the camera attached to a USB connector (the transfer of an image on the serial cable with the CMUcam requires additional time). As seen from the table, the location estimation task which uses a non–linear solver, has the highest execution time. The time to calibrate an individual camera is, 4 × (178 ms + 52 ms) – time to snap four images and recognize the location of object in each and 18365 ms for the location and orientation estimation algorithms, which is total time of 19.285 seconds. Thus, with a time of approximately 20 seconds to calibrate a single camera, *Snapshot* can easily calibrate tens of cameras on the scale of a few minutes.

Task	Duration(ms)
Snap Image	178 ± 2
Recognize Object Location	52 ± 0.1
Location Estimation	18365 ± 18

TABLE I
RUNTIME OF DIFFERENT CALIBRATION TASKS.

I. Summary of Key Results

The key results based on the implementation and experimental evaluation of *Snapshot* are as follows:

- *Feasibility:* By comparing the calibration accuracies of low and high-resolution cameras, we show that it is feasible to automatically calibrate low-resolution cameras such as CMUcams without significant loss in accuracy. The median error of location estimation of a CMUcam is 11 cm as compared to a webcam with a median error of 8 cm.
- *Accuracy: Snapshot* can localize a camera to within few centimeters of its actual location and determine its orientation with a median error of 1.3–2.5 degrees. More importantly, our experiments indicate that this level of accuracy is sufficient for tasks such as object tracking. We show that a system calibrated with *Snapshot* can localize an external object with a median error 11 cm, which is adequate for most tracking scenarios.
- *Efficiency:* The *Snapshot* protocol is computationally feasible on Stargate nodes and has running times in the order of a few seconds.
- *Scalability: Snapshot* can calibrate each camera sensor within 20 seconds on current hardware. Since a human needs to only specify a few reference points using the wireless calibration device—a process that takes a few seconds per sensor, *Snapshot* can easily scale to networks containing tens of camera sensors.

VIII. Related Work

Camera calibration using a set of known reference points is well studied in the computer vision community. Methods developed in [20], [21], [24] are examples of techniques that estimate both the intrinsic and extrinsic parameters of a camera using a set of known reference points. The goal of these efforts is to estimate a complete set of about twelve parameters of the camera. As a result, the methods require a larger number of reference points, are compute-intensive, and require multiple stages to determine all parameters. Snapshot is designed to estimate only the extrinsic parameters and requires only four known reference locations to estimate a camera's parameters. A recent effort [22] has proposed techniques to estimate only the extrinsic parameters and also requires four reference points. The technique requires three out of the four reference locations to be colinear. *Snapshot* is a more general technique and does impose such a requirement. Further, unlike [22], we demonstrate the feasibility of our approach through a detailed experimental evaluation.

Localization is well studied in the sensor networks community [10], [17], [23]. All these techniques assume a sensor node cable of position estimation. For example, a temperature sensor can use its RF wireless communication link to send and receive beacons for location estimation. Snapshot does not require any position estimation capability on the nodes and directly uses the imaging capability of the cameras for localization and calibration.

Several positioning and self-localization systems have been proposed in the literature. Active Badge [1] is a locationing system based in IR signals, where badges emit IR signals are used for location estimation. A similar successor system based on ultrasound signals is the Active Bat [2] system. Several other systems use RF signal strength measurements, like RADAR [3], for triangulation based localization. While most of these techniques are used indoors, GPS [4] is used for outdoor localization. While any of these methods can be used by the Snapshot calibration device instead of the Cricket, each has its own advantages and disadvantages. Based on the environment and desired error characteristics a suitable positioning system can be chosen.

IX. Conclusions

In this paper, we argued that prior vision-based techniques are not directly suitable for calibrating a camera sensor network and presented *Snapshot*, an automated calibration protocol that is explicitly designed and optimized for sensor networks. *Snapshot* uses the inherent imaging abilities of the cameras for calibration. Further, unlike techniques that require tens of reference points for calibration, *Snapshot* can determine the location and orientation of a camera sensor using only four reference points. Our techniques are based on principles from optics and geometry and are designed to work with low-fidelity, low-power camera sensors that are typical in sensor networks. As experimental evaluation of our prototype implementation showed that is feasible to employ *Snapshot* to calibrate low-resolution cameras and it is computationally feasible to run *Snapshot* on resource-constrained sensor nodes. Specifically, our experiments showed that *Snapshot* yields an error of 1-2.5 degrees when determining the camera orientation and 5-10cm when determining the camera location. We argued that this is a tolerable error in practice since a *Snapshot*-calibrated sensor network can track moving objects to within 11cm of their actual locations. Finally, our measurements showed that *Snapshot* can calibrate a camera sensor within 20 seconds, enabling it to calibrate a sensor network containing tens of cameras within minutes.

References

[1] Andy Harter and Andy Hopper. A Distributed Location System for the Active Office. *IEEE Network*, 8(1), January 1994.

[2] Andy Ward and Alan Jones and Andy Hopper. A New Location Technique for the Active Office. *IEEE Personal Communications*, 4(5):42–47, October 1997.

[3] P. Bahl and V. N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *INFOCOM*, pages 775–784, 2000.

[4] R. Bajaj, S. L. Ranaweera, and D. P. Agrawal. Gps: Location-tracking technology. *Computer*, 35(4):92–94, March 2002.

[5] The CMUcam2. http://www-2.cs.cmu.edu/ cmucam/cmucam2/index.html.

[6] T. F. Coleman and Y. Li. On the convergence of reflective newton methods for large-scale nonlinear minimization subject to bounds. *Mathematical Programming*, 67(2):189–224, 1994.

[7] T. F. Coleman and Y. Li. An interior, trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on Optimization*, 6(2):418–445, 1996.

[8] Sparton SP3003D Digital Compass. http://www.sparton.com/.

[9] J. L. Devore. *Probability and Statistics for Engineering and the Sciences*. Brooks/Cole, fifth edition, 1999.

[10] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher. Range-Free Localization Schemes in Large Scale Sensor Networks. In *Mobile Computing and Networking MOBICOM*, 2003.

[11] Joseph ORourke. *Computational Geometry in C*. Cambridge University Press, 2001.

[12] M. Rahimi and D. Estrin and R. Baer and H. Uyeno and J. Warrior. Cyclops, Image Sensing and Interpretation in Wireless Networks. Conference On Embedded Networked Sensor Systems, SenSys, 2004.

[13] Crossbow wireless sensor platform. http://www.xbow.com/Products/Wireless_Sensor_Networks.htm.

[14] N. B. Priyantha, A. Chakraborty, and H. Balakrishnan. The cricket location-support system. In *In Proceedings of the 6th annual ACM International Conference on Mobile Computing and Networking (MobiCom'00), Boston, MA*, pages 32–43, August 2000.

[15] A. Rosenfeld and J. L. Pfaltz. Sequential Operations in Digital Picture Processing. *J. ACM*, 13(4):471–494, 1966.

[16] A. Rowe, C. Rosenberg, and I. Nourbakhsh. A Low Cost Embedded Color Vision System. In *International Conference on Intelligent Robots and Systems*, 2002.

[17] A. Savvides, C.-C. Han, and M. B. Strivastava. Dynamic fine-grained localization in ad-hoc networks of sensors. In *Mobile Computing and Networking MOBICOM*, 2001.

[18] Stargate platform. http://www.xbow.com/Products/XScale.htm.

[19] TinyOS Website. http://www.tinyos.net/.

[20] R. Y. Tsai. An Efficient and Accurate Camera Calibration Technique for 3D Machine Vision. In *In Proceedings of 1986 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'86), Miami Beach, FL*, pages 364–374, June 1986.

[21] R. Y. Tsai. A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323–344, August 1987.

[22] F. Y. Wang. A Simple and Analytical Procedure for Calibrating Extrinsic Camera Parameters. *IEEE Transactions on Robotics and Automation*, 20(1):121–124, February 2004.

[23] K. Whitehouse and D. Culler. Calibration as Parameter Estimation in Sensor Networks. In *First ACM International Workshop on Sensor Networks and Applications (WSNA 2002)*, 2002.

[24] Z. Y. Zhang. A Flexible New Technique for Camera Calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, November 2000.