3D Target-based Distributed Smart Camera Network Localization

John Kassebaum, *Member, IEEE,* Nirupama Bulusu, *Member, IEEE,* and Wu-chi Feng, *Member, IEEE*

***Abstract*—For distributed smart camera networks to perform vision-based tasks such as subject recognition and tracking, every camera's position and orientation relative to a single 3D coordinate frame must be accurately determined. In this paper, we present a new camera network localization solution that requires successively showing a 3D feature point-rich target to all cameras in the network. Using the known geometry of a 3D target, cameras estimate and decompose projection matrices to compute their position and orientation relative to the coordinatization of the 3D target's feature points. As each 3D target position establishes a distinct coordinate frame, cameras that view more than one 3D target position compute translations and rotations relating different positions' coordinate frames, then share the transform data with neighbors to realign all cameras to a single coordinate frame established by one chosen target position. Compared to previous localization solutions that use opportunistically found visual data, our solution is more suitable to battery-powered, processing-constrained camera networks because it only requires pairwise view overlaps of sufficient size to see the 3D target and detect its feature points, and only requires communication to determine simultaneous target viewings and for the passing of the transform data. Finally, our solution gives camera positions in a 3D coordinate frame with meaningful units. We evaluate our algorithm in both real and simulated smart camera network deployments. In the real deployment, position error is less than 1" when the 3D target's feature points fill only 2.9% of the frame area.**

***Index Terms*—Camera calibration, smart cameras, camera network localization.**

I. INTRODUCTION

DISTRIBUTED smart camera networks consist of multiple cameras whose visual data is collectively processed to perform a task. The area covered by distributed smart camera networks can be small, viewing only a table for 3D capture and reconstruction of an object, covering a room, perhaps in a health care facility, or covering a very large area, such as an office building, airport, or outdoor environment for documentation, surveillance, or security.

Localization of a smart camera network means to determine all camera positions and orientations relative to a single 3D coordinate frame. Once localized, distributed smart camera networks can track a subject moving through the network by determining the subject's trajectory and triggering other cameras that are likely to soon view the subject. If the localization method provides camera positions in meaningful units such as feet or meters, the network can determine the actual size, depth, and position of detected subjects and objects, facilitating recognition and movement interpretation.

Due to obstructions in the deployment environment, such as walls or uneven terrain, hand-measuring camera positions and orientations is time consuming and prone to error. GPS is not accurate enough for vision-based tasks, nor does it provide camera orientation information. It is possible to use a network's available visual data to accurately localize the network, but these techniques impose a deployment constraint: the network's *vision graph*—in which vertices are cameras and edges indicate some view overlap—must be connected. A connected vision graph not only implies that each camera's view overlaps at least one other camera's, but also that some cameras in the network, if not most, have separate view overlaps with two or more cameras.

Vision-based localization has been well studied. The most recent solutions opportunistically search for robustly identifiable world features and correlate them between pairs of cameras with view overlaps [1], [2], [3]. Correlated features are used to estimate either the *essential* or *fundamental* matrix for two view overlapping cameras and which decomposed provides the camera pair's relative position and orientation, which is the data needed for network localization [4], [5]. The appeal of essential and fundamental matrix estimation localization methods—that they require image data only—can also be considered a shortcoming because they can only provide relative camera positions only up to an unknown scale factor, one which will vary for each separate pairwise localization. To adjust each pairwise localization to fit into a single network-wide coordinate frame, some solutions require triple-wise camera overlaps, implying the need for densely deployed networks. More recent solutions wave an LED-lit rod of known length through every camera's view, providing the means to establish a consistent scale [6], [7].

Our localization solution expands the advantage of the LED-lit rod by using a simple, 3D feature point-rich target of known geometry. A 3D target provides all feature points in one frame needed by one camera to determine its position and orientation relative to the target. Figure 1 shows a 3D target we designed and used to localize a small network. It has 288 3D feature points, far more than are needed for accurate localization, and includes colored areas to facilitate detection and correlation of the feature points projected to an image.

When a smart camera images the 3D target, it uses the well known DLT method [4], [8] to estimate a *projection* matrix from the feature points' known 3D and detected 2D coordinates. It then decomposes the estimated projection matrix to extract its position and orientation relative to the 3D target's coordinate frame, which is represented by the coordinatization of the 3D feature points. While the 3D target shown in Figure 1 is only one possible design, a 3D target is required for projection matrix estimation from a single frame. 2D planar

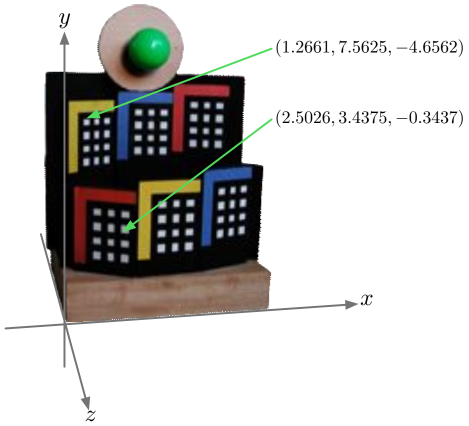


Fig. 1. The 3D target of known geometry used in our tests. The target contains 288 feature points, which are the corners of the white squares on the six separate grids. The green sphere on top of the target and the colored areas bounding the grid facilitate target detection, as described in Section IV.

targets, like the checkerboard patterns used to estimate a camera's intrinsic parameters [9], [10], do not satisfy the general position requirement, as explained in Section IV.

3D target-based localization still requires the network to have a connected vision graph. It also requires moving the 3D target through the network so it appears in each shared view overlap. But 3D target-based localization only requires pairwise view overlaps because when two or more cameras simultaneously localize to the 3D target, their position and orientation *relative to each other* is also automatically known, providing the data required for network localization. Of course, each time the 3D target is moved its coordinate frame changes relative to previous target positions—but our solution doesn't require measuring or tracking the target's movement. This is because any camera in the network that has separate view overlaps with two or more neighbors—and so localizes to more than one target position—can compute six transform parameters (3 rotation, 3 translation) between two of the target positions and pass the transform to the appropriate neighbors who then can realign their own positions and orientations to be relative to a target position they could not see. A simple algorithm running through the network can trigger the systematic realignment of all cameras to any one target position, thus localizing the network.

The contributions of this paper are:

- We present a new smart camera network localization solution with low processing and communication costs that requires only small pairwise camera view overlaps and provides camera positions in meaningful units. Cameras do not need to analyze frames to opportunistically extract local features because the 3D target provides all feature points necessary for projection matrix estimation. Also, because an algorithm to detect and localize to the 3D target runs independently on each smart camera, the only communication required to acquire two (or more) cameras relative position and orientation is to establish simultaneous viewings of the target. View overlaps that connect the network's vision graph need only be large enough for the two cameras to both see the target, while the target only need be large enough for its feature points to be clearly detected. Knowing camera positions in meaningful units allows the network to use computer vision algorithms to determine the actual size, depth, speed, and trajectory of detected objects.
- We present a design for a 3D target with sufficient feature points in general position to estimate a camera's projection matrix from a single frame. Colored areas of the 3D target facilitate detection of its feature points and correlation of the set of detected pixel coordinates with the feature points' known 3D coordinates.
- We validate our solution by localizing actual and simulated smart camera networks, and explore the impact of target size and position in frame, as well as the number of detectable feature points it provides.

The first item in particular makes our solution more appropriate for battery-powered smart camera networks with only a small CPU at each smart camera and for which wireless transmissions must be minimized to preserve network lifetime.

In Section II of this paper, we present an overview of extant smart camera network localization solutions and how they differ from our own. Section III presents our camera model and a brief overview of projection matrix estimation and decomposition. Sections IV and V discuss design considerations and requirements for 3D targets and how we use our to localize a network. Section VI presents localization results for both real and simulated camera networks.

II. Related Work

Automated methods to localize sensor networks require the means to gather range information between nodes, or between nodes and mutually observed targets. Non-camera-equipped networks consisting of resource-constrained scalar sensors can measure range information from ultrasound, radio, or acoustic signals [11]. Smart camera networks, to which our localization solution applies, can infer range information from visual data. Visual data needed for localization is gathered, in general, in one of two ways: 1) by tracking the motion of subjects viewed by multiple cameras, or 2) by finding the pixel coordinates of robustly identifiable features that either happen to appear or are deliberately placed in the views of cameras.

Solutions that track motion infer camera positions and orientations relative to the estimated trajectories of tracked subjects. These solutions pose the problem as a probabilistic inference task and maintain a joint distribution over camera and subject parameters. An issue with motion tracking is that subjects may change direction between camera views. Funiak et al. require view overlaps so that tracked subjects never fall out of view [12], while Meger et al. avoid requiring view overlaps by tracking a robot following a programmed path [13]. Another issue faced by motion tracking-based solutions is that posing the problem as a probabilistic inference task requires representing the joint distribution as a Gaussian, which so far is only done for camera positions and orientations in two dimensions. Rahimi et al. lift the solution to 3D by pre-computing homographies between each camera's image plane and a commonly viewed ground plane on which all tracked motion must occur [14].

Several researchers have proposed distributed smart camera network localization solutions that extract pixel coordinates of local features to estimate either essential or fundamental matrices between every pair of cameras with an overlapping view. These matrices express the epipolar geometry between two cameras in stereo configuration. The strength of epipolar geometry-based solutions is that they require only the network's available visual data; knowledge of the geometry of the 3D world is not required. Still, these solutions require detecting in images from both cameras the pixel coordinates of a minimum of eight commonly viewed world features. Moreover, these features must be correctly correlated between views. Detecting and correlating world features found in multiple camera views is commonly referred to as the *point correspondence problem*.

Lymberopoulos et al. solve the point correspondence problem by deploying nodes with self-identifying lights [3], but this solution may not be practical in bright or specular-filled environments. Mantzel et al. extract feature points by analyzing tracked motion, and correlate the features across views using time synchronization [1]. Mantzel et al. compensate for inaccurate correlations by determining a subset that produces the essential matrix estimate with the least error according to the epipolar constraint.

Devarajan et al. use SIFT [15] to find feature point correlations [2]. SIFT is an opportunistic feature point detection and correlation algorithm that searches a frame for candidate feature points, then gives ones it finds 128-dimension descriptors which are passed to other cameras in the network who probabilistically determine matches with their own descriptor sets. This type of opportunistic feature point detection and correlation has both high computation costs (for frame analysis) and communication costs (to discover correlations) that may be too expensive for resource-constrained smart camera network platforms. Also, it biases deployment towards larger view overlaps to increase the likelihood of finding sufficient correlations. Of course, it will fail in environments lacking trackable motion or extractable features. Our own solution avoids this possibility by providing all feature points required for network localization on a 3D target, the design considerations of which are discussed in Section IV.

As already mentioned, using essential or fundamental matrix estimation, the relative positions of all pairwise localized cameras will vary by an unknown scale factor. To adjust the different scale factors to fit a single coordinate frame, Manztel et al. and Devarajan et al. require that most cameras in the network view the same area as two others, which allows resolving camera triples to the same scale factor but implies the need for a dense deployment. Lymberopoulos et al.'s solution replaces the triple-wise overlapping view constraint with one requiring that some cameras see each other. Note that the units of the final coordinate frame which all cameras are resolved to will remain unknown, leaving the network unable to determine the actual size or depth of detected objects.

Two recent epipolar geometry-based solutions solve both the point correspondence and unknown scale factor problems by waving through the network a rod of known length with LED lights at each end [7], [6]. Both solutions use time-synchronized detections of the LED lights to provide correlated feature points. Then, after estimating fundamental matrices, both solutions use the known length of the bar to fix the units of relative camera positions. To reduce the error introduced by inaccuracies in the detected pixel coordinates of the bar's LED lights, Kurillo et al. use a bundle adjustment [16] performed at a central processor. Mederios et al. use a recursive technique that refines a camera's position and orientation estimate each time the camera or one of its known neighbors discover a new camera it can estimate a fundamental matrix with. Notably, both solutions require only pairwise view overlaps.

Like the rod-based solutions, our solution reduces the cost of feature point detection, eliminates the unknown scale factor problem, and reduces the number of required overlaps. However, our solution also eliminates the need to find world feature point correlations in shared camera views, and reduces the amount of required view overlap to only that needed for each camera to identify the target's feature points (which will vary based on target size, camera resolution, and lens focal length).

III. Camera Model and Projection Matrix Estimation

A. Camera model

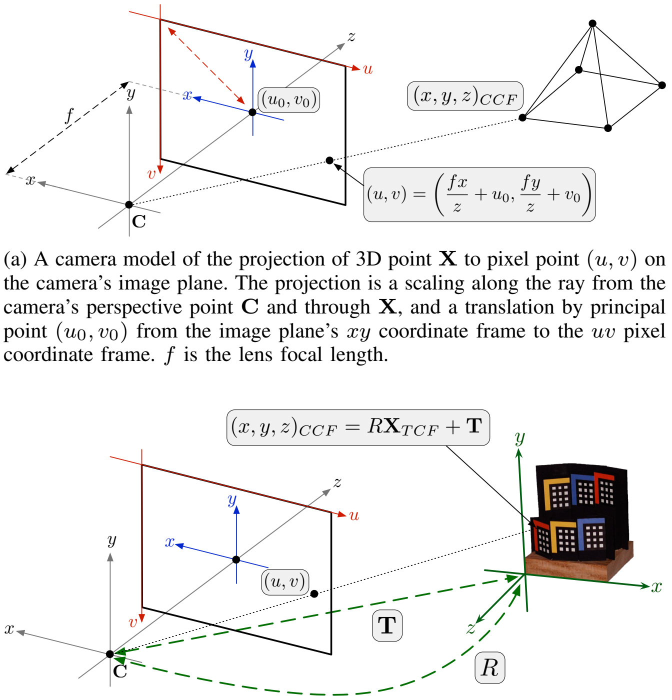
Cameras project the 3D Euclidean world to 2D images, dictated by the camera's perspective point, as shown in Figure 2a. We adopt the standard camera model in which the camera's perspective point $\mathbf{C}$ is the origin of a camera coordinate frame (CCF) and the image plane is parallel to the CCF's xy-plane. The mapping of a 3D point in the CCF to the image plane is performed by the camera's five *intrinsic* parameters contained in calibration matrix K:

$$K = \begin{bmatrix} \alpha_u & s & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

where α_u and α_v are the lens focal length in pixels and will differ if the horizontal spacing of pixels on the imager differs from their vertical spacing. s is a skew factor that accounts for image warping that occurs in images of other images [4] and can also account for distortion from the z (optical) axis not being strictly perpendicular to the image plane. *Principal point* (u_0, v_0) is the point at which the optical axis intersects the image plane expressed in pixel coordinates.

Camera calibration matrix K projects a 3D point in the CCF by scaling it to the image plane and translating it to pixel coordinates via the principal point. We treat the projected point as a homogenous point in order to get its 2D pixel coordinates in the uv plane:

$$\begin{bmatrix} \alpha_u & s & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \alpha_u x + sy + zu_0 \\ \alpha_v y + zv_0 \\ z \end{bmatrix} \equiv \left(\frac{\alpha_u x + sy}{z} + u_0, \frac{\alpha_v y}{z} + v_0 \right) \tag{2}$$



(a) A camera model of the projection of 3D point $\mathbf{X}$ to pixel point (u, v) on the camera's image plane. The projection is a scaling along the ray from the camera's perspective point $\mathbf{C}$ and through $\mathbf{X}$, and a translation by principal point (u_0, v_0) from the image plane's xy coordinate frame to the uv pixel coordinate frame. f is the lens focal length.

(b) Feature points in the world, in this case in the 3D target coordinate frame (TCF), prior to projection to the image plane must first be rotated and translated to the camera coordinate frame (CCF).

Fig. 2. The geometry of projection that maps 3D world points to 2D pixel coordinates.

In our localization solution, 3D points are expressed in the 3D target's coordinate frame (TCF), as shown in Figure 1. A 3D rotation and 3D translation realigns target feature points to the CCF, as shown in Figure 2b. We can rewrite this realignment using 3D homogenous coordinates:

$$(x, y, z)_{CCF} = R(X, Y, Z)_{TCF} + \mathbf{T} \equiv [R \mid R\mathbf{T}]\,(X, Y, Z, 1)^T \quad (3)$$

Including the transformation by K gives a camera's projection matrix:

$$(u, v) \equiv P\,(X, Y, Z, 1)^T \quad (4)$$

where

$$P = K\,[R \mid R\mathbf{T}] \quad (5)$$

B. Projection matrix estimation

We estimate a projection matrix using the well-known DLT method from the 3D coordinates of the target's feature points and their corresponding pixel coordinates in one image of the target taken by the camera. Each target and pixel point pair satisfies the equivalence:

$$\mathbf{u} \equiv P\mathbf{X} \quad (6)$$

where P is the camera's projection matrix and $\mathbf{X}$ is expressed in homogenous coordinates, as in Equation 4. We express the projection as an equivalence because P projects all 3D points on the ray through $\mathbf{X}$ and perspective point $\mathbf{C}$ to pixel point $\mathbf{u}$. Interpreting $\mathbf{u}$ and $P\mathbf{X}$ as vectors [4] yields the relation:

$$\mathbf{u} \times P\mathbf{X} = \mathbf{0} \quad (7)$$

Linearizing the cross product in the elements of P gives $A\mathbf{p} = 0$, where the elements of coefficient matrix A are expressions in the 3D target points' and correlated pixel points' coordinate values, and $\mathbf{p}$ is a 12×1 vector of the elements of P. Because P has 11 degrees of freedom, solving for $\mathbf{p}$ requires a minimum of six 3D and pixel point pairs in general position, as discussed in Section IV. The singular value decomposition of A gives $\mathbf{p}$.

C. Decomposing an estimated projection matrix

Decomposing an estimate of a projection matrix recovers the intrinsic parameter matrix K, the rotation matrix R and the translation vector $\mathbf{T}$. $\mathbf{T}$ and the three orientation angles in R comprise the camera's six *extrinsic* parameters, and are the camera's position and orientation in the TCF. If we let M be the left 3×3 submatrix of P, then $M = KR$ because:

$$P = K\,[R \mid R\mathbf{T}] \Rightarrow P = KR\,[I_{3\times3} \mid \mathbf{T}] \quad (8)$$

K is an upper triangular matrix and R is a rotation matrix, so we decompose M using the *RQ* decomposition, which multiplies by Givens rotation matrices *on the right* in order to successively zero the elements under M's diagonal [17].

To recover the camera's position in the 3D coordinate frame, we note that because $\mathbf{C}$ is the origin of the camera coordinate frame, $\mathbf{C} = -\mathbf{T}$, and that $\mathbf{C}$ is in the nullspace of P [4] because:

$$[I_{3\times3} \mid -\mathbf{C}]\left[\begin{array}{cc} \mathbf{C} & 1 \end{array}\right]^T = 0 \quad (9)$$

D. Extrinsic parameter refinement

Due to noise in the determination of the pixel coordinates of projected 3D target points, which is caused both by detection inaccuracies and lens distortion, we perform iterative refinement of the extracted parameters using the Levenberg-Marquadt optimization. The cost function we minimize is:

$$\sum_{i=1}^{n} (\mathbf{u}_i - P_{LD}\mathbf{X}_i)^2 \quad (10)$$

where n is the number of 3D and pixel point pairs used in the estimation, $\mathbf{u}_i$ and $\mathbf{X}_i$ are the coordinates of the pixel and 3D points, and P_{LD} is the projection of $\mathbf{X}$ using the refined camera parameters and two coefficients of radial lens distortion obtained using Zhang's well-known camera calibration technique [9]. Zhang's camera calibration technique also gives good estimates of the camera's five intrinsic parameters, so we use them as fixed values in the refinement. Thus, only the six extrinsic parameters giving the camera's position and orientation relative to the target are actually refined.

IV. The 3D Localization Target

Due to widely varying environmental and lighting conditions in which smart camera networks may be deployed, as well as variations in camera quality, lens focal length, and distance between cameras, it is unlikely that a single 3D target design will be detectable by all possible networks. Rather, the deployment conditions should dictate the type of target used.

Still, any target design must provide at least six feature points in general position to allow for projection matrix estimation. The general position requirement applies to both the 3D space of the target's feature points and the 2D space of the image plane. The requirement exists because, in the projection matrix estimation method described in Section III, coefficient matrix A must have rank eleven yet points not in general position produce linearly dependent rows. More precisely, any three or more collinear target feature points provide only five linearly independent rows for A, while any four or more coplanar target feature points (no three collinear) provide only eight linearly independent rows. This means that any target designed for projection matrix estimation must have at a minimum either two non-coplanar sets of three non-collinear points, or four coplanar points with no three collinear and two other points non-coplanar with the other four.

Checkerboard patterned targets and grids consisting of parallel circles or squares commonly used for intrinsic parameter estimation obviously violate the general position requirement. Calibration targets have these regular patterns—including our own test target design—to facilitate both feature point detection and correlation with the target's known geometry. Patterns also increase robustness to detection noise because collinear/coplanar feature points provide an oversampling of the constraints they represent. Using virtual cameras, we experimented with target configurations of two adjacent grids with varying angles between them, as well as a three-grid cube configuration. The results of single camera localizations relative to 3D targets with two adjacent grids set from 5° to 175° apart are shown in Figure 3. The results are acceptable but targets designed with only two grids risk the possibility that one grid may not be accurately detected, particularly for cameras set far apart. Similarly, three grids in a cube configuration requires an undesirably precise placement to ensure that all three sides are equally visible. Ultimately, we designed the 3D target shown in Figure 1 with six 48-point grids set vertically on a base at angles of $-30°$, $-30°$, $20°$, $-20°$, $5°$, and $-5°$ relative to the y-axis. The novelty of our design is that it combines ease of detection while supporting widely spaced cameras and has sufficient feature points to be robust to both single grid detection failures and detection noise. The results of single camera localization relative to our six grid target using either 2, 3, 4, 5, and all 6 grids is shown in Figure 4. In Section VI we present results using fewer feature points per grid.

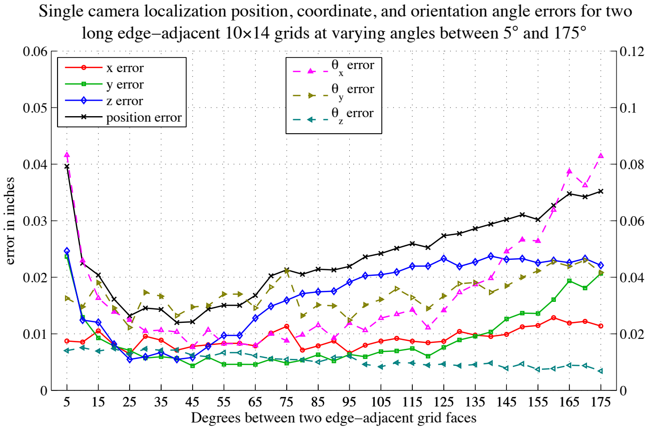


Fig. 3. Single camera localization error for a virtual 5mm camera with moderate barrel distortion localized to 3D targets comprised of two long-edge adjacent grids. Detection noise is simulated at 0.5 max pixel displacement.

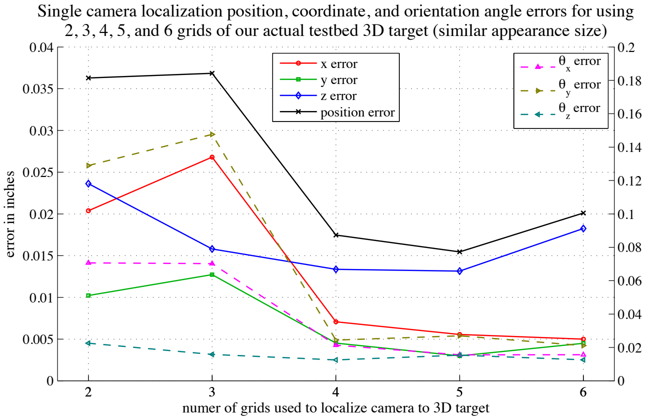


Fig. 4. Single camera localization error for a virtual 5mm camera with moderate barrel distortion localized to the target shown in Figure 1 using 2, 3, 4, 5, and all 6 grids. Detection noise is simulated at 0.5 max pixel displacement.

3D target design considerations also include efficient feature point detection and correlation. On our six grid 3D target, the green ball and colored areas bounding the left and top sides of each grid serve this purpose. The ball appears as a circle from any viewpoint and serves both as an initial indication that the target appears in frame and as a starting point for finding the rest of the target. Scanning along radii of the green circle, we find the top row's middle (blue) grid and thereby learn the orientation of the target (which is not required to appear upright). From the top middle grid we scan left and right (target orientation) to find the colored areas of the other two top row grids, then down to find the bottom row's colored areas. Next we find edge fits to the grid-side borders of the colored areas as they define two sides of a parallelogram that contain the grid's feature points. Finally, we can either scan over the parallelogram to find edge fits to the rows and columns of white squares, or use a corner detection algorithm such as that provided in OpenCV or the Camera Calibration Toolbox for Matlab. These steps to detecting the 3D target are shown in Figure 5.

V. Target-based Network Localization

When a camera localizes to the 3D target, we discover the camera's position and orientation relative to the target's coordinate frame, which is represented by the coordinatization of the target's feature points. If all cameras in a smart camera network are oriented such that they can see and localize to the 3D target simultaneously, then the network can be localized with just one target placement. More likely, though, the cameras do not all view the same area.

We can still localize the network, though, by successively placing the 3D target in the view overlaps that connect the network's vision graph. Of course when we move the 3D target from one view overlap to another, we lose its position

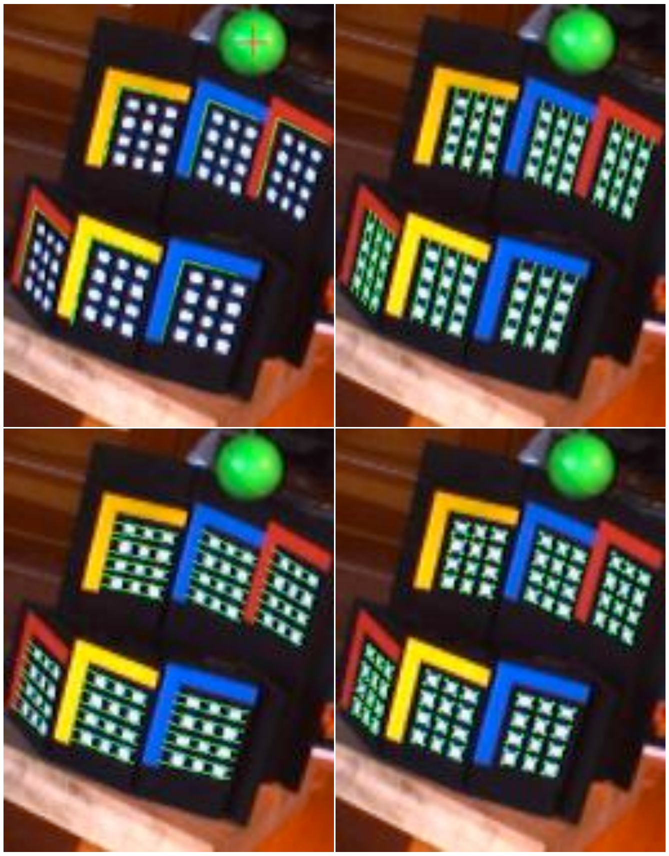
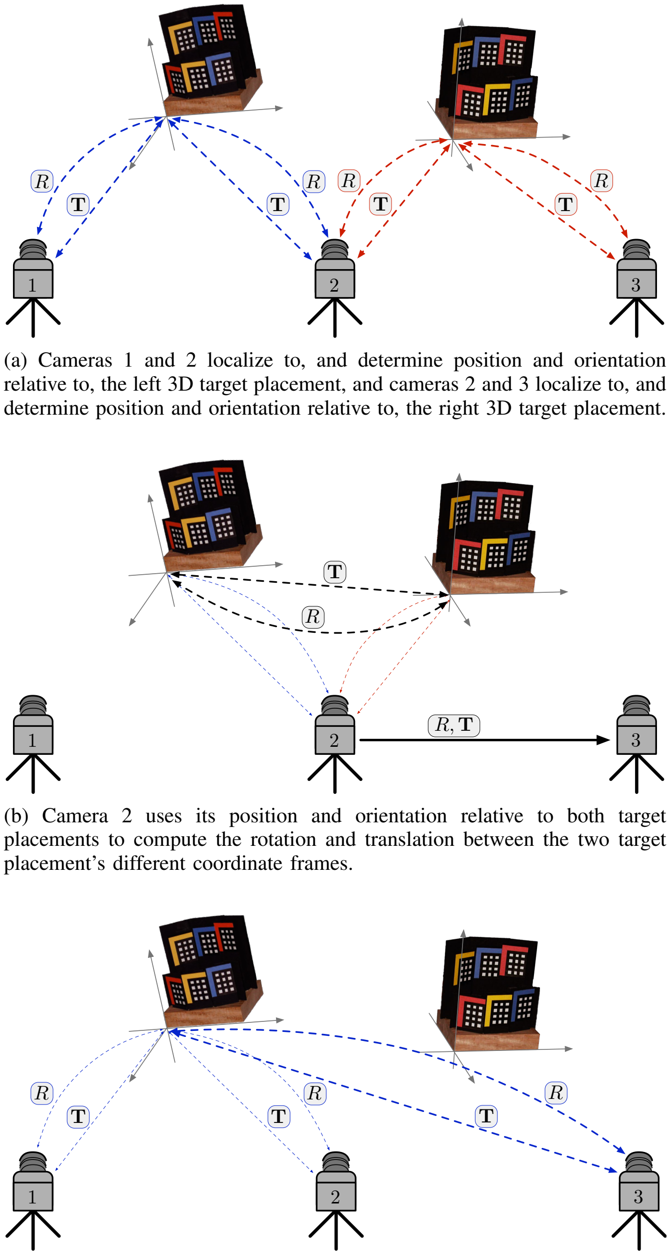


Fig. 5. Our detection algorithm first finds the center point of the green sphere, then uses it as a reference point to search in the direction of radii for the top outer edge of the top blue grid. The trajectory between these two features indicates the target's orientation in frame. (Detection does not require the target to appear upright.) Next, the algorithm finds line fits for the inner edges of the colored areas. The line fits define a parallelogram containing the corner points.

and orientation relative to the previous target placements. This means that camera positions and orientations computed by localizing to the different placements will be unrelated. Yet recall that in a network with a connected vision graph, many cameras have *separate* view overlaps with at least two other cameras, and so these cameras will localize to two different 3D target placements, one visible in each view overlap. A camera that localizes to two different 3D target placements can use its positions and orientations in both coordinate frames to compute the transform from one coordinate frame to the other, as shown in Figure 6a. Passing this six parameter transform (3 rotation, 3 translation) to a neighbor lets the neighbor realign its position and orientation to a target placement that it did not see, as shown in Figures 6b and c.



(a) Cameras 1 and 2 localize to, and determine position and orientation relative to, the left 3D target placement, and cameras 2 and 3 localize to, and determine position and orientation relative to, the right 3D target placement.

(b) Camera 2 uses its position and orientation relative to both target placements to compute the rotation and translation between the two target placement's different coordinate frames.

(c) Camera 3 applies the rotation and translation from camera 2 to is position and orientation relative to the right target placement and thereby determines its position and orientation relative to the left target placement, which it cannot see nor localize directly to.

Fig. 6. Realigning a camera to a target placement that it did not see. Cameras 1 can only see the target placement on the left, camera 2 sees both target placements, and camera 3 only sees the target placement on the right.

We can organize the realignment of cameras to a single network-wide coordinate frame in different ways. A linear approach chooses the first target placement as the global coordinate frame and then only moves the target into a view overlap of an already localized camera and one of its unlocalized neighbors. At this point all previously unlocalized cameras that can detect the target localize relative to it, and then are immediately passed the transform data to realign themselves to the global coordinate frame. While straightforward to implement and requiring the least amount of node-to-node communication, the linear approach is not ideal because single camera localization errors are inherently included in passed transform data. A better approach minimizes error propagation by choosing for the global coordinate frame the 3D target placement that results in the fewest number of camera realignments. We do this by finding a maximum spanning tree of the connected vision graph, finding the longest path in the spanning tree, and choosing for the global coordinate frame the middle 3D target placement on the longest path. Because the larger the 3D target appears in a camera's frame typically means the more accurate the localization, we use 3D target frame appearance size as the edge weight to determine the maximum spanning tree.

VI. Experimental Results

Earlier we stated one advantage 3D target-based localization has over essential or fundamental matrix estimation-based methods is smaller required view overlaps between cameras. This is because our solution requires view overlaps only large enough for both cameras to see and detect the 3D target simultaneously. Therefore we are interested in localization accuracy in relation to target size in frame, as this will dictate the amount of overlap needed.

We also stated that errors in estimates of camera positions and orientations from single camera localizations are propagated in realignment transforms passed to neighboring cameras. The size of these errors depends on the accuracy of the detection of the 3D target's feature points as well as on the accuracy of the estimated intrinsic parameters and lens distortion model used during the refinement step in Section III-D. Because estimated camera models are more accurate near the center of the frame than near the frame edges where the effects of lens distortion are greatest, we explore the propagation of localization error in a large virtual camera network with the 3D target placed both near the center of frame and in the corner, while also varying the amount of feature point detection noise.

Finally, to explore the effect on localization accuracy of the number of feature points provided on a target, we present results for single camera localizations using the 3D target design shown in Figure 1 but with only 32, 24, and 16 feature points per grid used to estimate the projection matrix. Note that fewer feature points on the target would also mean a physically smaller target that would further decrease the required overlap between cameras.

The goal of localization is accurate computation of camera positions and orientations. We report position error as the Euclidean distance between the actual and estimated camera positions, and directly measure the difference between actual and estimated camera orientation angles.

A. Implementation

1) Actual testbed: We test our 3D target-based localization algorithm on an actual network of five distributed smart cameras using the Panoptes [18] embedded smart camera platform. The nodes consist of COTS webcams with 640x480 pixel resolution connected to Crossbow Stargates with 400MHz Intel PXA255 processors running Linux. To facilitate measuring ground truth of camera positions and orientations, we disassembled the webcams and mounted their board cameras atop stands at various heights and angles, as shown in Figure 7, then positioned the camera stands on a flat, measured surface with our 3D target, also shown in Figure 7. Due to the small size of the network, we used the linear realignment approach described in Section V in order to force the most realignments.

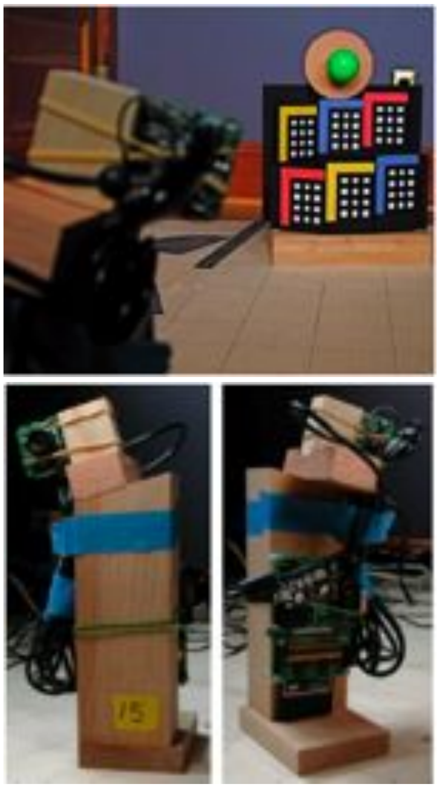


Fig. 7. To measure ground truth, we mounted board cameras taken out of webcams atop specially constructed stands, and set the stands and target on a flat, measured grid surface.

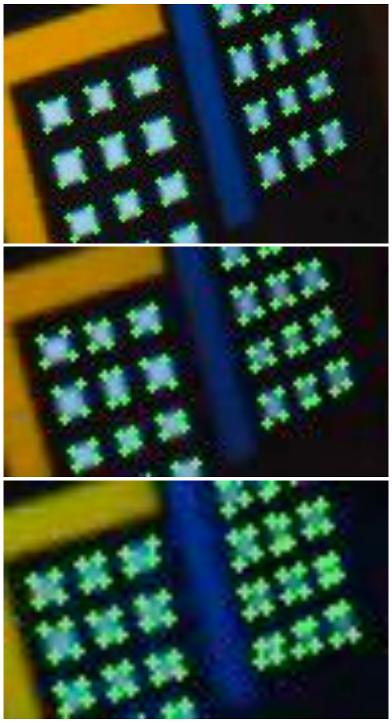
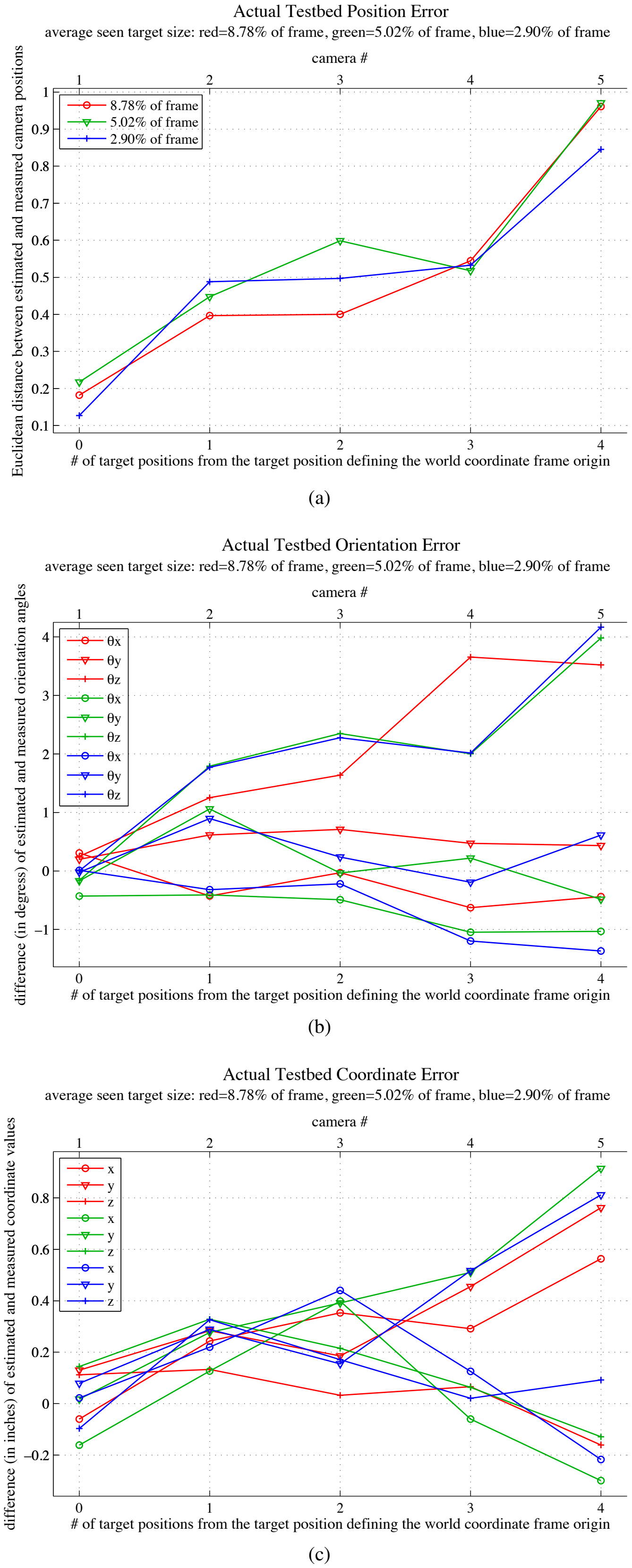


Fig. 8. The high accuracy of the feature point detections by a camera in the actual testbed experiments with the target successively smaller in frame. The target's feature points fill 9.21% of the frame area in the top image, 5.19% of the frame area in the middle image, and 2.94% of the frame area in the bottom image.

2) Simulated testbed: We simulate our 3D target-based localization algorithm in a virtual distributed smart camera network, with user specified camera and 3D target positions and orientations. We model cameras with five intrinsic parameters and 14 lens distortion parameters: four coefficients each for radial, decentering and thin prism distortions and separate axes of maximum tangential distortion for decentering and thin prism distortions [19], [20]. While there is no extant method to accurately estimate this many lens distortion parameters, we base the plausibility of the values we chose on results from the separate estimations of two radial distortion coefficients, and on the dominating effects of radial distortion over tangential distortion—on the order of 7 : 1—reported by Weng et al. [20]. The simulator recreates our localization algorithm as described in Sections III-B, III-C, and III-D, which includes the separate estimation of intrinsic and radial distortion parameters from nine virtual images of a 2D checkerboard calibration target. The simulator creates the set of pixel coordinates of detected feature points by first projecting a virtual target using a camera's actual extrinsic, intrinsic and lens distortion parameters, then introducing noise by randomly shifting each projected feature point up to a user-specified maximum in a random direction. To localize a network, the simulator either determines an optimal localization tree that minimizes realignment transforms as discussed in Section V, or allows for a user-specified tree. The simulator measures localization accuracy by comparing the user-specified actual extrinsic parameters of the virtual cameras with their estimated values.

B. Actual camera network results

To explore the effects of target appearance size, we performed the actual testbed localization three times, each time with the 3D target set further away from the cameras. In the third localization, the target's feature points occupy less than 3% of any camera's frame area. The localization results are shown in Figure 9. Figure 8 shows a sample of the feature point detection accuracy for each target distance. The graphs indicate that the frame appearance size of the 3D target has no bearing on the localization accuracy. Position, orientation angle, and coordinate errors are all almost unchanged across the three localizations. We presume, without clear validation, that the growth of z-axis orientation error compared to the nearly flat x- and y-axis orientation errors, and similar growth of the y coordinate error compared to the x and z coordinate errors, may be attributable to a combination of flaws in our ground truth measurements and an incomplete lens distortion model. In the configuration of our testbed with the target's z-axis pointing towards the cameras, flaws in the flatness of the table, the verticality of the camera stands, and the measurement of the tilt of the cameras will manifest most as y-coordinate and z-axis errors. Also, these singular parameter errors do not appear in virtual cameras with well-estimated lens distortion curves. For instance, in a simulated reconstruction of the actual testbed configuration, discussed in the next subsection, the localization error is significantly less and the coordinate and orientation errors do not vary as significantly.



(a)

(b)

(c)

Fig. 9. Error in the estimated camera positions, orientation angles, and coordinate values in an actual testbed of five cameras localized three times, each with a different general target size in frame.

TABLE I
ACTUAL TESTBED VS. SIMULATED TESTBED INTRINSIC AND LENS DISTORTION PARAMETER ESTIMATIONS FOR EACH CAMERA

C	act. est. α_x	sim. est. α_x	act. est. α_y	sim. est. α_y	act. est. s	sim. est. s	act. est. u_0	sim. est. u_0	act. est. v_0	sim. est. v_0	act. est. k_1	sim. est. k_1	act. est. k_2	sim. est. k_2
1	809.43	808.95	810.04	809.60	-2.854	-2.870	300.61	300.82	223.03	223.18	.0950	.0829	-.2097	-.1959
2	809.63	809.57	809.58	809.60	-1.627	-1.672	339.83	341.63	243.59	241.78	-.0404	-.0415	.0536	.0555
3	790.60	790.56	792.86	792.79	-2.321	-2.297	334.95	334.32	244.37	244.87	.1066	.1044	-.0242	-.0334
4	789.47	789.34	790.58	790.41	-2.341	-2.356	320.19	322.07	255.35	255.47	.0985	.1015	-.0303	-.0494
5	815.12	814.94	815.94	815.72	-1.156	-1.178	341.60	343.30	264.51	264.58	-.0258	-.0221	-.0088	0.0098

C. Virtual camera network results

Using the simulator, we recreated the actual five camera network configuration. To closely approximate the lens distortion of each actual camera, we used a set of 14 lens distortion parameters that resolve in the separate intrinsic and lens distortion parameter estimation to nearly the same two coefficients of radial distortion. Table I compares the actual and simulated parameters of the five cameras.

Because feature point detection should not be precisely recreated in the simulation, detection accuracy is a variable. The graphs in Figure 10a show the position, coordinate, and orientation errors from the simulated network localization at four different maximum random noise levels. As can be expected, as noise decreases, accuracy increases. The sudden growth of position error when the target is smallest and noise is the highest occurs because randomly introduced pixel noise is essentially a scattering effect on the closely spaced pixel coordinates of the detected feature points. The graphs in Figure 10 show the individual coordinate and orientation angle errors in the simulated reproduction at a maximum random noise displacement of two pixels.

In Figure 11 we show the average position errors of 25 localizations of a large simulated network that required 20 daisy-chained camera realignments prior to bringing the final camera into the global coordinate frame. All simulated cameras have wide aspect lenses and a maximum possible lens distortion displacement of 25 pixels, though the cameras averaged eight. We performed the localization at two different noise levels and with two different target positions: one in which the target appears either in the bottom left or right corners of the frame, and second in which the target appears either halfway between the center and left edge of frame or halfway between the center and right edge of frame. Despite the fact that the target has a larger appearance size when seen in the corner positions, the localization is more accurate at the smaller appearance size closer to the frame center. This is due to estimated camera model being less accurate for points near the edge of frame.

Lastly, in Figure 12 we show results of single camera localization to our 3D target when using only either 16, 24, or 32 of the available 48 feature points per grid at two different maximum noise levels and when the 3D target's feature points occupy 2.5% of the frame area. As expected, at the higher noise level there is greater localization error. Yet the use of fewer points has almost no impact on accuracy. These results indicate that our 3D target could be reduced to having only 16 feature points per grid.

VII. CONCLUSION AND FUTURE WORK

We have presented a new 3D target-based localization solution for smart camera networks that is a practical alternative to existing epipolar geometry-based localization solutions. Generally, epipolar geometry-based solutions are most useful for deployments where large view overlaps are desired, such as small stereoscopic vision systems. Our solution, though, requires only small, pairwise view overlaps, making it more suitable for larger networks deployed for human or environmental monitoring. Also, because the 3D target provides all feature points needed for localization in one frame, computation and communication costs are acceptable for battery-powered, processing-constrained networks. Our solution also gives camera positions in meaningful units, facilitating detected object measurements.

The accuracy of the camera positions and orientations provided by our solution depends solely on the accuracy of the estimation of the camera's projection matrix—which in turn depends on the accuracy of the detection of the pixel coordinates of the target's feature points. It also depends on the accuracy of prior estimates of the camera's intrinsic and lens distortion parameters. The many experiments we have performed strongly suggest the need for improvement in the latter, and so this will be a focus of our future work.

There are also other physical aspects of our solution to be explored, such as spatial resolution of the cameras and impact of other target designs, such as spherical. We also plan to explore possible error refinement methods, such as performing a sparse bundle adjustment on pairwise localizations, and a bundle adjustment over the entire network configuration estimation.

ACKNOWLEDGMENT

This material is based upon work supported by the National Science Foundation under Grant No. 0722063.

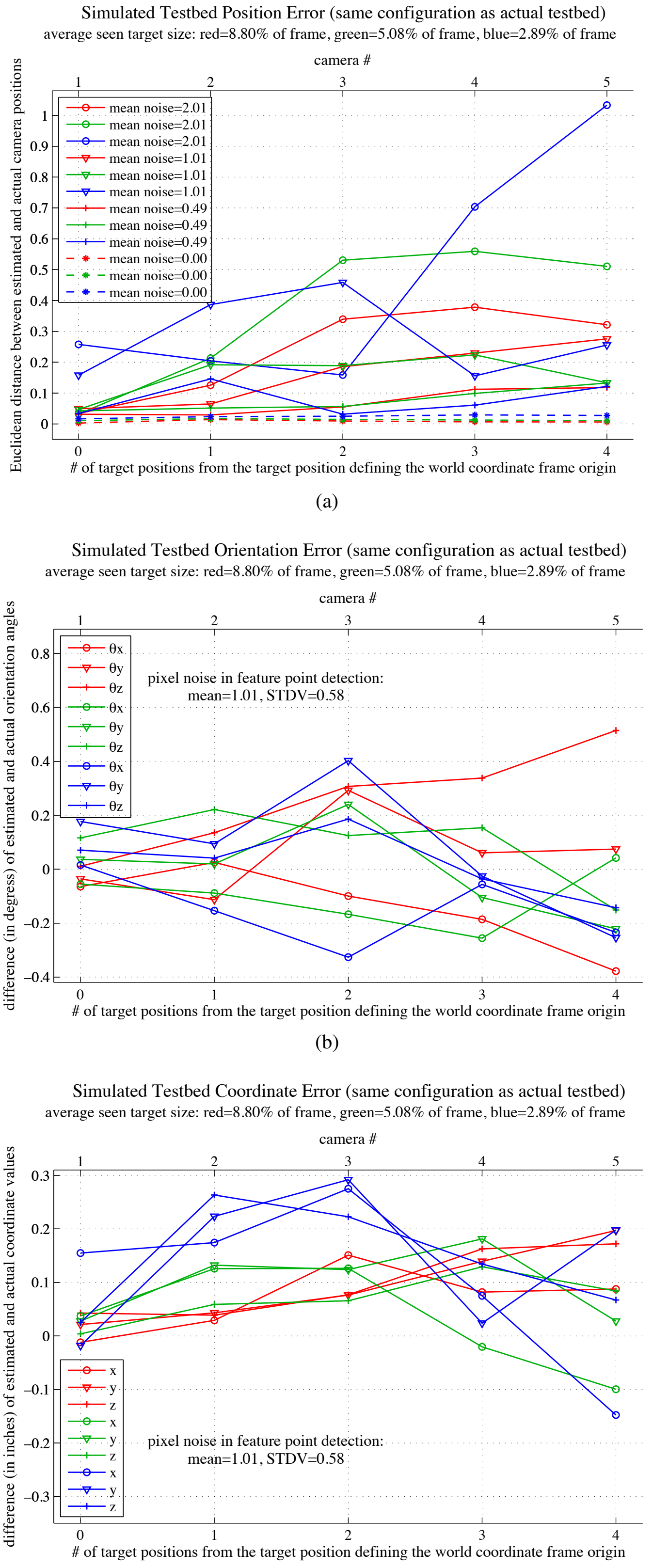
REFERENCES

[1] W. Mantzel, H. Choi, and R. Baraniuk, "Distributed camera network localization," in *Signals, Systems and Computers, 2004. Conference Record of the Thirty-Eighth Asilomar Conference on*, vol. 2, November 2004, pp. 1381–1386.

[2] D. Devarajan, R. J. Radke, and H. Chung, "Distributed metric calibration of ad hoc camera networks," *ACM Trans. Sen. Netw.*, vol. 2, no. 3, pp. 380–403, 2006.

[3] D. Lymberopoulos, A. Barton-Sweeny, and A. Savvides, "Sensor localization and camera calibration using low power cameras," ENALAB, Tech. Rep. 090105, September 2005.

[4] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 1st ed. Cambridge University Press, 2000.



(a)

(b)

(c)

Fig. 10. Error in the estimated position, coordinate, and orientation angles in a simulated reproduction of the actual testbed of five cameras and localized three times, each with a different general target size in frame. The graph of position errors includes results from different detection noise levels. The graphs of coordinate and orientation angle errors show results only for a maximum of two pixels noise displacement per detected feature point.

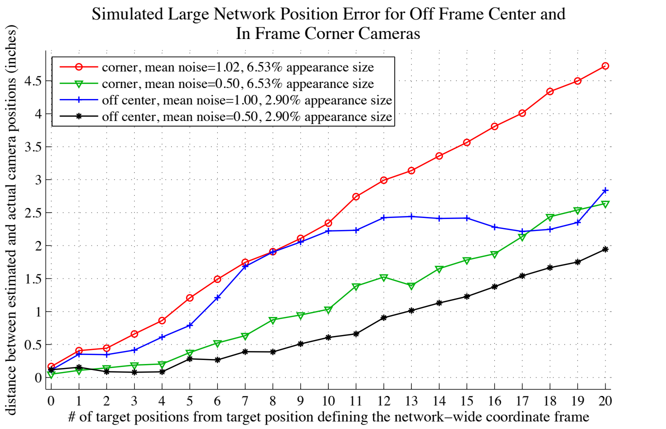
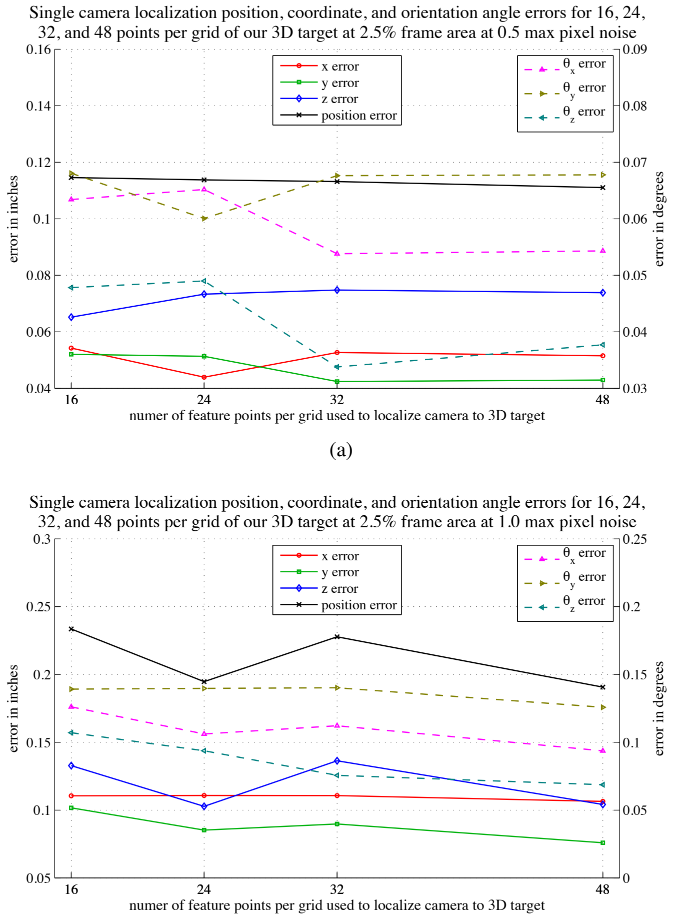


Fig. 11. Position error in the localization of a large simulated network at two different maximum noise levels and with the target appearing either only in the bottom left and right corners of the frame, where lens distortion is the greatest, or just to the left or right of the center of frame.



(a)

(b)

Fig. 12. Error in the estimated position, coordinate, and orientation angles for a simulated testbed camera when using fewer target feature points for localization. The top and bottom graphs show the results at a maximum random detection noise of 0.5 pixels and 1.0 pixels, respectively.

[5] O. Faugeras, Q.-T. Luong, and T. Papadopoulou, *The Geometry of Multiple Images: The Laws That Govern The Formation of Images of A Scene and Some of Their Applications*. Cambridge, MA, USA: MIT Press, 2001.

[6] H. Medeiros, H. Iwaki, and J. Park, “Online distributed calibration of a large network of wireless cameras using dynamic clustering,” in *Second ACM/IEEE International Conference on Distributed Smart Cameras, 2008*, September 2008, pp. 1–10.

[7] G. Kurillo, Z. Li, and R. Bajcsy, “Wide-area external multi-camera calibration using vision graphs and virtual calibration object,” in *Second ACM/IEEE International Conference on Distributed Smart Cameras, 2008*, 2008, pp. 1–9.

[8] Y. I. Abdel-Aziz and H. Karara, “Direct linear transformation from comparator coordinates into object space coordinates in close-range photogrammetry,” in *Proceedings ASP/UI symposium on close-range photogrammetry*, 1971, pp. 1–18.

[9] Z. Zhang, “A flexible new technique for camera calibration,” Microsoft Research, One Microsoft Way, Redmond, WA 98052, Tech. Rep. MSR-TR-98-71, 1999.

[10] J. Heikkila and O. Silven, “A four-step camera calibration procedure with implicit image correction,” in *CVPR ’97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR ’97)*. Washington, DC, USA: IEEE Computer Society, 1997, p. 1106.

[11] C. Taylor, A. Rahimi, J. Bachrach, H. Shrobe, and A. Grue, “Simultaneous localization, calibration, and tracking in an ad hoc sensor network,” in *IPSN ’06: Proceedings of the 5th international conference on Information processing in sensor networks*. New York, NY, USA: ACM, 2006, pp. 27–33.

[12] S. Funiak, M. Paskin, C. Guestrin, and R. Sukthankar, “Distributed localization of networked cameras,” in *IPSN ’06: Proceedings of the 5th international conference on Information processing in sensor networks*. New York, NY, USA: ACM, 2006, pp. 34–42.

[13] D. Meger, I. Rekleitis, and G. Dudek, “Simultaneous planning, localization, and mapping in a camera sensor network,” *Robotics and Autonomous Systems (RAS) Journal special*, vol. 54, p. 932, 2006.

[14] A. Rahimi, B. Dunagan, and T. Darrell, “Simultaneous calibration and tracking with a network of non-overlapping sensors,” in *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 1. IEEE Computer Society, 2004, pp. I–187–I–194.

[15] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, pp. 91–110, 2004.

[16] B. Triggs, P. McLauchlan, R. Hartley, and A. Fitzgibbon, “Bundle adjustment — a modern synthesis,” in *ICCV ’99: Proceedings of the International Workshop on Vision Algorithms*. London, UK: Springer-Verlag, 2000, pp. 298–372.

[17] D. Bindel, J. Demmel, W. Kahan, and O. Marques, “On computing givens rotations on computing givens rotations reliably and efficiently,” *ACM Transactions on Mathematical Software*, vol. 28, no. 2, pp. 206–238, June 2002.

[18] W.-C. Feng, E. Kaiser, W. C. Feng, and M. L. Baillif, “Panoptes: scalable low-power video sensor networking technologies,” *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 1, no. 2, pp. 151–167, 2005.

[19] C. Slama, Ed., *Manual of Photogrammetry*, 4th ed. American Society of Photogrammetry, 1980.

[20] J. Weng, P. Cohen, and M. Herniou, “Camera calibration with distortion models and accuracy evaluation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 10, pp. 965–980, 1992.

John Kassebaum John Kassebaum is a research assistant and working towards his Ph.D. at Portland State University.

Nirupama Bulusu Dr. Nirupama Bulusu is an Assistant Professor of Computer Science at Portland State University. At PSU, she leads an active research group in sensor networks, with a focus on audiovisual sensing applications. Besides authoring more than 50 publications on sensor networks, she was also the principal editor for the book, ”Wireless Sensor Networks: A Systems Perspective”, published by Artech House in August 2005.

She received her Ph.D from UCLA in 2002, her M.S from USC in 2000, and her B.Tech from IIT Madras in 1997, all in Computer Science. She is a recipient of the NSF CAREER award, the Institute Merit Prize from IIT Madras, the Provost’s PSU Foundation Faculty Development Award from Portland State University and a co-recipient of the UNSW GoldStar Award from the University of New South Wales.

She serves on the editorial board of Elsevier Ad Hoc Networks journal, and previously served on the program committees of several premier sensor network conferences including ACM SenSys, EWSN, IPSN, DCOSS and EmNetS. She is a member of ACM, IEEE, USENIX and the Sigma Xi honor society.

Wuchi Feng Wu-chi Feng received his B.S. degree in Computer Engineering from the Pennsylvania State University in 1990. He received his M.S. and Ph.D. degrees in Computer Science and Engineering from from the University of Michigan in 1992 and 1996, respectively. His research interests include multimodal sensor networking technologies, multimedia streaming, multimedia systems, and network games. Wu-chi Feng is currently Professor and Chairman of the Computer Science Department at Portland State University. He received an NSF CAREER award in 1998 for his research in network protocols for video-on-demand services. During the last several years, Dr. Feng has served on numerous program committees including ACM Multimedia, ACM/SPIE Multimedia Computing and Networking, and the International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV). He is also currently serving as an editor for the ACM Multimedia Systems Journal and chair of the steering committee for the ACM Multimedia Systems Conference. He has published over 90 journal and conference publications in area of multimedia streaming and networking.