PERGAMON

Pattern Recognition 34 (2001) 1105–1117

PATTERN RECOGNITION
THE JOURNAL OF THE PATTERN RECOGNITION SOCIETY

www.elsevier.com/locate/patcog

Calibrating a camera network using a domino grid

Bent David Olsen[a], Adam Hoover[b,*]

[a]*Laboratory of Image Analysis, Department of Medical Informatics and Image Analysis, Institute for Electronic Systems, Aalborg University, Denmark*

[b]*Visual Computing Lab, Electrical and Computer Engineering Department, University of California, San Diego, La Jolla, CA 92093-0407, USA*

Received 29 September 1998; received in revised form 29 July 1999; accepted 18 January 2000

Abstract

This work considers the problem of calibrating a distributed multi-camera system. As opposed to traditional multi-camera systems, such as stereo heads, in a distributed network the fields-of-view do not all overlap. Our novel method uses a grid of domino calibration targets. We also describe a novel grid-finding algorithm, to expedite the location of image-to-world correspondences. Experiments conducted in a hallway and two connecting rooms, using 12 cameras, demonstrate accuracies of 4.3 mm in world coordinates and 0.28 pixels in image coordinates. © 2001 Published by Elsevier Science Ltd on behalf of Pattern Recognition Society.

Keywords: Camera calibration; Distributed sensing; Multi-camera system; Grid detection; Correspondence problem; Camera network

1. Introduction

This paper describes a method to calibrate a network of cameras to a common world coordinate system. The working environment is assumed to resemble common indoor floorplans, with rooms and corridors connected via doorways. The camera network is assumed to resemble a security video network, where each camera views a moderate amount of floorspace from an elevated position.

Traditional installations for multi-camera systems include stereo heads, object modeling scanners, and factory workcells. In these configurations the fields-of-view of the cameras all overlap. In this case a single target can be used to calibrate all the cameras to a common world coordinate system [1]. Fig. 1 illustrates a common configuration.

In this work we consider a distributed multi-camera system. Fig. 2 illustrates a possible configuration. In this configuration the fields-of-view of the cameras do not overlap, so that multiple targets must be used for calibration. In this case methods are needed to establish the positions of the multiple calibration targets in a common world coordinate system.

Camera calibration also requires the establishment of correspondences between image points and world coordinates. A traditional multi-camera system might include two to five cameras. In this case, manual methods are tolerable. One approach is to carefully measure image coordinates using a graphical user interface. However, as the number of cameras increases, such methods become increasingly tedious and error prone.

In this work, multiple, identical tiles are used for calibration targets. The tiles are posterboard-sized. Each tile exhibits a 2×2 dot pattern. The dots are positioned so that when two tiles are placed side-by-side, or end-to-end, the spacing between consecutive dots remains constant. We refer to these calibration targets as dominoes.

Dominoes are placed on the floor, throughout the floorspace of the combined field-of-view of the cameras. All the dominoes must be adjoining side-by-side or end-to-end. In this manner, the dots deploy a common coordinate system. The dominoes do not require any

* Corresponding author. Present address: Electrical and Computer Engineering Department, Clemson University, Clemson, SC 29634-0915, USA. Tel.: + 1-864-656-3377; fax: + 1-864-656-5910.

E-mail addresses: einstein@vision.auc.dk (B.D. Olsen), ahoover@clemson.edu (A. Hoover).

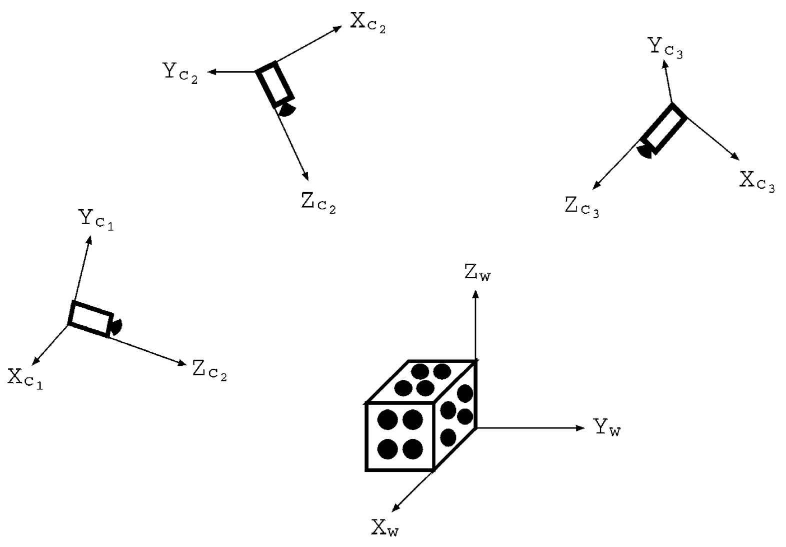

Fig. 1. In a traditional multi-camera system, the fields-of-view of all the cameras overlap. In this case a single target may be used to calibrate the system.

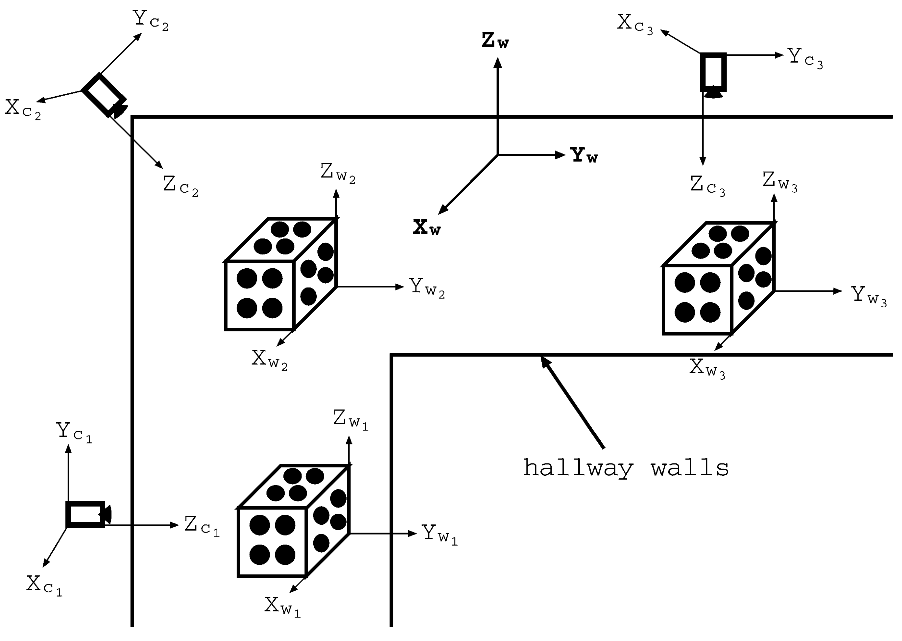

Fig. 2. In a distributed multi-camera system, the fields-of-view of the cameras do not overlap. In this case multiple targets are needed to calibrate the system. A method is required to locate the targets in a common world coordinate system (X_w, Y_w, Z_w).

permanent marking of the environment, or any particular floorspace configuration.

A novel grid-finding algorithm is applied to each image. The grid-finder automatically locates and segments a rectilinear configuration of dots. The user need only specify the size (in dots) of the visible domino configuration, the world coordinates of one of the visible dots, and the world axes orientations and scales relative to the dot axes. The grid-finder produces as output a list of correspondences between 2D image coordinates and 3D world coordinates, one per dot. To solve for the camera model, we use the coplanar solution introduced by Tsai [2,3].

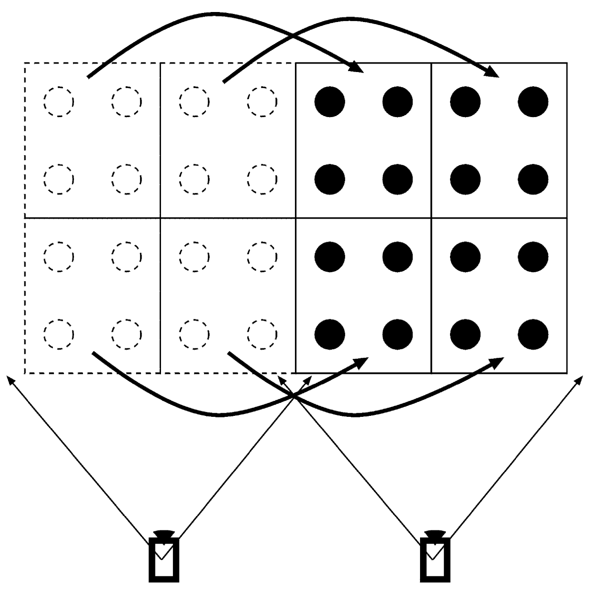

Fig. 3. A limited set of domino calibration targets may be deployed in 'leapfrog' fashion.

A single image is captured from each camera for calibration. The images do not need to be captured simultaneously. A limited set of dominoes may be used in 'leapfrog' fashion (see Fig. 3). The dominoes are deployed to fill the first camera's field-of-view, and its calibration image is acquired. Dominoes are then moved from one side of the initial deployment to the other side of the initial deployment. This motion iterates until the second camera's field-of-view is filled with dominoes, then its calibration image is acquired. This procedure iterates until a calibration image has been acquired for all cameras.

Blank dominoes may be substituted for dotted dominoes to cover floorspace on the boundary of a field-of-view. Blank and dotted dominoes can also be substituted between image captures for different cameras. In this manner, rectilinear dot grids can be presented to each camera, regardless of the amount of field-of-view overlap.

2. Related work

Camera calibration is the process of determining a camera's internal (focal point, lens distortion, etc.) and external (position and orientation) parameters. These parameters model the camera in a reference system in the space being imaged, often called world coordinates. Once calibrated, a camera's 2D image coordinates map to 3D rays in world coordinates.

The standard camera calibration process has two steps. First, a list of 2D image coordinates and their corresponding 3D world coordinates is established. Second, a set of equations using these correspondences is solved to obtain a camera model. Calibrating a distributed multiple camera system requires a third step: The world coordinates of the calibration targets used for each camera must be measured in the same reference system (see Fig. 2).

The majority of the literature on camera calibration considers the issues involved in step two. Issues include the complexity of the camera model (number of parameters), the number and distribution of correspondences required to solve for the model, and the method of solving (analytic or iterative, how many stages, etc.). Tsai [4] and Jain et al. [5] review these issues in the context of machine vision. Talluri and Aggarwal [6] review these issues in the context of mobile robot position estimation. Horaud et al. [7] review these issues in the context of exterior camera calibration. Robert [8] presents an interesting approach in which steps one and two are iteratively solved simultaneously. However, this method still requires an initial close estimate of correspondences. In this work (for step two), we use the popular camera model and coplanar solution method introduced by Tsai [2,3]. These methods are reviewed in the appendix.

This paper presents novel methods concerned with the issues involved in steps one and three. A new suite of applications is emerging, for which unified camera coverage of multiple connected areas is required. Examples include automated surveillance and monitoring (see for instance Refs. [9–12]), intelligent environment control, augmented virtual reality (see for instance Ref. [13]), and distributed robotic sensing and control (see for instance Refs. [14,15]). For these applications, three factors are increasing: the number of cameras, the size and distribution of coverage, and the number of installations. As these factors grow, manual methods become increasingly tedious and impractical.

This work describes two new methods for use in calibrating large numbers of cameras, that observe multiple connecting rooms and corridors, to common world coordinates. First, a domino calibration target is described and evaluated. In multiples, this target can be deployed throughout the observed floorspace, while maintaining a common coordinate system. Second, a novel grid-finding algorithm is described. The grid-finder minimizes the effort required of the user in establishing image-to-world correspondences, particularly for multiple cameras. Experiments are shown to demonstrate the efficacy of these methods. Accuracy and reliability estimates are also reported.

3. Domino calibration target

The purpose of the domino calibration target is to facilitate deployment of calibration points throughout

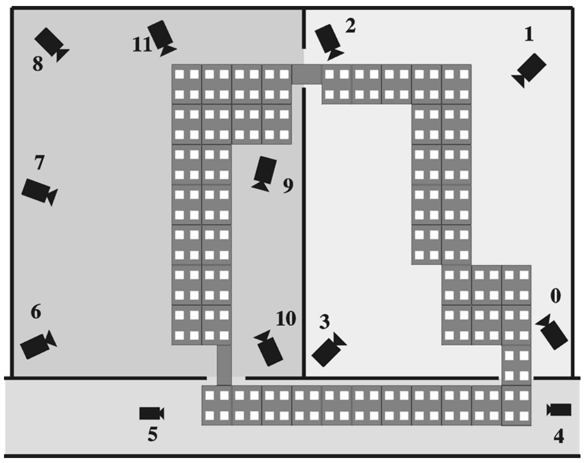

Fig. 4. A sample deployment of a domino calibration grid.

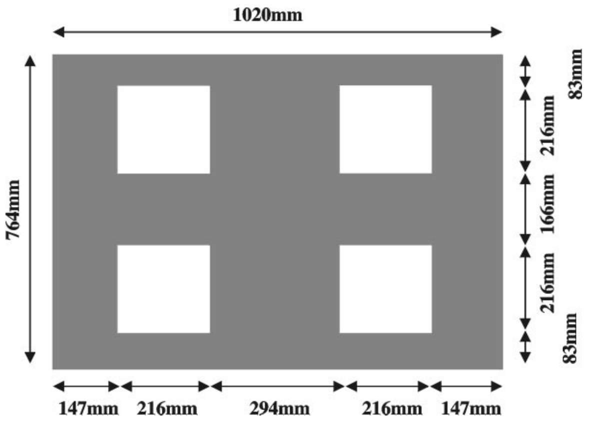

Fig. 5. A domino calibration target, with a 2 × 2 dot pattern.

multiple connecting rooms and corridors, while maintaining a common coordinate system. This is accomplished by constructing multiple identical dominoes, so that when positioned end-to-end and side-by-side, the distances between dots remains constant. Fig. 4 presents an overhead view of a floorplan, consisting of three areas (two rooms and a hallway), observed by 12 cameras, along with a possible deployment of dominoes for calibration. Each domino presents a 2×2 pattern. In this example, dominoes have been deployed in 49 positions (but not necessarily concurrently; see Fig. 3). Additionally, two blank half-tiles have been deployed through doorways.

To construct a domino, we considered the following issues:

1. *Materials*. The material comprising a tile should be light, so as to ease deployment, yet sturdy, so as to avoid non-rigid deformations.
2. *Uniformity*. All tiles must have the same dimensions. Although manual construction methods are possible, machined methods are obviously preferred.
3. *Tile size*. The individual tile size must allow for deployment through doorways. Conversely, the number of tiles required for total deployment should be kept reasonably small, to minimize potential errors in aligning tiles.
4. *Color*. The background (tile base) and foreground (dots) must be reasonably easy to discriminate via digital image analysis.
5. *Dot size*. The individual dot size, and distance between dots, should span a minimum number of pixels when imaged at the maximum expected distance. Conversely, the number of dots per tile should be maximized, for use in solving camera model equations.
6. *Cost*. Depending on the expected floorplan coverage, the required number of tiles may be estimated.

With these issues in mind, we selected an art-style poster-board for a tile base. The poster-board is relatively inexpensive, yet sturdy, easy to acquire in multiples, and is relatively uniform in size and color. The posterboard selected is 1020×764 mm in size, which is narrower than common doorways, yet big enough to span an average room in multiples of 10 or less. We selected white paper for tile dots, which contrasts well against black poster-boards. Each dot, a square 216×216 mm in size, was attached to the poster-board using clear tape. After some experimentation, we found that this dot size filled at least a 30-pixel-square area when imaged at distances up to 8 m using a 6.0 mm lens and a 480×640 frame-grabber. The final dimensions we used are displayed in Fig. 5. We constructed eight such targets, at a cost of approximately $7 (US) in materials each, to experiment in a floorspace of 11.5×14.5 m^2.

To facilitate domino deployment through doorways, two blank tiles were cut in half, one lengthwise (1020×382 mm) and one breadthwise (510×764 mm). The full-size and half-size dominoes were marked at the midpoints, offering three potential alignments (either side or the midpoint) to span narrow areas.

At least two improvements may be hypothesized. First, the complete domino could be constructed by machine, maximizing uniformity. Second, the ends of the dominoes could be fashioned with locking mechanisms. This would simplify the adjoining placement of dominoes (the positioning by hand would not have to be done as carefully). It would also increase the reliability of the adjoining placement of dominoes.

4. Establishing correspondences

The following describes an algorithm for automatically finding rectilinear grids, imaged as described in Section 1. Grid-like structures are often employed as calibration targets (see for instance Refs. [8,16–19]). A grid makes a good calibration target because it (a) uses features with maximum contrast (dots on a background), (b) spans the imaged area evenly (so the resulting camera model is accurate for the entire image), and (c) presents a recognizable macro-configuration (the grid structure). Our novel grid finding algorithm specifically takes advantage of property (c).

A segmentation of the input image is created by applying a single threshold. The grid is then found by searching the segmented regions for the user-specified grid structure. The algorithm takes as input the size of the grid (in rows and columns), the world coordinates of the left-most point (in the image) of the grid, and the world axes orientations and scales of the grid rows and columns. The output of the algorithm is a set of matching image coordinates and world coordinates. These correspondences are then used to solve for a camera model (see the appendix).

The image coordinates are found as the centroids of the domino dots. The projected centroid of a region does not in general coincide with the centroid of a projected region. However, at the scales we are considering (described in Section 3), the error is less than one pixel (described in Section 5). We consider this an acceptable bias, given the advantages region centroids have for automated detection. For applications requiring greater accuracy, our algorithm may still be used to locate the macro grid structure prior to sub-pixel analysis.

The algorithm can function given only the expected grid size, calibrating to an arbitrary world coordinate system. For applications using only a single camera, this calibration may be sufficient. To calibrate multiple cameras to the same world coordinate system, the other above-mentioned inputs are required.

4.1. Preprocessing

The input to the grid finder is a calibration image of the type shown in Fig. 6(a). The grid finder uses three preprocessing steps. First, the calibration image is histogram equalized to make the thresholding robust to different lighting levels. Second, a single threshold is used to separate the foreground dots from the background. Third, the regions in the thresholded image are found by performing a connected components analysis.

The candidate regions for the grid are those within the appropriate range of size (number of pixels). Fig. 6(b) shows an example of candidate regions, where each region has a different color. As can be seen, regions outside the grid have been segmented. The goal of the grid finder is to locate, from among these regions, a grid of the specified size. The grid finder considers each region as a point, computed as the centroid of the region.

4.2. Grid finder

The purpose of the grid finder is to search a set of n points (in our case the centroids of the segmented regions) to find a grid of size $r \times c$ (rows $\times$ columns), where $rc \leqslant n$. A brute force solution to this problem is to select all permutations of $r \times c$ points, checking whether the selected set forms the specified grid. The algorithmic

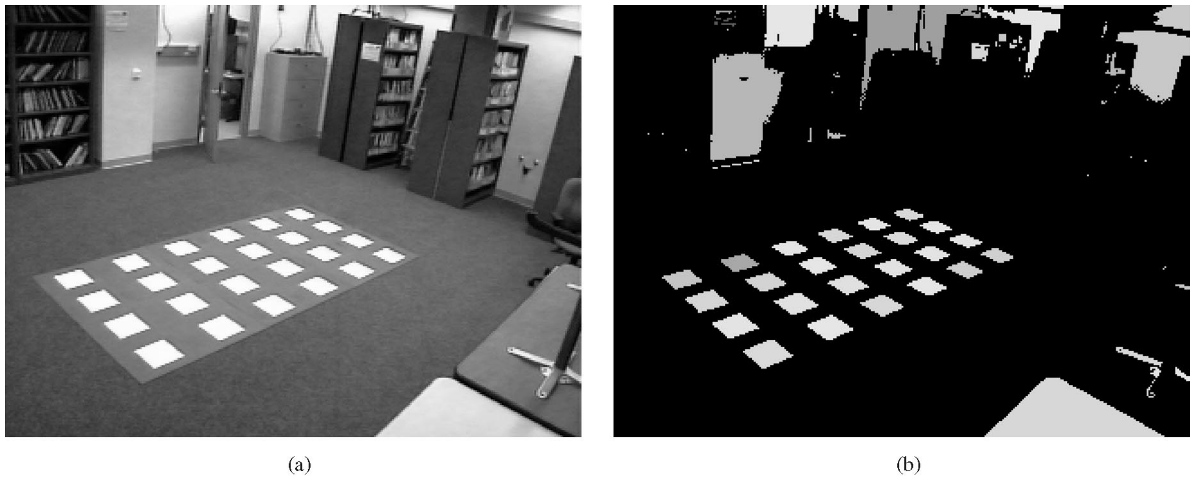
(a) (b)

Fig. 6. A calibration image after preprocessing. The centroids of the candidate grid regions are inputs to the grid finder algorithm. (a) Calibration image. (b) Candidate grid region.

complexity of this approach is $O(n^{rc})$, and thus is impractical for even relatively small grids.

A bootstrap solution to this problem is to select all permutations of x points, where x is the minimum number of points required to solve for the camera model. The image locations of the remaining $rc - x$ points may be solved for by projecting their coordinates using the minimally solved camera model. For Tsai's coplanar solution [3], $x = 5$, yielding a complexity of $O(n^5)$. A search for each of the remaining $rc - x$ grid points in the set of $n - x$ candidate points has a complexity of $O(rcn)$. The complexity of this bootstrap solution is therefore $O(rcn^6)$.

If the effects of internal camera parameters (except focal length) and lens distortion may be ignored, then $x = 4$ is sufficient to solve for a camera model [5]. This reduces the complexity of the bootstrap solution to $O(rcn^5)$. In any bootstrap solution, a hidden constant in the complexity is the time necessary to solve for a camera model. This step requires a nonlinear regression, which is costly.

Instead of searching for a grid, we propose an approach that searches for the grid corners. If the row and column distances between neighboring grid points are constant (for instance, as after an orthographic projection), then given three corner positions, the rest of the grid positions may be solved for by interpolation (see Fig. 7(a)). The complexity of this search is $O(n^3)$, since each permutation of three points must be tested. Given the corners, a search for each of the remaining $rc - 3$ grid points in the set of $n - 3$ candidate points has a complexity of $O(rcn)$. The complexity of the interpolation solution is therefore $O(rcn^4)$.

For a grid with non-uniform spacing (for instance, as after a perspective projection), the unknown spacing between interior grid points prevents a direct interpolation from the corners (see Fig. 7(b)). However, a line validation test may be used to determine whether or not the required number of points lies between two given points (when the size of the grid is known). The successive application of this test, first between corner points, and then between border points, is sufficient to validate a grid. The solution takes four steps, as illustrated in Fig. 8.

1. Find the topmost line.
2. Find the bottommost line.
3. Find the columns connecting these lines.
4. Find the rows connecting the two outermost columns.

The first step consists of selecting candidates for i and j (top corners of the grid, see Fig. 8) from n until a valid line is found. The procedure for testing whether or not a line is valid is outlined below. If a valid line is found, as in Fig. 8(a), then the algorithm continues to step two, otherwise it exits unsuccessfully. The second step consists of selecting candidates for k and l (bottom corners of the grid, see Fig. 8) from n until a valid line is found. If a valid line is found, as in Fig. 8(b), then the two lines are assumed to be the top- and bottommost lines in the grid. The third step validates the columns of the grid. The endpoints of the columns are selected as pairs of opposing points from the top and bottom rows, as illustrated in Fig. 8(c). In the fourth step the rows are validated by searching for lines between pairs of opposing points on the left and right most columns, as illustrated in Fig. 8(d). The left and right most columns are found as the lines spanned by i and k and j and l, respectively. The grid is found if all rows and columns in the grid are validated.

The grid finder algorithm is formally described by the following pseudo code:

```
for i = 1 to (total points-rows * columns)
  for j = 1 to total points
    if (validate_line(i,j) = success)
      for k = 1 to total points
        for l = 1 to total points
          if (validate_line(k,l) = success)
            for (m,n) = (points in line from
                         i to j,
                         points in line from
                         k to l)
              if (validate_line(m,n)
               = failure)
                select next l
              end if
            end for (m,n)
            for (m,n) = (points in line from
                         i to k,
                         points in line from
                         j to l)
              if (validate_line(m,n)
               = failure)
                select next l
              end if
            end for (m,n)
            exit successfully - grid found.
          end if
        end for
      end for
    end if
  end for
end for
exit unsuccessfully - no grid found.
```

The complexity of this algorithm is $O((r + c)n^5)$, which improves upon all other methods handling perspective projection. One drawback to this approach is that there must be at least three rows and columns for the solution to work, since any two points may pass the line validation test.

The algorithm can be further improved by ordering the selection of points for i, j, k and l according to their relative positions. The candidates for i, j, k and l should be selected in order from the left, right, top and bottom

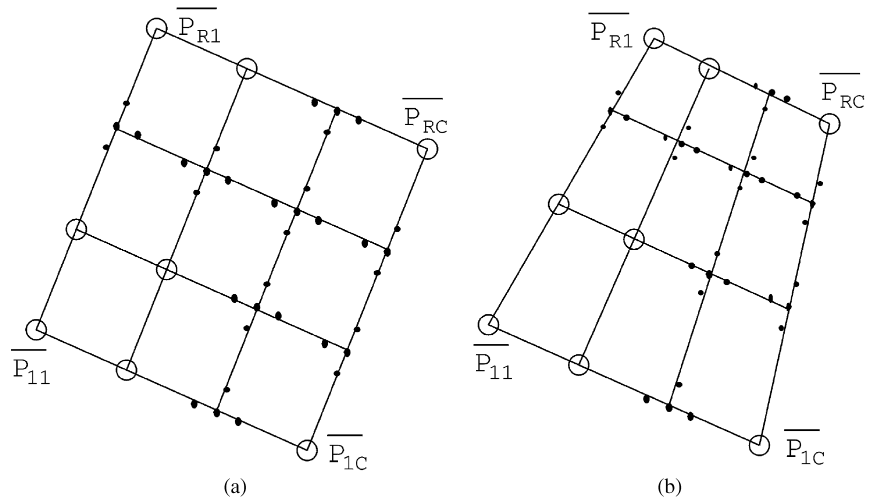

Fig. 7. Under orthographic projection, grid points may be interpolated using the locations of the corners: $\overline{P_{rc}} = \overline{P_{11}} + (r/(R-1))\overline{P_{R1} - P_{11}} + (c/(C-1))\overline{P_{1C} - P_{11}}$. Under perspective projection, the unknown nonlinear spacing between points prevents a direct solution. (a) Grid under orthographic projection. (b) Grid under perspective projection.

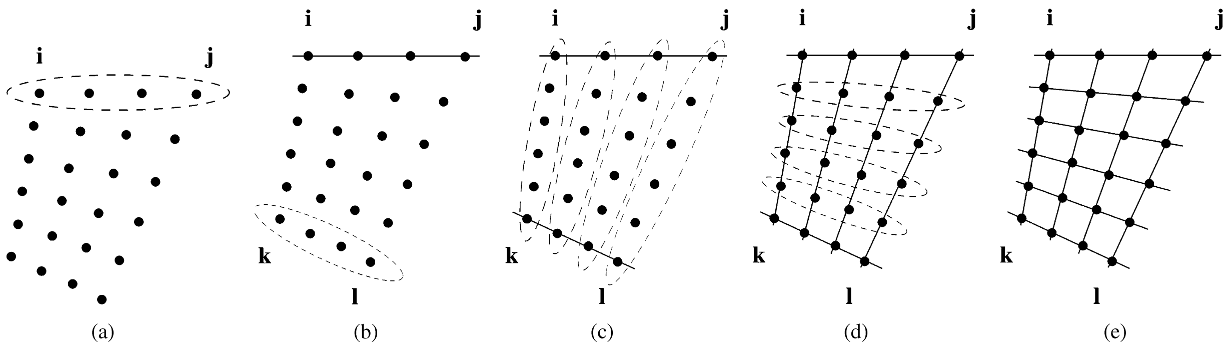

Fig. 8. The four steps in the grid finder: (a) the top line is found, (b) the bottom line is found, (c) the columns are found, (d) the rows are found and (e) the final result.

sides of the image, respectively. Although this selection scheme does not change the complexity of the algorithm, it does improve the performance, particularly when $rc \approx n$.

4.3. Line validation

The inputs to the line validation function are the two endpoints of the line, A and B, the number of points on the line, n_L, and a list of candidate points. The function goes through the list of candidate points and for each point C computes the distance from C to AB,

$$D(C, AB) = \frac{(A_x - B_x)(C_y - B_y) - (A_y - B_y)(C_x - B_x)}{\sqrt{(B_x - C_x)^2 + (B_y - C_y)^2}} \tag{1}$$

and ratios characterizing whether or not C lies in-between A and B:

$$I_x = \frac{A_x - B_x}{C_x - B_x} \quad \text{and} \quad I_y = \frac{A_y - B_y}{C_y - B_y}. \tag{2}$$

The candidate point C is found to lie within the line segment AB if

$$D(C, AB) < T_{\text{close}} \quad \text{and} \quad 0 < I_x < 1 \text{ and } 0 < I_y < 1. \tag{3}$$

The situation for a single point is shown in Fig. 9.

As the function proceeds through the list of candidate points, a running count of the number of points found to

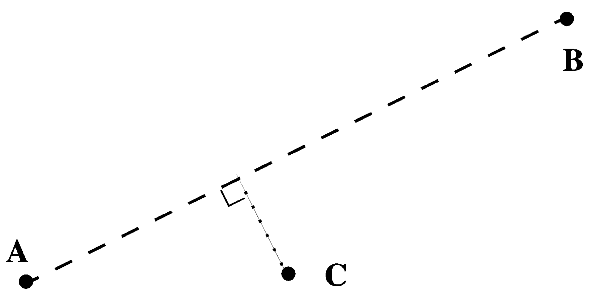

Fig. 9. The point C is tested against the line between A and B.

lie within AB is kept. If this count for the entire list equals n_L, then the function returns successfully. As soon as the count exceeds n_L, or if the entire list is processed and the count does not reach n_L, then the function returns unsuccessfully.

5. Experiments

To make the calibration process easier a graphical user interface has been constructed. From the interface it is possible to control the calibration of multiple cameras. The interface offers the option to select the input camera, as in Fig. 10(a), and adjust the algorithm parameters. The image on screen is a live video signal. This is especially useful while positioning dominoes. When the user chooses to perform a calibration, the program prompts for the size of the grid to look for, as in Fig. 10(b). If the grid finder is successful it prompts the user for the world coordinates of the left most point, as in Fig. 10(d), the orientations and scales of the world axes, and finally produces a set of matching world coordinates and image coordinates. This data set is then used as input for Tsai's coplanar solution to produce a camera model (see the appendix).

Experiments were performed to determine the efficacy and accuracy of the proposed methods. Twelve cameras were positioned to observe two rooms and a hallway, as

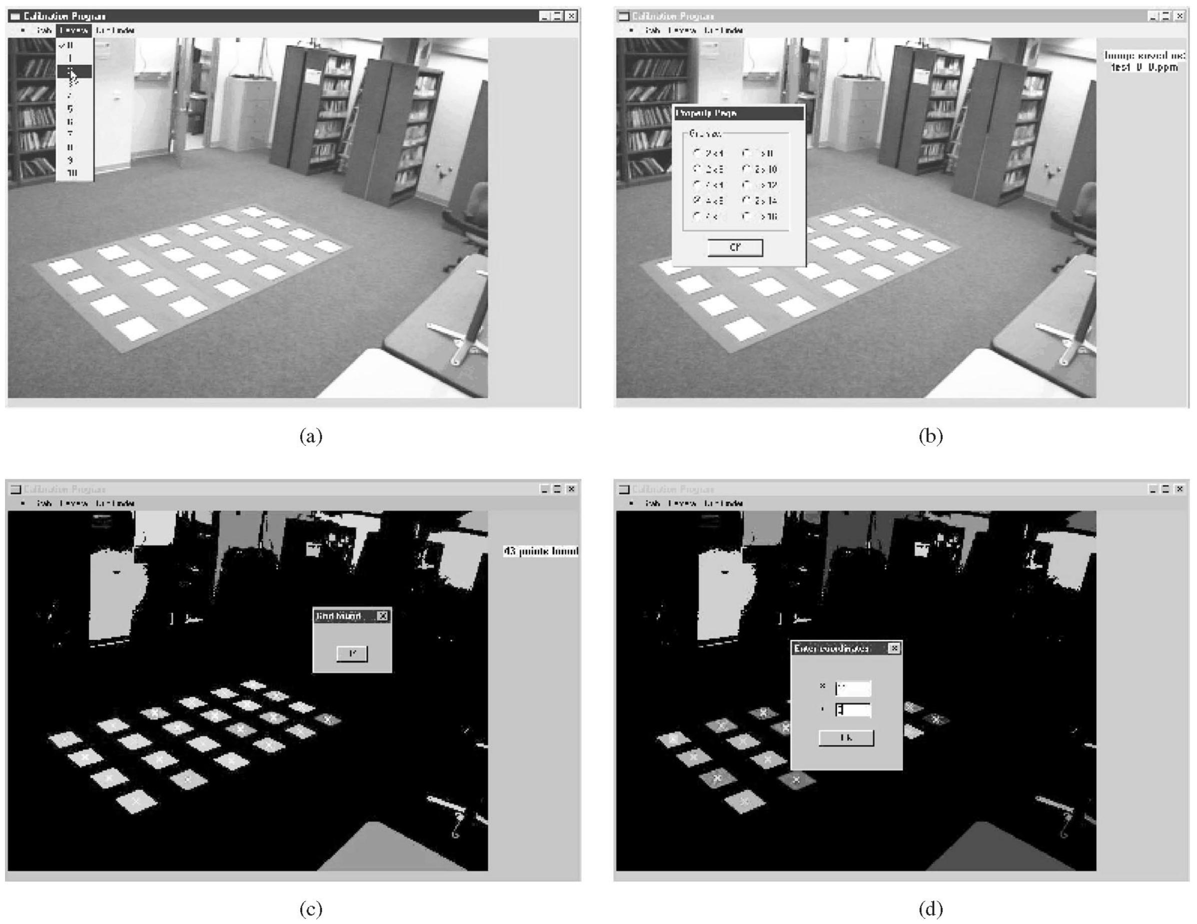

Fig. 10. Snapshots from the graphical user interface. (a) Camera selection. (b) Grid size. (c) Grid found. (d) Coordinates of left most point.

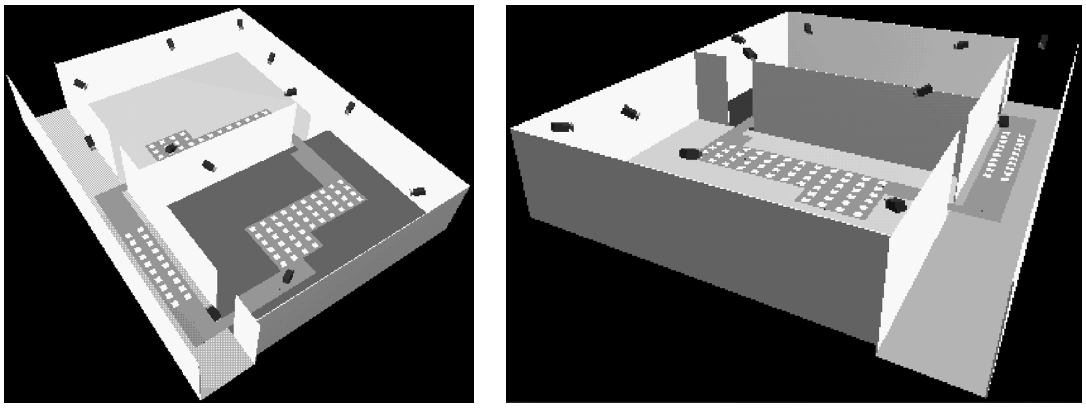

Fig. 11. Camera and grid positions for calibrated environment.

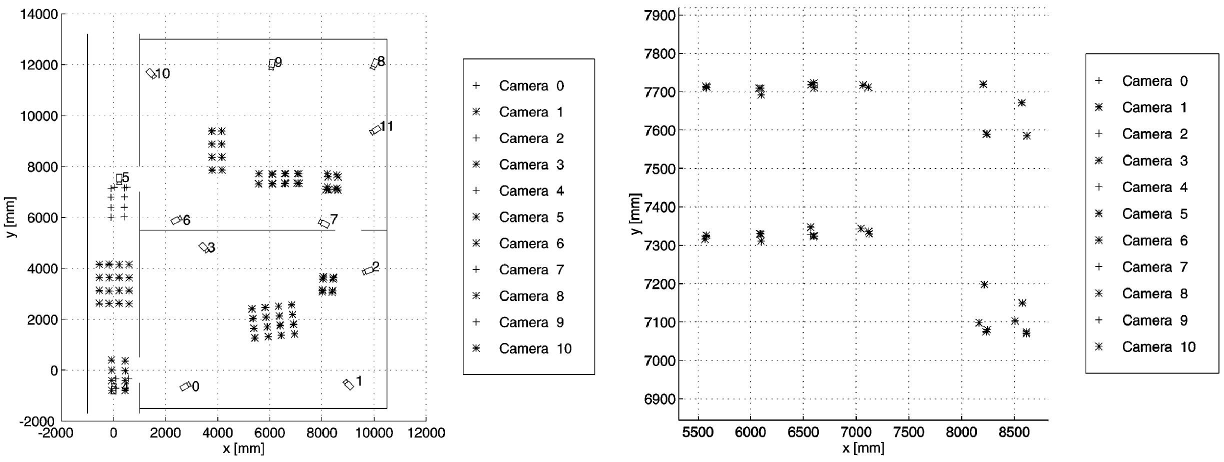

Fig. 12. A plot of points as observed by the camera network. A portion of the plot is zoomed to better discern the scale of the agreement of the cameras on point position.

depicted in Fig. 4. The cameras labeled 0, 2, 4, 5, 9 and 10 are Sony XC-999 CCDs with 6.0 mm lenses. The cameras labeled 1 and 3 are Sony XC-999 CCDs with 3.5 mm lenses. The cameras labeled 7 and 8 are Sony camcorders (model CCD-TR700). The camera labeled 6 is a Kodak camcorder (model E440). The camera labeled 11 is an SGI O2 digital camera.

Three doorways connect the areas in Fig. 4. Dominoes were deployed in a loop through these doorways, starting and ending at the same location. The eight dominoes we constructed were placed in 'leapfrog' fashion: The position of the domino deployed last in the previous leap was maintained, becoming the domino first deployed in the current leap. The loop spanned nine tiles (18 dots) in its furthest extent in the X direction and 12 tiles (24 dots) in its furthest extent in the Y direction. Thus, 42 side-by-side and end-to-end domino adjoinments were needed (minimally) to complete the loop.

The domino grid was deployed and imaged in three different patterns for calibrating all 12 cameras. Fig. 11 shows a 3D snapshot of the domino grid and camera models computed for the third test, superimposed upon the environment walls and floor. The tests took 105, 55, and 30 min, conducted by a two-person team. The differences in time reflect the learning curve associated with the method. The most important item learned was how to arrange blank and dotted dominoes for individual camera snapshots, so as to maximize the visible points while keeping a rectilinear grid in the field-of-view. Most of the time for each calibration test was spent in positioning the dominoes. The complete processing for each image takes less than 1 min. The average distance be-

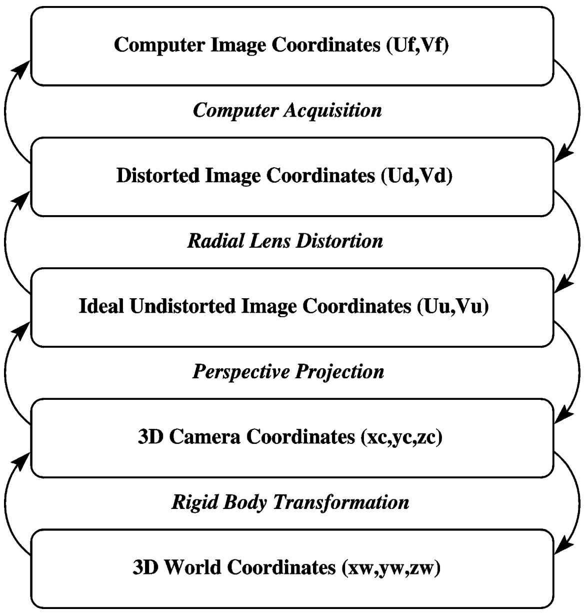
Computer Image Coordinates (Uf,Vf)
Computer Acquisition
Distorted Image Coordinates (Ud,Vd)
Radial Lens Distortion
Ideal Undistorted Image Coordinates (Uu,Vu)
Perspective Projection
3D Camera Coordinates (xc,yc,zc)
Rigid Body Transformation
3D World Coordinates (xw,yw,zw)

Fig. 13. Transformation between computer image coordinates and 3D world coordinates.

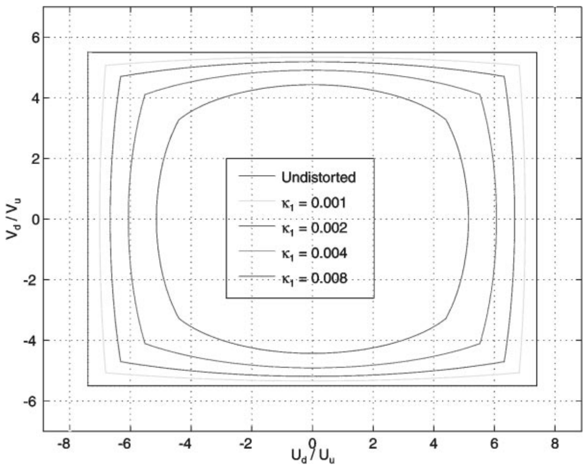

Fig. 14. Examples of the effect of radial lens distortion for various values of κ_1.

tween the first and last tiles placed (which completed the loop) was 50 mm. This error averages to 1.2 mm per tile adjoinment.

Using the results of the third calibration test, dominoes were imaged throughout the combined floorspace, positioned at unknown world coordinates. For each camera, the visible dot centroids were computed and projected back to the floor plane in world coordinates. These coordinates were then plotted and compared for alignment between cameras (with overlapping fields-of-view), to test the accuracy of the camera network's coverage. Fig. 12 shows an overhead view of the camera positions and the back-projected points, color-coded for each camera. In the global view, the point alignment from multiple cameras is too close to discern. In the zoomed view, the point alignment decreases at larger distances from cameras, as expected. For instance, the points in the zoomed view seen by camera one are far away (they are viewed through the doorway to the next room).

Another experiment was done to test the accuracy of the entire process. Cameras 0, 1, 2 and 3 were calibrated nine times using 16 points (4×4 grids) randomly translated and rotated throughout the room. The calibration dot centroids were projected back to the floor plane in world coordinates, and compared to the values used for calibration. The average error was 4.3 mm, with a 2.7 variance. The world coordinates used for calibration were also projected forward to the image plane, and compared to the calibration dot centroids. The average error was 0.28 pixels, with a 0.01 variance.

6. Conclusions and discussion

In this paper, novel methods were described to calibrate multiple cameras that observe multiple connecting rooms and corridors to a common coordinate system. A domino calibration target was described, along with a grid finder algorithm, to expedite the calibration process. Experiments were shown to demonstrate the efficacy and accuracy of the approach.

We imagine accuracies of 4.3 mm in world coordinates and 0.28 pixels in image coordinates should prove acceptable for a variety of applications, including surveillance, augmented virtual reality, and intelligent environment control. We are pursuing using this video network to track and control mobile robots [20]. An advantage for these applications is that the camera network covers an extensive (combined) field-of-view, while remaining stationary. Although cameras can be actively calibrated while moving (see for instance Ref. [21]), data fusion problems become much simpler using stationary sensors (see for instance Ref. [10]).

Five hours were required to mount, aim, and wire the 12 cameras used in these experiments to a central video patch board. The entire calibration process took (for our third test) one-half hour. We imagine our methods could save a great deal of time and effort for other parties working with distributed multi-camera networks.

7. Summary

This work considers the problem of calibrating a distributed multi-camera system. As opposed to traditional multi-camera systems, such as stereo heads, in a distributed network the fields-of-view do not all overlap. Methods are needed to align multiple distributed calibration targets in a common coordinate system. Methods are also needed in which the required effort scales reasonably as the number of cameras and installations increases. Our novel method uses a grid of domino calibration targets. We describe and evaluate a domino calibration target that may be constructed from inexpensive materials. We also describe a novel grid-finding algorithm, to expedite the location of image-to-world correspondences. To solve for the camera model, we use the coplanar solution introduced by Tsai [2,3]. Experiments conducted in a hallway and two connecting rooms, using 12 cameras, demonstrate accuracies of 4.3 mm in world coordinates and 0.28 pixels in image coordinates. This calibration method should prove useful for a variety of applications, including distributed sensing, robotics, augmented virtual reality, and intelligent environment control.

Appendix: Camera model and solution method

After using our methods to establish a set of correspondences, we use the camera model and coplanar solution method introduced by Tsai [2,3]. We have found it to be robust for a variety of industrial and commercial cameras and lenses, and a dependable implementation is publically available [22]. Here we review the model, using the terminology adopted from Tsai. The model describes a set of transformations between a digitized (or computer) image space and the world space as outlined in Fig. 13.

Coordinates in the world space are referred to as 3D world coordinates and are represented by a three-dimensional column vector

$$P_w = [x_w \quad y_w \quad z_w]^{\mathrm{T}}. \tag{A.1}$$

The coordinates in a camera space are referred to as 3D camera coordinates and are likewise represented by a three-dimensional column vector

$$P_c = [x_c \quad y_c \quad z_c]^{\mathrm{T}}. \tag{A.2}$$

The rigid body transformation between 3D world coordinates and 3D camera coordinates is written as

$$P_c = RP_w + T, \tag{A.3}$$

where T is the translation vector,

$$T = [T_x \quad T_y \quad T_z]^{\mathrm{T}} \tag{A.4}$$

and R is the 3×3 rotation matrix,

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \tag{A.5}$$

The rotation matrix is defined as three separate rotations θ_x, θ_y and θ_z around the x, y and z axes:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta_x) & -\sin(\theta_x) \\ 0 & \sin(\theta_x) & \cos(\theta_x) \end{bmatrix}, \tag{A.6}$$

$$R_y = \begin{bmatrix} \cos(\theta_y) & 0 & \sin(\theta_y) \\ 0 & 1 & 0 \\ -\sin(\theta_y) & 0 & \cos(\theta_y) \end{bmatrix}, \tag{A.7}$$

$$R_z = \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) & 0 \\ \sin(\theta_z) & \cos(\theta_z) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{A.8}$$

The product of these rotations yields the rotation matrix

$$R = R_z R_y R_x. \tag{A.9}$$

The perspective projection transforms points from the camera space (3D camera coordinates) to ideal undistorted image coordinates (U_u, V_u). This is done using a pin-hole camera model, which results in the following equations:

$$U_u = f\frac{x_c}{z_c}, \tag{A.10}$$

$$V_u = f\frac{y_c}{z_c}, \tag{A.11}$$

where f is the effective focal length of the camera.

There are two kinds of lens distortion: tangential and radial. For each kind of distortion an infinite series is required. In Ref. [3] it is argued that for most industrial applications only radial lens distortion needs to be modeled, and only by one term. This gives the following set of equations that relate distorted image coordinates (U_d, V_d) to ideal undistorted image coordinates

$$U_d = \frac{U_u}{1 + \kappa_1 r^2}, \tag{A.12}$$

$$V_d = \frac{V_u}{1 + \kappa_1 r^2}, \tag{A.13}$$

where κ_1 is the distortion coefficient and

$$r = \sqrt{U_d^2 + V_d^2}. \tag{A.14}$$

The effect of the radial lens distortion is illustrated in Fig. 14 for various values of κ_1. The values of κ_1 for the cameras used in this work was found to be in the range 0.001–0.007.

The digitization of distorted image coordinates is described by

$$U_f = \frac{s_x}{d'_x} U_d + C_x, \tag{A.15}$$

$$V_f = \frac{1}{d_y} V_d + C_y, \tag{A.16}$$

$$d'_x = d_x \frac{N_{cx}}{N_{fx}}, \tag{A.17}$$

where (U_f, V_f) is the digitized image coordinates, (C_x, C_y) is the center of the digitized image, d_x and d_y are the center to center distance between adjacent sensor elements in the x and y direction respectively, N_{cx} is the number of sensor elements in the x direction, N_{fx} is the number of pixels in a line as sampled by the computer, and s_x is the uncertainty scale factor. The digitized image coordinates (U_f, V_f) are also referred to as (x, y) coordinates in the image space.

The described model has a number of parameters whose values need to be estimated. The parameters related to the frame-grabber are determined from the manual of the frame-grabber, as provided by the manufacturer. The parameters related to the frame-grabber are d_x, d_y, N_{cx} and N_{fx}. The five intrinsic parameters of the model that need to be estimated are f, κ_1, C_x, C_y and s_x. The six external parameters that need to be estimated are $\theta_x, \theta_y, \theta_z, T_x, T_y$ and T_z. The parameter estimation itself is a two-step procedure, as described in detail in Ref. [3]. We use the coplanar case because the calibration points as described in this work are all in the same plane. This means that the parameter s_x cannot be estimated and is therefore set to 1, as recommended by Tsai.

References

[1] F. Pedersini, A. Sarti, S. Tubaro, Multi-Camera Systems, IEEE Signal Process. 16 (3) (1999) 55–65.

[2] R.K. Lenz, R.Y. Tsai, Techniques for calibration of the scale factor and image center for high accuracy 3D machine vision metrology, IEEE Trans. Pattern Anal. Mach. Intell. 10 (5) (1988) 713–720.

[3] R.Y. Tsai, A versatile camera calibration technique for high accuracy 3D vision metrology using off-the-shelf tv cameras and lenses, IEEE Trans. Robotics Automat. 3 (4) (1987) 323–344.

[4] R.Y. Tsai, Synopsis of recent progress on camera calibration for 3D machine vision, in: O. Khatib, J.J. Craig, T. Lozano-Perez (Eds.), The Robotics Review, MIT Press, Cambridge, MA, 1989, pp. 147–159.

[5] R. Jain, R. Kasturi, B.G. Schunck, Calibration Machine Vision, McGraw-Hill Inc., New York, 1995, pp. 309–362 (Chapter 12).

[6] R. Talluri, J.K. Aggarwal, Position estimation techniques for an autonomous mobile robot – a review, in: C.H. Chen, L.F. Pau, P.S.P. Wang (Eds.), Handbook of Pattern Recognition and Computer Vision, 1993, pp. 769–801.

[7] R. Horaud, B. Conio, O. Leboulleux, An analytic solution for the perspective 4-point problem, Comput. Vision Graphics Image Process. 47 (1989) 33–44.

[8] L. Robert, Camera calibration without feature extraction, Comput. Vision Image Understanding 63 (2) (1996) 314–325.

[9] D. Gibbins, G.N. Newsam, M.J. Brooks, Detecting suspicious background changes in video surveillance of busy scenes, Proceedings of Third IEEE Workshop on Applications of Computer Vision, Sarasota, FL, 1996, pp. 22–26.

[10] A. Hoover, B. Olsen, A real-time occupancy map from multiple video streams, Proceedings of IEEE International Conference on Robotics and Automation, 1999, pp. 2261–2266.

[11] B.S.Y. Rao, H.F. Durrant-Whyte, J.A. Sheen, A fully decentralized multi-sensor system for tracking and surveillance, Int. J. Robotics Res. 12 (1) (1993) 20–44.

[12] T.M. Strat (Ed.), Video surveillance and monitoring, Section 1 of Volume 1 of the Proceedings of Image Understanding Workshop, New Orleans, LA, 1997, pp. 3–377.

[13] M. Tuceryan et al., Calibration requirements and procedures for a monitor-based augmented reality system, IEEE Trans. Visualization Comput. Graphics 1 (3) (1995) 255–273.

[14] A. Hoover, B. Olsen, Path planning for mobile robots that sense using a video camera network, Proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechatronics, 1999.

[15] H. Ishiguro, Distributed vision system: a perceptual information infrastructure for robot navigation, Proceedings of 15th International Joint Conference on Artificial Intelligence, August 1997.

[16] C. Chen, Y.F. Zheng, A new robotic hand/eye calibration method by active viewing of a Checkerboard pattern, Proceedings of IEEE Conference on Robotics and Automation, Vol. 2, 1993, pp. 770–775.

[17] J.Z.C. Lai, On the sensitivity of camera calibration, Image Vision Comput. 11 (10) (1993) 656–664.

[18] G.-Q. Wei, S.D. Ma, Implicit and explicit camera calibration: theory and experiments, IEEE Trans. Pattern Anal. Mach. Intell. 16 (5) (1994) 469–480.

[19] J. Weng, P. Cohen, M. Herniou, Camera calibration with distortion models and accuracy evaluation, IEEE Trans. Pattern Anal. Mach. Intell. 14 (10) (1992) 965–980.

[20] B.D. Olsen, Robot navigation using a sensor network, Master's Thesis, Laboratory of Image Analysis, Aalborg University, Denmark, June 1998.

[21] A. Basu, Active calibration of cameras: theory and implementation, IEEE Trans. Systems Man Cybernet. 25 (2) (1995) 256–265.

[22] R. Willson (maintainer), Tsai Camera Calibration Software, http://www.ius.cs. cmu.edu/afs/cs.cmu.edu/user/rgw/www/TsaiCode.html.

About the Author—BENT OLSEN received his M.S. degree (1998) in Electrical Engineering from Aalborg University, Denmark. During that time his research focused on medical image processing and sensor network control of mobile robots. From 1997 to 1998 Mr. Olsen was a visiting scholar at the University of California, San Diego. In 1998 Mr. Olsen held a position as a Research Assistant at Aalborg University, where his research focused on virtual reality. Since 1999 Mr. Olsen has been working at Praja, Inc. on multimedia content management systems.

About the Author—ADAM HOOVER received a B.S. (1992) and M.S. (1993) in Computer Engineering, and a Ph.D. (1996) in Computer Science and Engineering, all from the University of South Florida. During this time his research focused on range image processing and 3D model construction for object recognition and mobile robot navigation. From 1996 to 1998 Dr. Hoover held a post-doctoral position at the University of California, San Diego, in the Electrical and Computer Engineering Department. During this time his research focused on medical (retinal) image processing, data fusion for multiple video streams, and sensor network control of mobile robots. In January of 1999 Dr. Hoover joined the Electrical and Computer Engineering Department of Clemson University as an Assistant Professor. His research continues on the aforementioned projects, and also blends with computer and robot architecture issues as they relate to machine vision.