

Calibration of a Multicamera Network

Patrick T. Baker

Yiannis Aloimonos

Center for Automation Research
University of Maryland
College Park, MD 20742

Center for Automation Research
University of Maryland
College Park, MD 20742

Abstract

With the advent of laboratories containing dozens of cameras, and the possibility of laboratories containing hundreds of cameras, the question of how to calibrate all the cameras has become pressing. While it is certainly possible to calibrate these networks in a labor intensive manner, a simple, stable, and accurate calibration method is still needed. This paper presents such a method, based on textures printable on a laser printer and mounted on a board. We will show what the problems with the current methods are, and show how these problems can be overcome with a novel use of a trilinear constraint related to the vanishing point constraint, which we call the primatic line constraint. High accuracy with little user effort is achieved with this method.

1 Introduction

Calibration is an integral part of any multicamera lab and must be addressed before any data can be used. There are remarkably few papers on the calibration problem as applied to many cameras, because it has been thought that standard calibration techniques would prove adequate to calibrate the new laboratories, and that new techniques would not be necessary. We explain briefly why these techniques break down with large camera networks, then explain what is necessary for a good solution, then show our solution and why we believe it satisfies our requirements.

1.1 Possible Calibration Methods

The calibration method used in photogrammetry [8] has been to construct a 3D object which has been precisely measured. This method, while highly accurate and fast, can be expensive to construct (or purchase) and maintain. Second, an operator needs to look at the output from each camera, and this is infeasible for networks with more than a few cameras. Third, the 3D object would have to be constructed

for a particular network and thus a camera configuration change could render the calibration object useless.

If we do not use a 3D object, then we are forced to use some kind of structure from motion techniques where we simultaneously find the internal and external calibration parameters for all our cameras. This problem is well-studied for the case of two cameras [3].

The second problem could be solved with a technological solution such as having LEDs blink in a specified sequence. Unfortunately, the LEDs are difficult to measure more accurately than a pixel, because if the LED image is contained within a single pixel, there is no way to tell where it is. There are ways to get around this, but there are simpler ways to obtain calibration, as we shall see.

We tried using a single corner as the source for our points [1], but we sometimes had trouble with spurious point detection due to the clothing of the data collector. Also, even when we obtained clean data, the optimization tended to be unstable due to the projective ambiguity. While this can be solved to some extent with the addition of some metric information [6, 4] (such as right angles, or a constant distance), the solution was not as stable as we might have wished, and was not accurate as accurate as we would have hoped with regard to nonlinear calibration [9].

The homography based methods [11, 10] combine the flexibility of an easily generated data set with the accuracy and stability given by having known distances between points. These methods are excellent for easy calibration of one or a few cameras. Since the point extraction is based on the intersection of two lines, this step is accurate, and was attractive to us for implementation. Unfortunately, the data collection requires the user to show a pattern to each camera, an $O(n)$ requirement (n being the number of cameras), and also requires the user to click on points before the calibration, another $O(n)$ requirement. We therefore could not use these methods. However, we did take away the valuable knowledge that the homography based methods can be very stable and accurate, because they are based on entire patterns for which the absolute position of the corners on the pattern is known. Picking an appropriate projective dis-

ortion is then not a factor in this case, and even nonlinear calibration can be obtained accurately with a simple technique.

Our conclusion was that the problem of calibrating networks of dozens (or more) of cameras is qualitatively different from the problem of calibration of just a few cameras. Many engineering issues, manageable when calibrating a few cameras, become overwhelming in the case of dozens (or more) of cameras. For these large networks of cameras, we need new methods of calibration so that we do not need to be so careful with each individual camera yet can still get high accuracy.

1.2 Requirements for New Solution

From an engineering (as opposed to geometric) point of view, let us discuss what is necessary for a many camera calibration solution. Such considerations for the single camera calibration were discussed in [5]. First of all, the standard calibration parameters [2] are not as important as the ability to reconstruct the Euclidean geometry of the viewing space as accurately as possible. We use a reprojection error to determine whether we have calibrated our viewing space properly.

The calibration should involve some object which is easy to create. A particularly cheap and accurate object then is some sort of printed sheet mounted on a rigid board, as in the homography methods. Our camera networks may be quite large, and depending on how much the fields of view of the cameras overlap, we may need a larger or smaller board. The larger the pattern the more likely that many cameras will have at least some view of the pattern, but the more difficult it is to keep the board rigid. Beyond noting this we do not delve into the mechanical properties of our boards.

We are not concerned with minimizing the number of views of our texture, because we are using video cameras. Therefore we always assume in this paper that we have many views of the pattern, and that the pattern is positioned in multiple orientations everywhere in the space that we anticipate our objects of interest will be in.

We don't want to have to confirm that the entire board is in any particular frame, an impossible task when you have many cameras viewing from different directions. Therefore we need a method which will be able to use frames in which only a part of the board is visible. Linked to this is a desire to make sure that the entirety projection of the viewing volume is well-calibrated for all cameras, no matter whether the pattern is in or out of the depth of field of the camera.

The method should also be stable in the sense that if the input is reasonable then the calibration should proceed from just the image sequences to the calibration parameters with no other input necessary.

Finally, the method should be highly accurate, enabling

us to carve voxels precisely, or use stereo with high accuracy. We believe that our method addresses all these requirements.

2 Camera Geometry and Calibration

We describe in this section our camera model and the constraints we use to determine our camera calibration.

2.1 Cameras and Points

We define world points in three dimensions without homogeneous coordinates.

Definition 1 A world point is an element $P \in \mathbb{R}^3$. We may represent such a point in a particular coordinate system as:

$$\mathbf{P} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (1)$$

Our image points exist in the projective plane \mathbb{P}^2 .

Definition 2 An image point is an element $p \in \mathbb{P}^2$. We may represent such a point in a particular coordinate system as:

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \quad (2)$$

These coordinates are homogeneous.

A camera takes world points and projects them into image points. Usually this would be accomplished using a 3×4 projection matrix. However, since we are calibrating cameras, and we have nonlinear terms in our projection equations, we define our camera as follows.

Definition 3 A camera C is a map $C : \mathbb{R}^3 \rightarrow \mathbb{P}^2$ from world points to image points. If our world points are coordinatized in a fiducial coordinate system, we may represent this map with a pair (B, \mathbf{T}) , where $B : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ is a nonlinear function, and \mathbf{T} is a 3-vector representing the camera center. The action of the map on a world point coordinate \mathbf{P} is:

$$C(\mathbf{P}) = B(\mathbf{P} - \mathbf{T}) \quad (3)$$

where $C(\mathbf{P})$ is considered as a member of \mathbb{P}^2 .

For every camera we would like to find this B and \mathbf{T} , which will be the calibration parameters for our multicamera network. Here we have defined a camera as taking a world point and mapping it to an image point through a nonlinear transformation on the world coordinates. In the interest of simplicity, we will treat B as a linear function for most of the paper, although our method works well for nonlinear calibration.

2.2 Cameras and Lines

Definition 4 An image line is represented by $\ell \in \mathbb{P}^2$. The line may be given coordinates

$$\ell = \begin{bmatrix} \ell_1 \\ \ell_2 \\ \ell_3 \end{bmatrix} \quad (4)$$

The point p and line ℓ must satisfy

$$\mathbf{p}^\top \ell = 0 \quad (5)$$

We use the following to go between our calibration equations for points and those for lines.

Proposition 1 If we linearly transform image points \mathbf{p} as $\hat{\mathbf{p}} = B\mathbf{p}$ with a matrix B , then the image line segment ℓ is transformed by $\hat{\ell} = B^{-\top}\ell$.

Also, in this paper we use both *calibrated* image points (and lines) and *uncalibrated* image points (and lines). A calibrated point \mathbf{p} (line ℓ) exists in a coordinate frame which is only a translation away from the fiducial frame, while the uncalibrated point $\hat{\mathbf{p}}$ (line $\hat{\ell}$) exists in the actual camera coordinate system, so that we have:

$$\mathbf{p} = B^{-1}\hat{\mathbf{p}} \quad (6)$$

$$\ell = B^\top \hat{\ell} \quad (7)$$

2.3 Constraints

The vector representing a calibrated image line is perpendicular to all the calibrated image points on the line. This line vector is thus perpendicular to the plane containing the calibrated image points and the center of projection. This in turn means that our image line must be perpendicular to the direction of the world line, since that world line is wholly contained in the plane.

We may use this observation to form a constraint which operates independently of translation. In particular, the cameras may be separate or identical. Also, the image lines can have been created from a single line or two parallel lines. Because all of our lines, when put in the fiducial coordinate system, must be perpendicular to the line direction, we obtain a type of vanishing point constraint which operates on three cameras.

Proposition 2 If we have one, two, or three parallel world lines, and three cameras with rotation/calibration matrices B_i , then if these three cameras view images of one of our world lines as $\hat{\ell}_i$, with the lines not necessarily the same in all cameras, then we obtain **the prismatic line constraint**.

$$\hat{\ell}_2^\top B_2 (B_1^\top \hat{\ell}_1 \times B_3^\top \hat{\ell}_3) = 0 \quad (8)$$

If we identify cameras 2 and 1 by setting $B_2 = B_1$, which corresponds to the case where both ℓ_1 and ℓ_2 are taken from the same camera. If these are different parallel lines, then we obtain the *vanishing point constraint*

Proposition 3 We have two or three parallel world lines, and two cameras with rotation/calibration matrices B_i . If camera 1 views image lines $\hat{\ell}_1$ and $\hat{\ell}_3$ and camera 2 views image line $\hat{\ell}_2$ we obtain **the vanishing point constraint**.

$$\hat{\ell}_2^\top B_2 B_1^{-1} (\hat{\ell}_1 \times \hat{\ell}_3) = 0 \quad (9)$$

$$\hat{\ell}_2^\top B_2 B_1^{-1} \hat{\mathbf{p}} = 0 \quad (10)$$

The quantity $\hat{\mathbf{p}} = \hat{\ell}_1 \times \hat{\ell}_3$ is called a vanishing point, and it is the point through which all images of world lines of direction \mathbf{L}_d will pass. The constraint says that if we have a vanishing point in one image and a line in another image which we know is parallel to the lines in the first camera, then we have a constraint on the B_i .

If we further identify cameras 2 and 1, then given an image of a set of parallel lines in one camera, we know that we must still have a zero triple product.

Proposition 4 We have three parallel world lines, and a camera with rotation/calibration nonlinear function $B : \mathbb{R}^3 \rightarrow \mathbb{R}^3$. Given images of these three world lines $\hat{\ell}_i$, $i \in [1, \dots, 3]$. We obtain **the vanishing point existence constraint**.

$$|B^\top \hat{\ell}_1 B^\top \hat{\ell}_2 B^\top \hat{\ell}_3| = 0 \quad (11)$$

This last constraint means nothing if B is a linear function, since the constraint would be trivially satisfied. However, in the case where there is some nonlinear distortion in the projection equation, there will be a constraint on B . We may say that the prismatic line constraint operates on 1, 2, or 3 cameras. We use the prismatic line constraint to obtain an initial estimate for the internal and rotational calibration parameters.

We use the prismatic line constraint to obtain an initial estimate of our functions B_i . We run a nonlinear optimization to find the best B_i for the data. Please note that it was not necessary for the cameras to share any parts of their field of view to obtain an initial estimate for these functions, if we can ensure that they view parallel lines. Having these functions B_i , we show how to obtain estimates for our camera positions \mathbf{T}_i .

Because we are using many cameras, we use a slightly different formulation of the epipolar constraint [7], designed so that we use the cameras' absolute position \mathbf{T}_i rather than the epipoles between the cameras.

$$(B_1^{-1} \hat{\mathbf{p}}_1 \times B_2^{-1} \hat{\mathbf{p}}_2)^\top \mathbf{T}_1 + (B_2^{-1} \hat{\mathbf{p}}_2 \times B_1^{-1} \hat{\mathbf{p}}_1)^\top \mathbf{T}_2 = 0 \quad (12)$$

Since we will have initial estimates for both B_1 and B_2 , we also note that by using calibrated image points:

$$(\mathbf{p}_1 \times \mathbf{p}_2)^\top \mathbf{T}_1 + (\mathbf{p}_2 \times \mathbf{p}_1)^\top \mathbf{T}_2 = 0 \quad (13)$$

For some camera networks we may not have a single point visible from multiple cameras, if the cameras do not share a field of view. In this case we may use a line to obtain initial translational estimates.

Proposition 5 *If we have three cameras with parameters (B_i, \mathbf{T}_i) , and a world line which projects to ℓ_i , then the prismatic line constraint holds. There is only one other independent constraint, and it is the **line trifocal constraint**:*

$$0 = \sum_{[i_1..i_3] \in \mathcal{P}^+[1..3]} (\mathbf{T}_{i_1})^T \ell_{i_1} | \ell_{i_2} \times \ell_{i_3} | \quad (14)$$

where $|\cdot|$ is the signed magnitude.

These constraints are sufficient to fix the cameras, but we need an efficient mechanism for combining the measurements into a coherent whole.

2.4 Putting the Constraints Together

Let us assume that we have M cameras and that we want to extend our constraints to find the \mathbf{T}_i considering all of the cameras. Note that our epipolar and trilinear constraints are of the form

$$\sum_{i=1}^n c_i \mathbf{m}_i^T \mathbf{T}_i = 0 \quad (15)$$

where n is 2 for the epipolar and 3 for the line trilinear, and \mathbf{m}_i is either a point \mathbf{p} or a line ℓ . Let us also assume that $n < M$. We may form an equation for every choice $\{k_1..k_n\}$ of n numbers from the set $\{1..M\}$. Each of our constraints is a linear equation over the $\mathbf{m}_{k_i}, \mathbf{T}_{k_i}$. We can form this into a linear equation:

$$\sum_{i=1}^n f_i(\mathbf{m}_{k_1}, \dots, \mathbf{m}_{k_n}) \mathbf{m}_{k_i}^T \mathbf{T}_{k_i} \quad (16)$$

Let us now set

$$\mathbf{T} = \begin{bmatrix} \mathbf{T}_1 \\ \vdots \\ \mathbf{T}_i \\ \vdots \\ \mathbf{T}_M \end{bmatrix} \quad \mathbf{T}' = \begin{bmatrix} \mathbf{T}_2 \\ \vdots \\ \mathbf{T}_i \\ \vdots \\ \mathbf{T}_M \end{bmatrix} \quad (17)$$

With each choice of n measurements from the M measurements, we obtain another constraint, so we get $\binom{M}{n}$ equations, which we put into matrix form:

$$\begin{bmatrix} \cdots & \vdots & \cdots \\ \cdots & f_i(\mathbf{m}_{k_1}, \dots, \mathbf{m}_{k_n}) \mathbf{m}_{k_i}^T & \cdots \\ \cdots & \vdots & \cdots \end{bmatrix} \mathbf{T} = 0 \quad (18)$$

where the terms \mathbf{m}_{k_i} are always in the k_i column. In the case of the epipolar equation, we will have two columns in each row and in the case of the trifocal equation, we will have three columns in each row.

Let us call this matrix on the LHS A . A has $\binom{M}{n}$ rows and n columns. Let us form the matrix $C = A^T A$. It is well known that the eigenvector corresponding to the smallest eigenvalue of C will be the unit vector which solves $AT = 0$. Because the problem is translation invariant, we have three extra degrees of freedom, so that our matrix C will have rank $M - 4$ in the general case. If we set $\mathbf{T}_1 = \mathbf{0}$, then we fix the location of the first camera to be at the origin of the fiducial coordinate system. If we form C' to be the bottom right $3(M - 1)$ square matrix of C , then we know that

$$(\mathbf{T}')^T C' \mathbf{T}' = 0 \quad (19)$$

That is, the zero eigenvector of C' gives the solution for our \mathbf{T}_i , if $\mathbf{T}_1 = \mathbf{0}$. We thus have a method for integrating either our epipolar or trifocal constraints over any number of cameras to obtain an initial translation translation. We only discuss the epipolar equation here.

2.5 Nonlinear Factors and Lines

Let us look at how to find radial calibration parameters with lines. Since we work with image measurements, we will parameterize our functions over B^{-1} . We do this because it is very difficult to invert the function $\mathbf{r} = B^{-1}(\hat{\mathbf{r}})\hat{\mathbf{r}}$. It is easy to invert the matrix function $B(\hat{\mathbf{r}})$, and we do use this inverse. Note that this is the reverse of the way that most authors have defined it, but when the object is to use the images in order to reconstruct, it makes just as much sense to parametrize the inverse of the projection than the forward direction.

In the linear case, this inverse mapping is a matrix B^{-1} which is the same over the whole image. In this case the function inverse is easy and corresponds to the matrix inverse. In the nonlinear case, we have a matrix function $B^{-1}(\mathbf{r})$, which varies depending on the location of \mathbf{r} . So to put an image point into the translated fiducial coordinate system, we use

$$\mathbf{r} = B^{-1}(\hat{\mathbf{r}})\hat{\mathbf{r}} \quad (20)$$

The function is defined in terms of points $\hat{\mathbf{r}}$. If we have a measurement on a line which we would like to put in that same coordinate system, we can use the same technique as before and form:

$$\ell = B^T(\hat{\mathbf{r}})\hat{\ell} \quad (21)$$

Note that this means that we need to know the location $\hat{\mathbf{r}}$ where we made our measurement of $\hat{\ell}$. Because the location of our measurement is crucial our results, if we are to calibrate it is important to make a measurement of $\hat{\ell}$ as locally as possible for maximum accuracy.

We consider in this section radial distortion, which in high quality lenses accounts for most of the distortion. We express our matrix function as

$$B^{-1}(\hat{\mathbf{r}}) = R^T S(K^{-1}\hat{\mathbf{r}})K^{-1} \quad (22)$$

where K is upper triangular, R is orthogonal, and

$$S(\hat{\mathbf{r}}) = I + (\hat{\mathbf{z}} \times (\hat{\mathbf{r}} \times \hat{\mathbf{z}}))(\rho_1 + \rho_2 |\hat{\mathbf{z}} \times \hat{\mathbf{r}}|^2 + \rho_3 |\hat{\mathbf{z}} \times \hat{\mathbf{r}}|^4) (\hat{\mathbf{z}} \times (\hat{\mathbf{r}} \times \hat{\mathbf{z}}))^T \quad (23)$$

This is the reverse of the usual method of describing radial calibration, and while the parameters are not the same, the result encodes just the same information. To map lines, then, we use the equation

$$(B^{-1}(\hat{\mathbf{r}}))^{-T} = B^T(\hat{\mathbf{r}}) \quad (24)$$

$$= R^T S^{-T}(K^{-1}\hat{\mathbf{r}})K^T \quad (25)$$

But since S is symmetric

$$(B^{-1}(\hat{\mathbf{r}}))^{-T} = R^T S^{-1}(K^{-1}\hat{\mathbf{r}})K^T \quad (26)$$

Next note that if we can invert a matrix $I + \mathbf{v}\mathbf{v}^T$ as

$$(I + \mathbf{v}\mathbf{v}^T)^{-1} = I - \frac{\mathbf{v}\mathbf{v}^T}{1 + \mathbf{v}^T\mathbf{v}} \quad (27)$$

$$= I - \mathbf{v}\mathbf{v}^T(1 - \mathbf{v}^T\mathbf{v} + (\mathbf{v}^T\mathbf{v})^2 - \dots) \quad (28)$$

From here on we set $\mathbf{r}' = K^{-1}\hat{\mathbf{r}}$

$$\mathbf{v} = (\hat{\mathbf{z}} \times (\mathbf{r}' \times \hat{\mathbf{z}})) \sqrt{\rho_1 + \rho_2 |\hat{\mathbf{z}} \times \mathbf{r}'|^2 + \rho_3 |\hat{\mathbf{z}} \times \mathbf{r}'|^4} \quad (29)$$

then we can see that

$$S(\mathbf{r}') = I + \mathbf{v}\mathbf{v}^T \quad (30)$$

so that

$$\begin{aligned} S^{-1}(\mathbf{r}') &= I - (\hat{\mathbf{z}} \times (\mathbf{r}' \times \hat{\mathbf{z}}))(\hat{\mathbf{z}} \times (\mathbf{r}' \times \hat{\mathbf{z}}))^T \\ &\quad (\rho_1 + \rho_2 |\hat{\mathbf{z}} \times \mathbf{r}'|^2 + \rho_3 |\hat{\mathbf{z}} \times \mathbf{r}'|^4 \\ &\quad - |\hat{\mathbf{z}} \times \mathbf{r}'|^2(\rho_1 + \rho_2 |\hat{\mathbf{z}} \times \mathbf{r}'|^2 + \rho_3 |\hat{\mathbf{z}} \times \mathbf{r}'|^4)^2 \\ &\quad + |\hat{\mathbf{z}} \times \mathbf{r}'|^4(\rho_1 + \rho_2 |\hat{\mathbf{z}} \times \mathbf{r}'|^2 + \rho_3 |\hat{\mathbf{z}} \times \mathbf{r}'|^4)^3 \dots) \end{aligned} \quad (31)$$

We throw away all terms attached to $|\hat{\mathbf{z}} \times \mathbf{r}'|^6$ and higher to obtain

$$\begin{aligned} S^{-1}(\mathbf{r}') &= I - (\hat{\mathbf{z}} \times (\mathbf{r}' \times \hat{\mathbf{z}}))(\rho_1 + (\rho_2 - \rho_1^2) |\hat{\mathbf{z}} \times \mathbf{r}'|^2 \\ &\quad + (\rho_3 - 2\rho_1\rho_2 + \rho_1^3) |\hat{\mathbf{z}} \times \mathbf{r}'|^4) (\hat{\mathbf{z}} \times (\mathbf{r}' \times \hat{\mathbf{z}}))^T \end{aligned} \quad (32)$$

We can have the tools to put the line $\hat{\ell}$ into the fiducial frame with radial distortion, using the parameters of radial distortion designed for points $\hat{\mathbf{r}}$. We use this theory to find an initial estimate for radial distortion by just using lines.

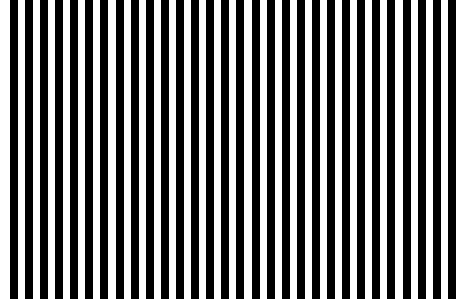


Figure 1: Parallel lines are on one side of the board

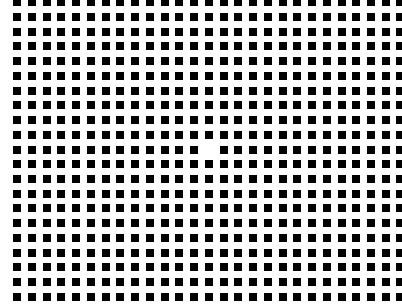


Figure 2: Boxes are on the other side of the board

3 Measurements on the Image

We now show how to use the constraints in a realistic engineering setting to solve the problem of multicamera calibration. While the geometry of multicamera calibration is mostly known, the image processing is just as important. We must make the signal processing easy to obtain stable initial estimates. We have only tested our solution in the case where cameras have highly overlapping fields of view, but we see no reason that nonoverlapping fields of view could not be calibrated with a common line.

3.1 Proposed New Solution

We can break up the problem into a two stage solution in order to obtain a stable solution. We work with textures printed on either side of a board. On one side of the board is printed a set of lines (as in figure 1). On the other side of the board is printed a set of boxes with one missing in the middle (as in figure 2).

In the first stage the set of lines is used to find the internal and rotational calibrations to within an affine distortion. Because we use the prismatic line constraint, we do not have to actually extract particular lines from the set of parallel lines. It is enough to extract only the line direction at different places in the image, as in figure 3. In each local area centered at \mathbf{r}_i there is a texture which can be approximated by set of lines with no foreshortening, if the area is small enough. We can easily find the orientation in this area. Us-

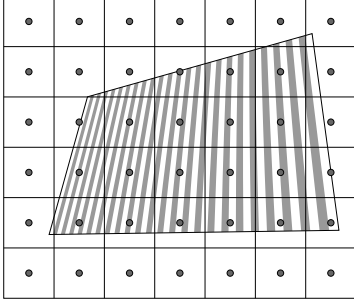


Figure 3: A projected board is measured locally

ing this orientation \mathbf{d}_i we can find the uncalibrated image lines as $\hat{\ell}_i = \mathbf{r}_i \times \mathbf{d}_i$. We then obtain a collection of these ℓ_i for each image.

Knowing the aspect ratio and skew (probably 1 and 0) allows us to find the radial distortion parameters using these line measurements and the vanishing point existence constraint. We use this method to obtain a reasonable initial estimate for the first radial parameter, which should contain most of the distortion [11]. We have found that this procedure overestimates the radial distortion somewhat, but it is good enough to give us starting conditions for a nonlinear minimization.

Once we do this we can collect our measurements into a matrix $\mathbf{L} = \sum_i \ell_i \ell_i^T$. The null space of this matrix, which is the uncalibrated point at infinity, we call \mathbf{s} .

To get an initial estimate for the internal and rotational calibration to within an affine distortion, we must fix some arbitrary affine distortion. We can do this by selecting four video frames to fix our space (3 to define our three cardinal directions, and one more frame to define dot products). The first three frames define the directions $\mathbf{e}_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$,

$\mathbf{e}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$, and $\mathbf{e}_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$. With these we can see easily

that the directions of the column of B will be $\mathbf{s}_1, \mathbf{s}_2,$ and \mathbf{s}_3 . If we assume that the fourth frame is $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$, then we can see that $\mathbf{s}_4 = \lambda_1 \mathbf{s}_1 + \lambda_2 \mathbf{s}_2 + \lambda_3 \mathbf{s}_3$, which will solve for the λ_i up to an entire factor, which is enough to fix B . This gives us our initial estimates for all cameras which have views of the fiducial four video frames. Once we have fixed this, we may then add camera by camera to our initial set of cameras, using a linear method followed by a nonlinear minimization. We have found this to be a stable method for finding our internal and rotational parameters to within an affine distortion.

On the other side of the board we have boxes which are aligned in the same direction as the lines. Since these boxes define orthogonal directions, we may extract these orthogonal directions and use them to find the proper affine distortion

to rectify our entire space [6]. We have after this stage an initial estimate for the internal, nonlinear, and rotational parameters for all the cameras.

At this point one may ask why we do not simply use the boxes themselves to obtain an initial estimate for the rotation, and internal calibration. The answer is that the correspondence problem militates against this. We wouldn't know which orientation would correspond to which in the images. We need to use the parallel lines first to get an initial orientation for all the cameras. Having that orientation, we can then use the orthogonal sets of parallel lines.

3.2 Finding the Translational Parameters

We mentioned at the beginning of this paper that precisely measuring features is an important component the calibration method. Unfortunately, in a multicamera system it can be difficult to have consistent features because if the texture is out of the depth of field there will be significant blurring. The scale differences can cause any features to be quite small with respect to the cameras. We therefore take our feature to be the entire pattern of boxes.

Remember that at this point in the calibration, we have a good estimate for the internal and rotational parameters. We also know the principal directions of the box texture. With this information, it is an easy matter to unwarp the box texture. This unwarped pattern will be a frontoparallel set of boxes with one box missing in the middle. If a camera sees the boxes but the missing box is out of the field of view then that frame for that camera is not used. In any case, given a sufficient number of views of the box texture from each camera, we can extract corresponding points with pixel accuracy. Using these correspondences and the matrix eigenspace technique showed earlier, we can find estimates for the positions of all the cameras.

3.3 Final Optimization

At this point we have initial estimates for the camera parameters, and an estimate for the normal and position of the texture. The only point measurement we have made is the one for the missing box. In a single camera calibration system, it would be a simple matter to extract points from the boxes. However, due to the great variability in photometric properties, blurred images, and foreshortened images, we found we needed to hand-check each image to make sure that the points were extracted properly. This was unacceptable. We therefore decided to optimize over the images themselves. While this is computationally expensive, we found that it gave us high accuracy with little user effort.

We can describe the position of a box at frame i with a point \mathbf{P}_i at the missing box, and two perpendicular unit

vectors $\mathbf{v}_{i,1}$, and $\mathbf{v}_{i,2}$ which represent the displacement between contiguous boxes in each direction. The cameras j are described with the function B_j and position \mathbf{T}_j . We desire to minimize over the sum of all the pixel values which we expect would be inside of a box.

Let us determine which pixels should be projections of the black box. We have a pixel \mathbf{r} in a camera described by (B, \mathbf{T}) , at a frame where the box texture is described by $(\mathbf{P}, \mathbf{v}_1, \mathbf{v}_2)$. If we consider the \mathbf{v}_1 and \mathbf{v}_2 as defining a coordinate system on the texture, then the coordinates of where the ray \mathbf{r} intersects the texture are

$$x = \frac{|B^{-1}\mathbf{r} \mathbf{v}_1 \mathbf{P}| + |B^{-1}\mathbf{r} \mathbf{T} \mathbf{v}_1|}{|B^{-1}\mathbf{r} \mathbf{v}_1 \mathbf{v}_2|} \quad (33)$$

$$y = \frac{|B^{-1}\mathbf{r} \mathbf{v}_2 \mathbf{P}| + |B^{-1}\mathbf{r} \mathbf{T} \mathbf{v}_2|}{|B^{-1}\mathbf{r} \mathbf{v}_1 \mathbf{v}_2|} \quad (34)$$

Thus our pixel is in the box if x and y are both within $\frac{1}{4}$ of an integer, and also x and y are within the boundaries of the number of boxes contained on the board. It may be the case that for many frames of a camera, there are no boxes. For some frames, there may be partial showing of a board. For some frames the full board may be shown. Additionally, we cannot just add up the pixels we think will be in the box, since that will cause discontinuous jumps in our error metric. We therefore multiply the pixel value by the distance of the ray from the center of the box, which we call $d(\mathbf{r})$

In any case, we define our error measure for a camera j and frame i as:

$$E_{i,j} = \sum_{\mathbf{r} \in \text{box pixels}} \text{pixel value}(\mathbf{r})d(\mathbf{r}) \quad (35)$$

We may then minimize

$$\sum_{i,j} E_{i,j} \quad (36)$$

over the parameters for the cameras and the textured planes in a bundle adjustment over texture reprojection.

Due to computational considerations, it was not possible to minimize all the parameters at once. Instead, we alternately optimized for the parameters of each texture and then for the parameters of each camera. Since the cameras are only loosely coupled together through the textures, this alternating minimization worked well to give us good results.

4 Results

We have tested this algorithm on a camera network with 33 cameras pointing towards a roughly 1 cubic foot volume. These cameras all had high quality lenses with minimal radial distortion. 9 of the cameras were black and white with

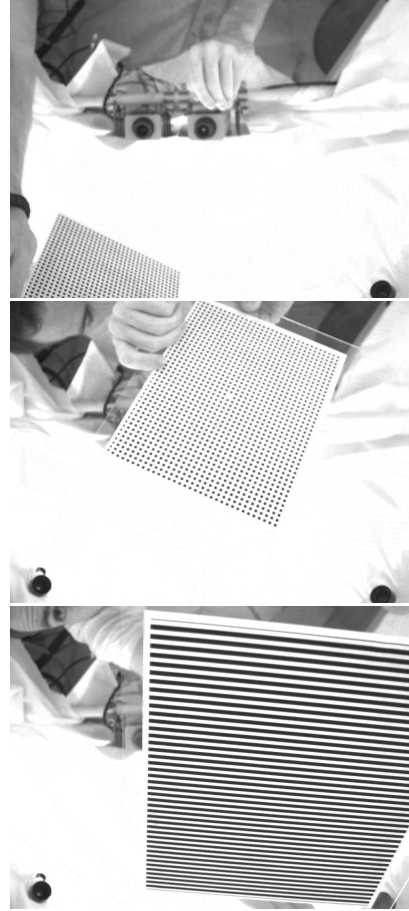


Figure 4: Texture Images from Camera Network

1024 × 768 pixels, 12 of the cameras were black and white with 640 × 480 pixels, and 12 of the cameras were color with 640 × 480 pixels.

We showed a board which had been printed on one side with lines and on the other side with boxes. We used no special technique besides attempting to fill the viewing volume in various orientations. We did take care in the beginning to show the parallel lines in the three cardinal directions and one more, so that we could fix the affine distortion in our search for the B matrices. We took 300 frames of the board. We show a few representative pictures in figure 4.

The most important result is that we obtained calibration with no human intervention (in $O(1)$ user time). Since the optimization ran over the sum of pixel values, we cannot give numerical results in terms of reprojection error. The individual boxes were so small that it was not feasible to run a corner detector. However, there are a few options to show why we believe that our calibration is good.

1. We can project our hypothesized locations for the box texture and see how closely they match with the actual

projections of the boxes.

2. In terms of camera parameters, we did not enforce the skewless constraint, so we can give statistics on this parameter. We also know that our camera had an aspect ratio of one, so that the statistics on the ratio of the x and y focal lengths should also give an indication of the quality of the calibration.
3. The best indication of quality would be an accurate 3D model created with this calibration, but unfortunately this work is not yet completed.

With respect to the first measure, we have looked at the reprojections of our box texture and found that the reprojections of the boxes were at most about a pixel off from the real images. We were hoping for better results, but it is possible that our two parameter radial calibration model for nonlinear parameters may not be sufficient. This is, however, significantly better than we have ever achieved by any other calibration method for our multicamera laboratory.

The statistics of the camera parameters are shown in the following table. The first column shows statistics on the skew divided by the focal length. The second column shows the statistics absolute value of the difference of the aspect ratio from one.

	$\text{abs}(\frac{\text{skew}}{\text{focallength}})$	$\text{abs}(1 - \text{aspectratio})$
mean	0.00167	0.00228
median	0.00102	0.00106
max	0.00852	0.01526

The median gives much better results than the mean because there were a few cameras which were not pointed quite to the center of the viewing volume. For these cameras, large parts of their field of view had virtually no data, so that the parameters should be expected to be off.

5 Discussion

The calibration of many cameras needs to be extremely robust, since we are not able to check the output from individual cameras. We have shown how to use parallel lines to obtain robust initial estimates for the nonlinear distortion and internal calibration. We have also shown a method to get initial translational estimates by using the missing box from a texture. Important to the method, however, is also being able to find initial estimates for the location of our textured plane. Using all of these initial estimates, we are then able to run a nonlinear minimization which converges on a solution accurate to at most pixel error. We expect that with a better nonlinear model we could improve the error to subpixel accuracy.

We calibrated a camera network which had highly overlapping fields of view. Although it has not been tested as yet, we anticipate that cameras with non-overlapping fields

of view could be calibrated using the prismatic and line trifocal constraints, together with a box texture for a final nonlinear optimization.

References

- [1] P. Baker and Y. Aloimonos. Complete calibration of a multi-camera network. In *Proc. IEEE Workshop on Omnidirectional Vision*, pages 134–141, Hilton Head Island, SC, 2000. IEEE Computer Society.
- [2] D. Brown. Close-range camera calibration. *Photogrammetric Engineering*, 37(8):855–866, 1971.
- [3] O. Faugeras and G. Toscani. The calibration problem for stereo. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 364–374, Miami Beach, FL, 1986.
- [4] O. D. Faugeras. Stratification of 3-D vision: Projective, affine, and metric representations. *Journal of the Optical Society of America A*, 12:465–484, 1995.
- [5] J. Heikkilä and O. Silvén. A four-step camera calibration procedure with implicit image correction. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, pages 1106–1112, 1997.
- [6] D. Liebowitz and A. Zisserman. Metric rectification for perspective images of planes. In *Proceedings IEEE Conference on CVPR*, pages 261–289, June 1998.
- [7] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.
- [8] C. C. Slama, C. Theurer, and S. W. Henriksen. *Manual of Photogrammetry*. American Society of Photogrammetry, Falls Church, VA, 1980.
- [9] G. P. Stein. Lens distortion calibration using point correspondence. Technical report, Massachusetts Institute of Technology, 1996.
- [10] P. Sturm and S. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *Proc of CVPR*, 1999.
- [11] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proc International Conference on Computer Vision*, volume 1, pages 666–673, 1999.