**Z. Zhang, "Camera Calibration", Chapter 2, pages 4-43,
in G. Medioni and S.B. Kang, eds., *Emerging Topics in Computer Vision*,
Prentice Hall Professional Technical Reference, 2004.**

Chapter 1

CAMERA CALIBRATION

Camera calibration is a necessary step in 3D computer vision in order to extract metric information from 2D images. It has been studied extensively in computer vision and photogrammetry, and even recently new techniques have been proposed. In this chapter, we review the techniques proposed in the literature include those using 3D apparatus (two or three planes orthogonal to each other, or a plane undergoing a pure translation, etc.), 2D objects (planar patterns undergoing unknown motions), 1D objects (wand with dots) and unknown scene points in the environment (self-calibration). The focus is on presenting these techniques within a consistent framework.

1.1 Introduction

Camera calibration is a necessary step in 3D computer vision in order to extract metric information from 2D images. Much work has been done, starting in the photogrammetry community (see [3, 6] to cite a few), and more recently in computer vision ([12, 11, 33, 10, 37, 35, 22, 9] to cite a few). According to the dimension of the calibration objects, we can classify those techniques roughly into three categories.

3D reference object based calibration. Camera calibration is performed by observing a calibration object whose geometry in 3-D space is known with very good precision. Calibration can be done very efficiently [8]. The calibration object usually consists of two or three planes orthogonal to each other. Sometimes, a plane undergoing a precisely known translation is also used [33], which equivalently provides 3D reference points. This approach requires an expensive calibration apparatus and an elaborate setup.

2D plane based calibration. Techniques in this category requires to observe a planar pattern shown at a few different orientations [42, 31]. Different from Tsai's technique [33], the knowledge of the plane motion is not necessary. Because almost anyone can make such a calibration pattern by him/her-self, the setup is easier for camera calibration.

1D line based calibration. Calibration objects used in this category are composed of a set of collinear points [44]. As will be shown, a camera can be

calibrated by observing a moving line around a fixed point, such as a string of balls hanging from the ceiling.

Self-calibration. Techniques in this category do not use any calibration object, and can be considered as 0D approach because only image point correspondences are required. Just by moving a camera in a static scene, the rigidity of the scene provides in general two constraints [22, 21] on the cameras' internal parameters from one camera displacement by using image information alone. Therefore, if images are taken by the same camera with fixed internal parameters, correspondences between three images are sufficient to recover both the internal and external parameters which allow us to reconstruct 3-D structure up to a similarity [20, 17]. Although no calibration objects are necessary, a large number of parameters need to be estimated, resulting in a much harder mathematical problem.

Other techniques exist: vanishing points for orthogonal directions [4, 19], and calibration from pure rotation [16, 30].

Before going further, I'd like to point out that no single calibration technique is the best for all. It really depends on the situation a user needs to deal with. Following are my few recommendations:

- Calibration with apparatus vs. self-calibration. Whenever possible, if we can pre-calibrate a camera, we should do it with a calibration apparatus. Self-calibration cannot usually achieve an accuracy comparable with that of pre-calibration because self-calibration needs to estimate a large number of parameters, resulting in a much harder mathematical problem. When pre-calibration is impossible (e.g., scene reconstruction from an old movie), self-calibration is the only choice.

- Partial vs. full self-calibration. Partial self-calibration refers to the case where only a subset of camera intrinsic parameters are to be calibrated. Along the same line as the previous recommendation, whenever possible, partial self-calibration is preferred because the number of parameters to be estimated is smaller. Take an example of 3D reconstruction with a camera with variable focal length. It is preferable to pre-calibrate the pixel aspect ratio and the pixel skewness.

- Calibration with 3D vs. 2D apparatus. Highest accuracy can usually be obtained by using a 3D apparatus, so it should be used when accuracy is indispensable and when it is affordable to make and use a 3D apparatus. From the feedback I received from computer vision researchers and practitioners around the world in the last couple of years, calibration with a 2D apparatus seems to be the best choice in most situations because of its ease of use and good accuracy.

- Calibration with 1D apparatus. This technique is relatively new, and it is hard for the moment to predict how popular it will be. It, however, should be

useful especially for calibration of a camera network. To calibrate the relative geometry between multiple cameras as well as their intrinsic parameters, it is necessary for all involving cameras to simultaneously observe a number of points. It is hardly possible to achieve this with 3D or 2D calibration apparatus[1] if one camera is mounted in the front of a room while another in the back. This is not a problem for 1D objects. We can for example use a string of balls hanging from the ceiling.

This chapter is organized as follows. Section 1.2 describes the camera model and introduces the concept of the absolute conic which is important for camera calibration. Section 1.3 presents the calibration techniques using a 3D apparatus. Section 1.4 describes a calibration technique by observing a freely moving planar pattern (2D object). Its extension for stereo calibration is also addressed. Section 1.5 describes a relatively new technique which uses a set of collinear points (1D object). Section 1.6 briefly introduces the self-calibration approach and provides references for further reading. Section 1.7 concludes the chapter with a discussion on recent work in this area.

1.2 Notation and Problem Statement

We start with the notation used in this chapter.

1.2.1 Pinhole Camera Model

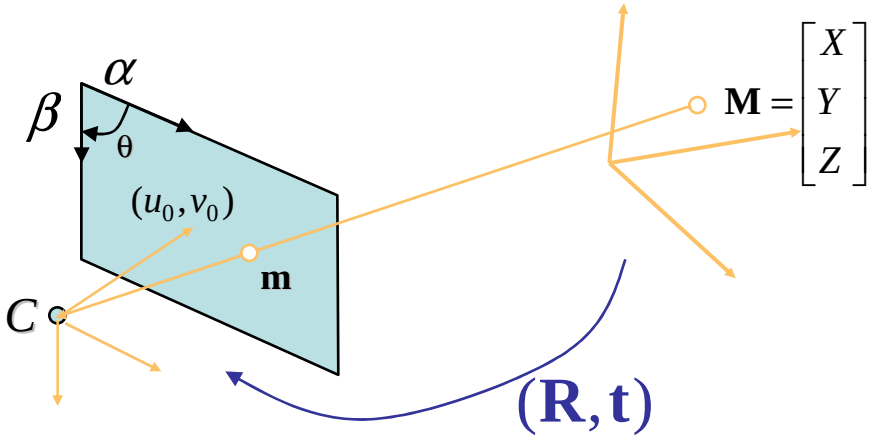


Figure 1.1. Pinhole camera model

A 2D point is denoted by $\mathbf{m} = [u, v]^T$. A 3D point is denoted by $\mathtt{M} = [X, Y, Z]^T$.

[1] An exception is when those apparatus are made transparent; then the cost would be much higher.

We use $\widetilde{\mathbf{x}}$ to denote the augmented vector by adding 1 as the last element: $\widetilde{\mathbf{m}} = [u, v, 1]^T$ and $\widetilde{\mathtt{M}} = [X, Y, Z, 1]^T$. A camera is modeled by the usual pinhole (see Figure 1.1): The image of a 3D point $\mathtt{M}$, denoted by $\mathbf{m}$ is formed by an optical ray from $\mathtt{M}$ passing through the optical center C and intersecting the image plane. The three points $\mathtt{M}$, $\mathbf{m}$, and C are collinear. In Figure 1.1, for illustration purpose, the image plane is positioned between the scene point and the optical center, which is mathematically equivalent to the physical setup under which the image plane is in the other side with respect to the optical center. The relationship between the 3D point $\mathtt{M}$ and its image projection $\mathbf{m}$ is given by

$$s\widetilde{\mathbf{m}} = \underbrace{\mathbf{A}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix}}_{\mathbf{P}}\widetilde{\mathtt{M}} \equiv \mathbf{P}\widetilde{\mathtt{M}}\,, \tag{1.2.1}$$

$$\text{with } \mathbf{A} = \begin{bmatrix}\alpha & \gamma & u_0\\ 0 & \beta & v_0\\ 0 & 0 & 1\end{bmatrix} \tag{1.2.2}$$

$$\text{and} \quad \mathbf{P} = \mathbf{A}\begin{bmatrix}\mathbf{R} & \mathbf{t}\end{bmatrix} \tag{1.2.3}$$

where s is an arbitrary scale factor, $(\mathbf{R}, \mathbf{t})$, called the extrinsic parameters, is the rotation and translation which relates the world coordinate system to the camera coordinate system, and $\mathbf{A}$ is called the camera intrinsic matrix, with (u_0, v_0) the coordinates of the principal point, α and β the scale factors in image u and v axes, and γ the parameter describing the skew of the two image axes. The 3×4 matrix $\mathbf{P}$ is called the camera projection matrix, which mixes both intrinsic and extrinsic parameters. In Figure 1.1, the angle between the two image axes is denoted by θ, and we have $\gamma = \alpha \cot\theta$. If the pixels are rectangular, then $\theta = 90^\circ$ and $\gamma = 0$.

The task of camera calibration is to determine the parameters of the transformation between an object in 3D space and the 2D image observed by the camera from visual information (images). The transformation includes

- Extrinsic parameters (sometimes called external parameters): orientation (rotation) and location (translation) of the camera, i.e., $(\mathbf{R}, \mathbf{t})$;
- Intrinsic parameters (sometimes called internal parameters): characteristics of the camera, i.e., $(\alpha, \beta, \gamma, u_0, v_0)$.

The rotation matrix, although consisting of 9 elements, only has 3 degrees of freedom. The translation vector $\mathbf{t}$ obviously has 3 parameters. Therefore, there are 6 extrinsic parameters and 5 intrinsic parameters, leading to in total 11 parameters.

We use the abbreviation $\mathbf{A}^{-T}$ for $(\mathbf{A}^{-1})^T$ or $(\mathbf{A}^T)^{-1}$.

1.2.2 Absolute Conic

Now let us introduce the concept of the absolute conic. For more details, the reader is referred to [7, 15].

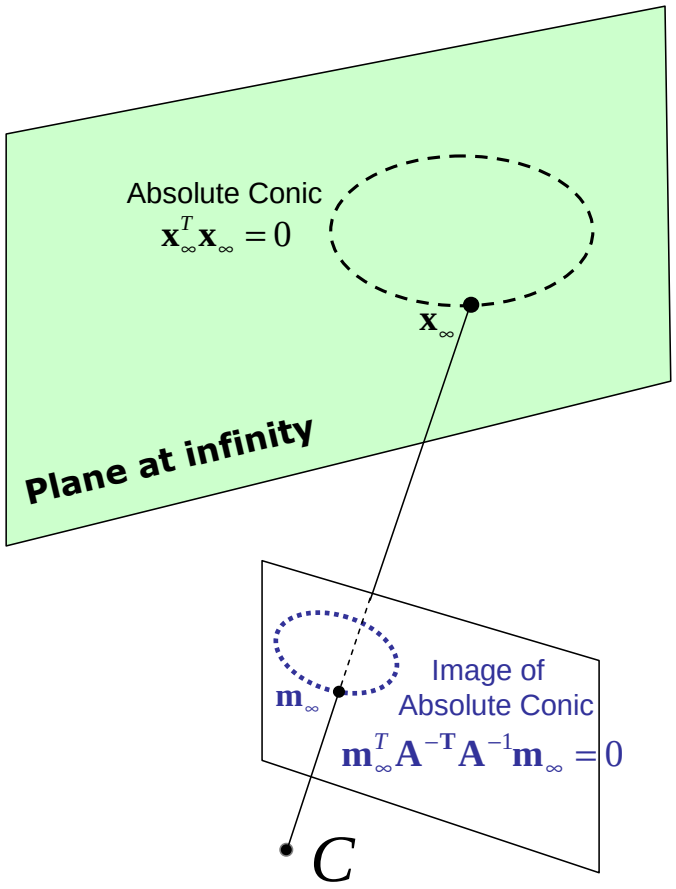


Figure 1.2. Absolute conic and its image

A point $\mathbf{x}$ in 3D space has projective coordinates $\widetilde{\mathbf{x}} = [x_1, x_2, x_3, x_4]^T$. The equation of the plane at infinity, Π_∞, is $x_4 = 0$. The absolute conic Ω is defined by a set of points satisfying the equation

$$\begin{aligned} x_1^2 + x_2^2 + x_3^2 &= 0 \\ x_4 &= 0\,. \end{aligned} \tag{1.2.4}$$

Let $\mathbf{x}_\infty = [x_1, x_2, x_3]^T$ be a point on the absolute conic (see Figure 1.2). By definition, we have $\mathbf{x}_\infty^T \mathbf{x}_\infty = 0$. We also have $\widetilde{\mathbf{x}}_\infty = [x_1, x_2, x_3, 0]^T$ and $\widetilde{\mathbf{x}}_\infty^T \widetilde{\mathbf{x}}_\infty = 0$. This can be interpreted as a conic of purely imaginary points on Π_∞. Indeed, let $x = x_1/x_3$ and $y = x_2/x_3$ be a point on the conic, then $x^2 + y^2 = -1$, which is an imaginary circle of radius $\sqrt{-1}$.

An important property of the absolute conic is its invariance to any rigid trans-

formation. Let the rigid transformation be $\mathbf{H} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix}$. Let $\mathbf{x}_\infty$ be a point on Ω. By definition, its projective coordinates: $\widetilde{\mathbf{x}}_\infty = \begin{bmatrix} \mathbf{x}_\infty \\ 0 \end{bmatrix}$ with $\mathbf{x}_\infty^T \mathbf{x}_\infty = 0$. The point after the rigid transformation is denoted by $\mathbf{x}'_\infty$, and

$$\widetilde{\mathbf{x}}'_\infty = \mathbf{H}\widetilde{\mathbf{x}}_\infty = \begin{bmatrix} \mathbf{R}\mathbf{x}_\infty \\ 0 \end{bmatrix} .$$

Thus, $\mathbf{x}'_\infty$ is also on the plane at infinity. Furthermore, $\mathbf{x}'_\infty$ is *on the same* Ω because

$$\mathbf{x}'^T_\infty \mathbf{x}'_\infty = (\mathbf{R}\mathbf{x}_\infty)^T (\mathbf{R}\mathbf{x}_\infty) = \mathbf{x}_\infty^T (\mathbf{R}^T \mathbf{R}) \mathbf{x}_\infty = 0 .$$

The image of the absolute conic, denoted by ω, is also an imaginary conic, and is determined only by the intrinsic parameters of the camera. This can be seen as follows. Consider the projection of a point $\mathbf{x}_\infty$ on Ω, denoted by $\mathbf{m}_\infty$, which is given by

$$\widetilde{\mathbf{m}}_\infty = s\mathbf{A}[\mathbf{R} \quad \mathbf{t}] \begin{bmatrix} \mathbf{x}_\infty \\ 0 \end{bmatrix} = s\mathbf{A}\mathbf{R}\mathbf{x}_\infty .$$

It follows that

$$\widetilde{\mathbf{m}}^T \mathbf{A}^{-T} \mathbf{A}^{-1} \widetilde{\mathbf{m}} = s^2 \mathbf{x}_\infty^T \mathbf{R}^T \mathbf{R} \mathbf{x}_\infty = s^2 \mathbf{x}_\infty^T \mathbf{x}_\infty = 0 .$$

Therefore, the image of the absolute conic is an imaginary conic, and is defined by $\mathbf{A}^{-T}\mathbf{A}^{-1}$. It does not depend on the extrinsic parameters of the camera.

If we can determine the image of the absolute conic, then we can solve the camera's intrinsic parameters, and the calibration is solved.

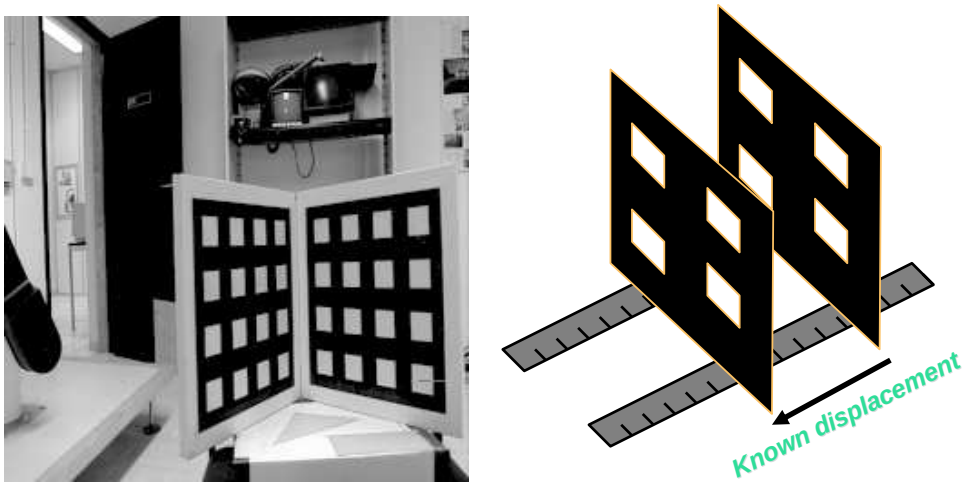
We will show several ways in this chapter how to determine ω, the image of the absolute conic.

1.3 Camera Calibration with 3D Objects

The traditional way to calibrate a camera is to use a 3D reference object such as those shown in Figure 1.3. In Fig. 1.3a, the calibration apparatus used at INRIA [8] is shown, which consists of two orthogonal planes, on each a checker pattern is printed. A 3D coordinate system is attached to this apparatus, and the coordinates of the checker corners are known very accurately in this coordinate system. A similar calibration apparatus is a cube with a checker patterns painted in each face, so in general three faces will be visible to the camera. Figure 1.3b illustrates the device used in Tsai's technique [33], which only uses one plane with checker pattern, but the plane needs to be displaced at least once with known motion. This is equivalent to knowing the 3D coordinates of the checker corners.

A popular technique in this category consists of four steps [8]:

1. Detect the corners of the checker pattern in each image;



(a) (b)

Figure 1.3. 3D apparatus for calibrating cameras

2. Estimate the camera projection matrix $\mathbf{P}$ using linear least squares;
3. Recover intrinsic and extrinsic parameters $\mathbf{A}$, $\mathbf{R}$ and $\mathbf{t}$ from $\mathbf{P}$;
4. Refine $\mathbf{A}$, $\mathbf{R}$ and $\mathbf{t}$ through a nonlinear optimization.

Note that it is also possible to first refine $\mathbf{P}$ through a nonlinear optimization, and then determine $\mathbf{A}$, $\mathbf{R}$ and $\mathbf{t}$ from the refined $\mathbf{P}$.

It is worth noting that using corners is not the only possibility. We can avoid corner detection by working directly in the image. In [25], calibration is realized by maximizing the gradients around a set of control points that define the calibration object. Figure 1.4 illustrates the control points used in that work.

1.3.1 Feature Extraction

If one uses a generic corner detector, such as Harris corner detector, to detect the corners in the check pattern image, the result is usually not good because the detector corners have poor accuracy (about one pixel). A better solution is to leverage the known pattern structure by first estimating a line for each side of the square and then computing the corners by intersecting the fitted lines. There are two common techniques to estimate the lines. The first is to first detect edges, and then fit a line to the edges on each side of the square. The second technique is to directly fit a line to each side of a square in the image such that the gradient on the line is maximized. One possibility is to represent the line by an elongated Gaussian, and estimate the parameters of the elongated Gaussian by maximizing the total gradient covered by the Gaussian. We should note that if the lens distortion is not

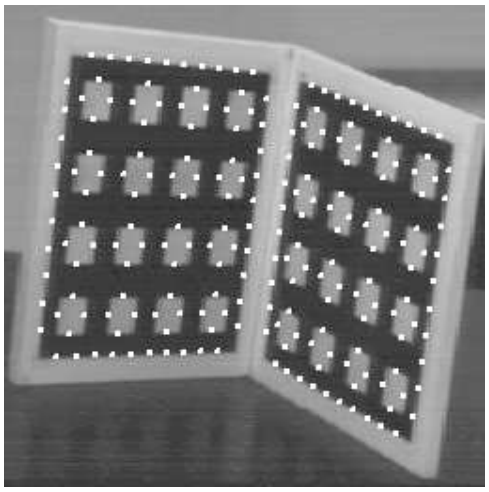


Figure 1.4. Control points used in a gradient-based calibration technique

severe, a better solution is to fit just one single line to all the collinear sides. This will leads a much more accurate estimation of the position of the checker corners.

1.3.2 Linear Estimation of the Camera Projection Matrix

Once we extract the corner points in the image, we can easily establish their correspondences with the points in the 3D space because of knowledge of the patterns. Based on the projection equation (1.2.1), we are now able to estimate the camera parameters. However, the problem is quite nonlinear if we try to estimate directly $\mathbf{A}$, $\mathbf{R}$ and $\mathbf{t}$. If, on the other hand, we estimate the camera projection matrix $\mathbf{P}$, a linear solution is possible, as to be shown now.

Given each 2D-3D correspondence $\mathbf{m}_i = (u_i, v_i) \leftrightarrow \mathtt{M}_i = (X_i, Y_i, Z_i)$, we can write down 2 equations based on (1.2.1):

$$\underbrace{\begin{bmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & u_iX_i & u_iY_i & u_iZ_i & u_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & v_iX_i & v_iY_i & v_iZ_i & v_i \end{bmatrix}}_{\mathbf{G}_i} \mathbf{p} = \mathbf{0}$$

where $\mathbf{p} = [p_{11}, p_{12}, \ldots, p_{34}]^T$ and $\mathbf{0} = [0, 0]^T$.

For n point matches, we can stack all equations together:

$$\mathbf{G}\mathbf{p} = \mathbf{0} \qquad \text{with } \mathbf{G} = [\mathbf{G}_1^T, \ldots, \mathbf{G}_n^T]^T$$

Matrix $\mathbf{G}$ is a $2n \times 12$ matrix. The projection matrix can now be solved by

$$\min_{\mathbf{p}} \|\mathbf{G}\mathbf{p}\|^2 \quad \text{subject to } \|\mathbf{p}\| = 1$$

The solution is the eigenvector of $\mathbf{G}^T\mathbf{G}$ associated with the smallest eigenvalue.

In the above, in order to avoid the trivial solution $\mathbf{p} = \mathbf{0}$ and considering the fact that $\mathbf{p}$ is defined up to a scale factor, we have set $\|\mathbf{p}\| = 1$. Other normalizations are possible. In [1], $p_{34} = 1$, which, however, introduce a singularity when the correct value of p_{34} is close to zero. In [10], the constraint $p_{31}^2 + p_{32}^2 + p_{33}^2 = 1$ was used, which is singularity free.

Anyway, the above linear technique minimizes an algebraic distance, and yields a *biased* estimation when data are noisy. We will present later an unbiased solution.

1.3.3 Recover Intrinsic and Extrinsic Parameters from P

Once the camera projection matrix $\mathbf{P}$ is known, we can uniquely recover the intrinsic and extrinsic parameters of the camera. Let us denote the first 3×3 submatrix of $\mathbf{P}$ by $\mathbf{B}$ and the last column of $\mathbf{P}$ by $\mathbf{b}$, i.e., $\mathbf{P} \equiv [\mathbf{B} \quad \mathbf{b}]$. Since $\mathbf{P} = \mathbf{A}[\mathbf{R} \quad \mathbf{t}]$, we have

$$\mathbf{B} = \mathbf{AR} \tag{1.3.1}$$

$$\mathbf{b} = \mathbf{At} \tag{1.3.2}$$

From (1.3.1), we have

$$\mathbf{K} \equiv \mathbf{BB}^T = \mathbf{AA}^T = \begin{bmatrix} \underbrace{\alpha^2 + \gamma^2 + u_0^2}_{k_u} & \underbrace{u_0\, v_0 + c\, \beta}_{k_c} & u_0 \\ \underbrace{u_0\, v_0 + c\, \alpha}_{k_c} & \underbrace{\alpha_v^2 + v_0^2}_{k_v} & v_0 \\ u_0 & v_0 & 1 \end{bmatrix}$$

Because $\mathbf{P}$ is defined up to a scale factor, the last element of $\mathbf{K} = \mathbf{BB}^T$ is usually not equal to 1, so we have to *normalize* it such that $\mathbf{K}_{33}$(the last element) $= 1$. After that, we immediately obtain

$$u_0 = \mathbf{K}_{13} \tag{1.3.3}$$

$$v_0 = \mathbf{K}_{23} \tag{1.3.4}$$

$$\beta = \sqrt{k_v - v_0^2} \tag{1.3.5}$$

$$\gamma = \frac{k_c - u_0\, v_0}{\beta} \tag{1.3.6}$$

$$\alpha = \sqrt{k_u - u_0^2 - \gamma^2} \tag{1.3.7}$$

The solution is unambiguous because: $\alpha > 0$ and $\beta > 0$.

Once the intrinsic parameters, or equivalently matrix $\mathbf{A}$, are known, the extrinsic parameters can be determined from (1.3.1) and (1.3.2) as:

$$\mathbf{R} = \mathbf{A}^{-1}\mathbf{B} \tag{1.3.8}$$

$$\mathbf{t} = \mathbf{A}^{-1}\mathbf{b}\,. \tag{1.3.9}$$

1.3.4 Refine Calibration Parameters Through a Nonlinear Optimization

The above solution is obtained through minimizing an algebraic distance which is not physically meaningful. We can refine it through maximum likelihood inference.

We are given n 2D-3D correspondences $\mathbf{m}_i = (u_i, v_i) \leftrightarrow \mathtt{M}_i = (X_i, Y_i, Z_i)$. Assume that the image points are corrupted by independent and identically distributed noise. The maximum likelihood estimate can be obtained by minimizing the distances between the image points and their predicted positions, i.e.,

$$\min_{\mathbf{P}} \sum_i \|\mathbf{m}_i - \boldsymbol{\phi}(\mathbf{P}, \mathtt{M}_i)\|^2 \tag{1.3.10}$$

where $\boldsymbol{\phi}(\mathbf{P}, \mathtt{M}_i)$ is the projection of $\mathtt{M}_i$ onto the image according to (1.2.1).

This is a nonlinear minimization problem, which can be solved with the Levenberg-Marquardt Algorithm as implemented in `Minpack` [23]. It requires an initial guess of $\mathbf{P}$ which can be obtained using the linear technique described earlier. Note that since $\mathbf{P}$ is defined up to a scale factor, we can set the element having the largest initial value as 1 during the minimization.

Alternatively, instead of estimating $\mathbf{P}$ as in (1.3.10), we can directly estimate the intrinsic and extrinsic parameters, $\mathbf{A}$, $\mathbf{R}$, and $\mathbf{t}$, using the same criterion. The rotation matrix can be parameterized with three variables such as Euler angles or scaled rotation vector.

1.3.5 Lens Distortion

Up to this point, we use the pinhole model to describe a camera. It says that the point in 3D space, its corresponding point in image and the camera's optical center are collinear. This linear projective equation is sometimes not sufficient, especially for low-end cameras (such as WebCams) or wide-angle cameras; lens distortion has to be considered.

According to [33], there are four steps in camera projection including lens distortion:

Step 1: *Rigid transformation* from world coordinate system (X_w, Y_w, Z_w) to camera one (X, Y, Z):

$$[X \quad Y \quad Z]^T = \mathbf{R}\,[X_w \quad Y_w \quad Z_w]^T + \mathbf{t}$$

Step 2: *Perspective projection* from 3D camera coordinates (X, Y, Z) to *ideal* image coordinates (x, y) under pinhole camera model:

$$x = f\frac{X}{Z}\,, \qquad y = f\frac{Y}{Z}$$

where f is the effective focal length.

Step 3: *Lens distortion*[2]:

$$\breve{x} = x + \delta_x \ , \qquad \breve{y} = y + \delta_y$$

where $(\breve{x}, \breve{y})$ are the *distorted* or *true* image coordinates, and (δ_x, δ_y) are distortions applied to (x, y).

Step 4: *Affine transformation* from real image coordinates $(\breve{x}, \breve{y})$ to *frame buffer* (pixel) image coordinates $(\breve{u}, \breve{v})$:

$$\breve{u} = d_x^{-1}\breve{x} + u_0 \ , \qquad \breve{v} = d_y^{-1}\breve{y} + v_0 \ ,$$

where (u_0, v_0) are coordinates of the principal point; d_x and d_y are distances between adjacent pixels in the horizontal and vertical directions, respectively.

There are two types of distortions:

Radial distortion: It is symmetric; ideal image points are distorted along radial directions from the distortion center. This is caused by imperfect lens shape.

Decentering distortion: This is usually caused by improper lens assembly; ideal image points are distorted in both radial and tangential directions.

The reader is referred to [29, 3, 6, 37] for more details.

The distortion can be expressed as power series in radial distance $r = \sqrt{x^2 + y^2}$:

$$\delta_x = x(k_1 r^2 + k_2 r^4 + k_3 r^6 + \cdots) + [p_1(r^2 + 2x^2) + 2p_2 xy](1 + p_3 r^2 + \cdots) \ ,$$
$$\delta_y = y(k_1 r^2 + k_2 r^4 + k_3 r^6 + \cdots) + [2p_1 xy + p_2(r^2 + 2y^2)](1 + p_3 r^2 + \cdots) \ ,$$

where k_i's are coefficients of radial distortion and p_j's and coefficients of decentering distortion.

Based on the reports in the literature [3, 33, 36], it is likely that the distortion function is totally dominated by the radial components, and especially dominated by the first term. It has also been found that any more elaborated modeling not only would not help (negligible when compared with sensor quantization), but also would cause numerical instability [33, 36].

Denote the ideal pixel image coordinates by $u = x/d_x$, and $v = y/d_y$. By combining Step 3 and Step 4 and if only using the first two radial distortion terms, we obtain the following relationship between $(\breve{u}, \breve{v})$ and (u, v):

$$\breve{u} = u + (u - u_0)[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2] \tag{1.3.11}$$
$$\breve{v} = v + (v - v_0)[k_1(x^2 + y^2) + k_2(x^2 + y^2)^2] \ . \tag{1.3.12}$$

[2]Note that the lens distortion described here is different from Tsai's treatment. Here, we go from ideal to real image coordinates, similar to [36].

Following the same reasoning as in (1.3.10), camera calibration including lens distortion can be performed by minimizing the distances between the image points and their predicted positions, i.e.,

$$\min_{\mathbf{A},\mathbf{R},\mathbf{t},k_1,k_2} \sum_i \|\mathbf{m}_i - \breve{\mathbf{m}}(\mathbf{A},\mathbf{R},\mathbf{t},k_1,k_2,\mathtt{M}_i)\|^2 \tag{1.3.13}$$

where $\breve{\mathbf{m}}(\mathbf{A},\mathbf{R},\mathbf{t},k_1,k_2,\mathtt{M}_i)$ is the projection of $\mathtt{M}_i$ onto the image according to (1.2.1), followed by distortion according to (1.3.11) and (1.3.12).

1.3.6 An Example

Figure 1.5 displays an image of a 3D reference object, taken by a camera to be calibrated at INRIA. Each square has 4 corners, and there are in total 128 points used for calibration.

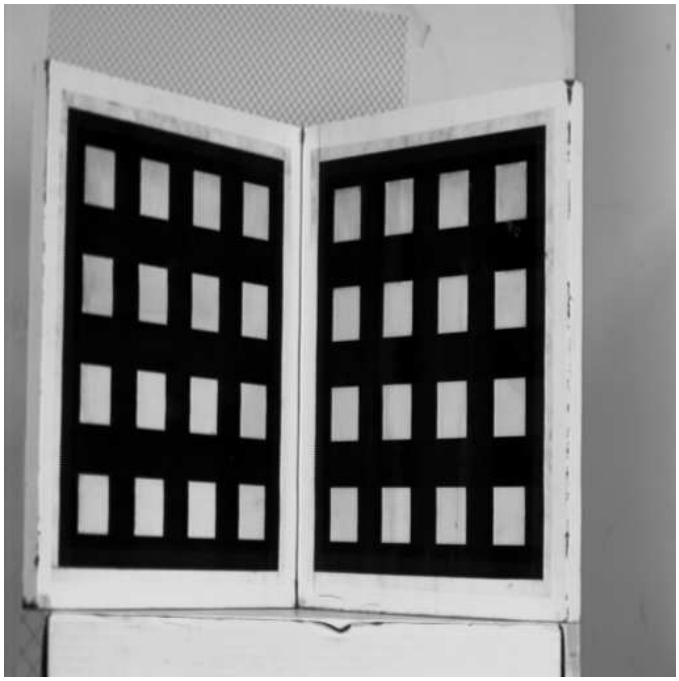


Figure 1.5. An example of camera calibration with a 3D apparatus

Without considering lens distortion, the estimated camera projection matrix is

$$\mathbf{P} = \begin{bmatrix} 7.025659e{-}01 & -2.861189e{-}02 & -5.377696e{-}01 & 6.241890e{+}01 \\ 2.077632e{-}01 & 1.265804e{+}00 & 1.591456e{-}01 & 1.075646e{+}01 \\ 4.634764e{-}04 & -5.282382e{-}05 & 4.255347e{-}04 & 1 \end{bmatrix}$$

From $\mathbf{P}$, we can calculate the intrinsic parameters: $\alpha = 1380.12$, $\beta = 2032.57$, $\gamma \approx 0$, $u_0 = 246.52$, and $v_0 = 243.68$. So, the angle between the two image axes is $90°$, and the aspect ratio of the pixels is $\alpha/\beta = 0.679$. For the extrinsic parameters, the translation vector $\mathbf{t} = [-211.28, -106.06, 1583.75]^T$ (in mm), i.e., the calibration object is about 1.5m away from the camera; the rotation axis is $[-0.08573, -0.99438, 0.0621]^T$ (i.e., almost vertical), and the rotation angle is $47.7°$.

Other notable work in this category include [27, 38, 36, 18].

1.4 Camera Calibration with 2D Objects: Plane-based Technique

In this section, we describe how a camera can be calibrated using a moving plane. We first examine the constraints on the camera's intrinsic parameters provided by observing a single plane.

1.4.1 Homography between the model plane and its image

Without loss of generality, we assume the model plane is on $Z = 0$ of the world coordinate system. Let's denote the i^{th} column of the rotation matrix $\mathbf{R}$ by $\mathbf{r}_i$. From (1.2.1), we have

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} .$$

By abuse of notation, we still use $\mathtt{M}$ to denote a point on the model plane, but $\mathtt{M} = [X, Y]^T$ since Z is always equal to 0. In turn, $\widetilde{\mathtt{M}} = [X, Y, 1]^T$. Therefore, a model point $\mathtt{M}$ and its image $\mathbf{m}$ is related by a homography $\mathbf{H}$:

$$s\widetilde{\mathbf{m}} = \mathbf{H}\widetilde{\mathtt{M}} \qquad \text{with} \quad \mathbf{H} = \mathbf{A} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} . \tag{1.4.1}$$

As is clear, the 3×3 matrix $\mathbf{H}$ is defined up to a scale factor.

1.4.2 Constraints on the intrinsic parameters

Given an image of the model plane, an homography can be estimated (see Appendix 1.A). Let's denote it by $\mathbf{H} = [\mathbf{h}_1 \quad \mathbf{h}_2 \quad \mathbf{h}_3]$. From (1.4.1), we have

$$[\mathbf{h}_1 \quad \mathbf{h}_2 \quad \mathbf{h}_3] = \lambda \mathbf{A} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix} ,$$

where λ is an arbitrary scalar. Using the knowledge that $\mathbf{r}_1$ and $\mathbf{r}_2$ are orthonormal, we have

$$\mathbf{h}_1^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{h}_2 = 0 \tag{1.4.2}$$

$$\mathbf{h}_1^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{h}_1 = \mathbf{h}_2^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{h}_2 . \tag{1.4.3}$$

These are the two basic constraints on the intrinsic parameters, given one homography. Because a homography has 8 degrees of freedom and there are 6 extrinsic parameters (3 for rotation and 3 for translation), we can only obtain 2 constraints on the intrinsic parameters. Note that $\mathbf{A}^{-T}\mathbf{A}^{-1}$ actually describes the image of the absolute conic [20]. In the next subsection, we will give an geometric interpretation.

1.4.3 Geometric Interpretation

We are now relating (1.4.2) and (1.4.3) to the absolute conic [22, 20].

It is not difficult to verify that the model plane, under our convention, is described in the camera coordinate system by the following equation:

$$\begin{bmatrix} \mathbf{r}_3 \\ \mathbf{r}_3^T\mathbf{t} \end{bmatrix}^T \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} = 0 \ ,$$

where $w = 0$ for points at infinity and $w = 1$ otherwise. This plane intersects the plane at infinity at a line, and we can easily see that $\begin{bmatrix} \mathbf{r}_1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} \mathbf{r}_2 \\ 0 \end{bmatrix}$ are two particular points on that line. Any point on it is a linear combination of these two points, i.e.,

$$\mathbf{x}_\infty = a \begin{bmatrix} \mathbf{r}_1 \\ 0 \end{bmatrix} + b \begin{bmatrix} \mathbf{r}_2 \\ 0 \end{bmatrix} = \begin{bmatrix} a\mathbf{r}_1 + b\mathbf{r}_2 \\ 0 \end{bmatrix} \ .$$

Now, let's compute the intersection of the above line with the absolute conic. By definition, the point $\mathbf{x}_\infty$, known as the *circular point* [26], satisfies: $\mathbf{x}_\infty^T\mathbf{x}_\infty = 0$, i.e., $(a\mathbf{r}_1 + b\mathbf{r}_2)^T(a\mathbf{r}_1 + b\mathbf{r}_2) = 0, \quad \text{or} \quad a^2 + b^2 = 0$. The solution is $b = \pm ai$, where $i^2 = -1$. That is, the two intersection points are

$$\mathbf{x}_\infty = a \begin{bmatrix} \mathbf{r}_1 \pm i\mathbf{r}_2 \\ 0 \end{bmatrix} \ .$$

The significance of this pair of complex conjugate points lies in the fact that they are invariant to Euclidean transformations. Their projection in the image plane is given, up to a scale factor, by

$$\widetilde{\mathbf{m}}_\infty = \mathbf{A}(\mathbf{r}_1 \pm i\mathbf{r}_2) = \mathbf{h}_1 \pm i\mathbf{h}_2 \ .$$

Point $\widetilde{\mathbf{m}}_\infty$ is on the image of the absolute conic, described by $\mathbf{A}^{-T}\mathbf{A}^{-1}$ [20]. This gives

$$(\mathbf{h}_1 \pm i\mathbf{h}_2)^T\mathbf{A}^{-T}\mathbf{A}^{-1}(\mathbf{h}_1 \pm i\mathbf{h}_2) = 0 \ .$$

Requiring that both real and imaginary parts be zero yields (1.4.2) and (1.4.3).

1.4.4 Closed-form solution

We now provide the details on how to effectively solve the camera calibration problem. We start with an analytical solution. This initial estimation will be followed by a nonlinear optimization technique based on the maximum likelihood criterion, to be described in the next subsection.

Let

$$\mathbf{B} = \mathbf{A}^{-T}\mathbf{A}^{-1} \equiv \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix} \tag{1.4.4}$$

$$= \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{\gamma}{\alpha^2\beta} & \frac{v_0\gamma - u_0\beta}{\alpha^2\beta} \\ -\frac{\gamma}{\alpha^2\beta} & \frac{\gamma^2}{\alpha^2\beta^2} + \frac{1}{\beta^2} & -\frac{\gamma(v_0\gamma - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} \\ \frac{v_0\gamma - u_0\beta}{\alpha^2\beta} & -\frac{\gamma(v_0\gamma - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} & \frac{(v_0\gamma - u_0\beta)^2}{\alpha^2\beta^2} + \frac{v_0^2}{\beta^2} + 1 \end{bmatrix} . \tag{1.4.5}$$

Note that $\mathbf{B}$ is symmetric, defined by a 6D vector

$$\mathbf{b} = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]^T . \tag{1.4.6}$$

Let the i^{th} column vector of $\mathbf{H}$ be $\mathbf{h}_i = [h_{i1}, h_{i2}, h_{i3}]^T$. Then, we have

$$\mathbf{h}_i^T \mathbf{B} \mathbf{h}_j = \mathbf{v}_{ij}^T \mathbf{b} \tag{1.4.7}$$

with $\mathbf{v}_{ij} = [h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}, h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}]^T$. Therefore, the two fundamental constraints (1.4.2) and (1.4.3), from a given homography, can be rewritten as 2 homogeneous equations in $\mathbf{b}$:

$$\begin{bmatrix} \mathbf{v}_{12}^T \\ (\mathbf{v}_{11} - \mathbf{v}_{22})^T \end{bmatrix} \mathbf{b} = \mathbf{0} . \tag{1.4.8}$$

If n images of the model plane are observed, by stacking n such equations as (1.4.8) we have

$$\mathbf{V}\mathbf{b} = \mathbf{0} , \tag{1.4.9}$$

where $\mathbf{V}$ is a $2n \times 6$ matrix. If $n \geq 3$, we will have in general a unique solution $\mathbf{b}$ defined up to a scale factor. If $n = 2$, we can impose the skewless constraint $\gamma = 0$, i.e., $[0, 1, 0, 0, 0, 0]\mathbf{b} = 0$, which is added as an additional equation to (1.4.9). (If $n = 1$, we can only solve two camera intrinsic parameters, e.g., α and β, assuming u_0 and v_0 are known (e.g., at the image center) and $\gamma = 0$, and that is indeed what we did in [28] for head pose determination based on the fact that eyes and mouth are reasonably coplanar. In fact, Tsai [33] already mentions that focal length from one plane is possible, but incorrectly says that aspect ratio is not.) The solution to (1.4.9) is well known as the eigenvector of $\mathbf{V}^T\mathbf{V}$ associated with the smallest eigenvalue (equivalently, the right singular vector of $\mathbf{V}$ associated with the smallest singular value).

Once $\mathbf{b}$ is estimated, we can compute all camera intrinsic parameters as follows. The matrix $\mathbf{B}$, as described in Sect. 1.4.4, is estimated up to a scale factor, i.e.,, $\mathbf{B} = \lambda \mathbf{A}^{-T}\mathbf{A}$ with λ an arbitrary scale. Without difficulty, we can uniquely extract the intrinsic parameters from matrix $\mathbf{B}$.

$$\begin{aligned} v_0 &= (B_{12}B_{13} - B_{11}B_{23})/(B_{11}B_{22} - B_{12}^2) \\ \lambda &= B_{33} - [B_{13}^2 + v_0(B_{12}B_{13} - B_{11}B_{23})]/B_{11} \\ \alpha &= \sqrt{\lambda/B_{11}} \\ \beta &= \sqrt{\lambda B_{11}/(B_{11}B_{22} - B_{12}^2)} \\ \gamma &= -B_{12}\alpha^2\beta/\lambda \\ u_0 &= \gamma v_0/\alpha - B_{13}\alpha^2/\lambda \ . \end{aligned}$$

Once $\mathbf{A}$ is known, the extrinsic parameters for each image is readily computed. From (1.4.1), we have

$$\mathbf{r}_1 = \lambda \mathbf{A}^{-1}\mathbf{h}_1 \ , \quad \mathbf{r}_2 = \lambda \mathbf{A}^{-1}\mathbf{h}_2 \ , \quad \mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \ , \quad \mathbf{t} = \lambda \mathbf{A}^{-1}\mathbf{h}_3$$

with $\lambda = 1/\|\mathbf{A}^{-1}\mathbf{h}_1\| = 1/\|\mathbf{A}^{-1}\mathbf{h}_2\|$. Of course, because of noise in data, the so-computed matrix $\mathbf{R} = [\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3]$ does not in general satisfy the properties of a rotation matrix. The best rotation matrix can then be obtained through for example singular value decomposition [13, 41].

1.4.5 Maximum likelihood estimation

The above solution is obtained through minimizing an algebraic distance which is not physically meaningful. We can refine it through maximum likelihood inference.

We are given n images of a model plane and there are m points on the model plane. Assume that the image points are corrupted by independent and identically distributed noise. The maximum likelihood estimate can be obtained by minimizing the following functional:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \|\mathbf{m}_{ij} - \hat{\mathbf{m}}(\mathbf{A}, \mathbf{R}_i, \mathbf{t}_i, \mathtt{M}_j)\|^2 \ , \tag{1.4.10}$$

where $\hat{\mathbf{m}}(\mathbf{A}, \mathbf{R}_i, \mathbf{t}_i, \mathtt{M}_j)$ is the projection of point $\mathtt{M}_j$ in image i, according to equation (1.4.1). A rotation $\mathbf{R}$ is parameterized by a vector of 3 parameters, denoted by $\mathbf{r}$, which is parallel to the rotation axis and whose magnitude is equal to the rotation angle. $\mathbf{R}$ and $\mathbf{r}$ are related by the Rodrigues formula [8]. Minimizing (1.4.10) is a nonlinear minimization problem, which is solved with the Levenberg-Marquardt Algorithm as implemented in `Minpack` [23]. It requires an initial guess of $\mathbf{A}, \{\mathbf{R}_i, \mathbf{t}_i | i = 1..n\}$ which can be obtained using the technique described in the previous subsection.

Desktop cameras usually have visible lens distortion, especially the radial components. We have included these while minimizing (1.4.10). See my technical report [41] for more details.

1.4.6 Dealing with radial distortion

Up to now, we have not considered lens distortion of a camera. However, a desktop camera usually exhibits significant lens distortion, especially radial distortion. The reader is referred to Section 1.3.5 for distortion modeling. In this section, we only consider the first two terms of radial distortion.

Estimating Radial Distortion by Alternation. As the radial distortion is expected to be small, one would expect to estimate the other five intrinsic parameters, using the technique described in Sect. 1.4.5, reasonable well by simply ignoring distortion. One strategy is then to estimate k_1 and k_2 after having estimated the other parameters, which will give us the ideal pixel coordinates (u, v). Then, from (1.3.11) and (1.3.12), we have two equations for each point in each image:

$$\begin{bmatrix} (u-u_0)(x^2+y^2) & (u-u_0)(x^2+y^2)^2 \\ (v-v_0)(x^2+y^2) & (v-v_0)(x^2+y^2)^2 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} \breve{u}-u \\ \breve{v}-v \end{bmatrix} .$$

Given m points in n images, we can stack all equations together to obtain in total $2mn$ equations, or in matrix form as $\mathbf{D}\mathbf{k} = \mathbf{d}$, where $\mathbf{k} = [k_1, k_2]^T$. The linear least-squares solution is given by

$$\mathbf{k} = (\mathbf{D}^T\mathbf{D})^{-1}\mathbf{D}^T\mathbf{d} . \tag{1.4.11}$$

Once k_1 and k_2 are estimated, one can refine the estimate of the other parameters by solving (1.4.10) with $\hat{\mathbf{m}}(\mathbf{A}, \mathbf{R}_i, \mathbf{t}_i, \mathtt{M}_j)$ replaced by (1.3.11) and (1.3.12). We can alternate these two procedures until convergence.

Complete Maximum Likelihood Estimation. Experimentally, we found the convergence of the above alternation technique is slow. A natural extension to (1.4.10) is then to estimate the complete set of parameters by minimizing the following functional:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \|\mathbf{m}_{ij} - \breve{\mathbf{m}}(\mathbf{A}, k_1, k_2, \mathbf{R}_i, \mathbf{t}_i, \mathtt{M}_j)\|^2 , \tag{1.4.12}$$

where $\breve{\mathbf{m}}(\mathbf{A}, k1, k2, \mathbf{R}_i, \mathbf{t}_i, \mathtt{M}_j)$ is the projection of point $\mathtt{M}_j$ in image i according to equation (1.4.1), followed by distortion according to (1.3.11) and (1.3.12). This is a nonlinear minimization problem, which is solved with the Levenberg-Marquardt Algorithm as implemented in `Minpack` [23]. A rotation is again parameterized by a 3-vector $\mathbf{r}$, as in Sect. 1.4.5. An initial guess of $\mathbf{A}$ and $\{\mathbf{R}_i, \mathbf{t}_i | i = 1..n\}$ can be obtained using the technique described in Sect. 1.4.4 or in Sect. 1.4.5. An initial guess of k_1 and k_2 can be obtained with the technique described in the last paragraph, or simply by setting them to 0.

1.4.7 Summary

The recommended calibration procedure is as follows:

1. Print a pattern and attach it to a planar surface;

2. Take a few images of the model plane under different orientations by moving either the plane or the camera;

3. Detect the feature points in the images;

4. Estimate the five intrinsic parameters and all the extrinsic parameters using the closed-form solution as described in Sect. 1.4.4;

5. Estimate the coefficients of the radial distortion by solving the linear least-squares (1.4.11);

6. Refine all parameters, including lens distortion parameters, by minimizing (1.4.12).

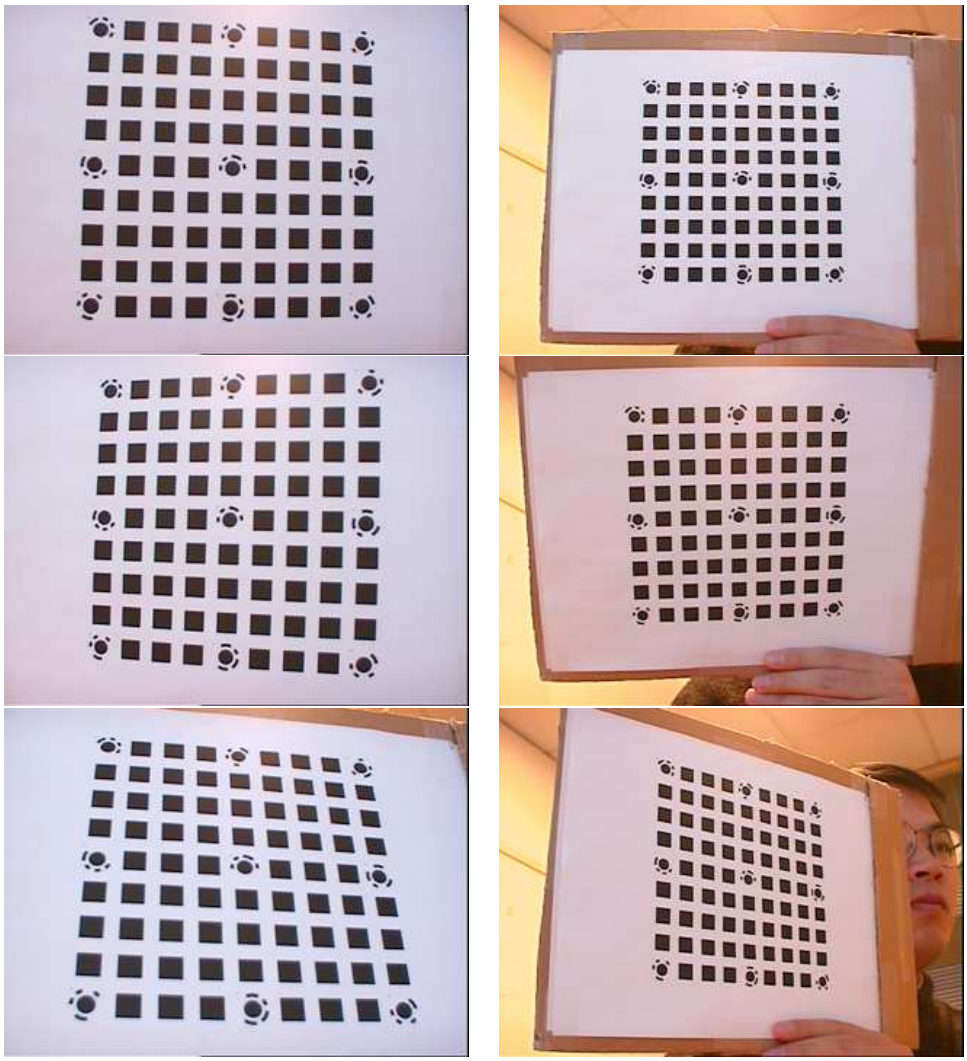
There is a degenerate configuration in my technique when planes are parallel to each other. See my technical report [41] for a more detailed description.

In summary, this technique only requires the camera to observe a planar pattern from a few different orientations. Although the minimum number of orientations is two if pixels are square, we recommend 4 or 5 different orientations for better quality. We can move either the camera or the planar pattern. The motion does not need to be known, but should not be a pure translation. When the number of orientations is only 2, one should avoid positioning the planar pattern parallel to the image plane. The pattern could be anything, as long as we know the metric on the plane. For example, we can print a pattern with a laser printer and attach the paper to a reasonable planar surface such as a hard book cover. We can even use a book with known size because the four corners are enough to estimate the plane homographies.

1.4.8 Experimental Results

The proposed algorithm has been tested on both computer simulated data and real data. The closed-form solution involves finding a singular value decomposition of a small $2n \times 6$ matrix, where n is the number of images. The nonlinear refinement within the Levenberg-Marquardt algorithm takes 3 to 5 iterations to converge. Due to space limitation, we describe in this section one set of experiments with real data when the calibration pattern is at different distances from the camera. The reader is referred to [41] for more experimental results with both computer simulated and real data, and to the following Web page:
`http://research.microsoft.com/~zhang/Calib/`
for some experimental data and the software.

The example is shown in Fig. 1.6. The camera to be calibrated is an off-the-shelf PULNiX CCD camera with 6 mm lens. The image resolution is 640×480. As can be seen in Fig. 1.6, the model plane contains a 9×9 squares with 9 special dots which are used to identify automatically the correspondence between reference points on the model plane and square corners in images. It was printed on a A4 paper with a 600 DPI laser printer, and attached to a cardboard.



(A) (B)

Figure 1.6. Two sets of images taken at different distances to the calibration pattern. Each set contains five images. On the left, three images from the set taken at a close distance are shown. On the right, three images from the set taken at a larger distance are shown.

Table 1.1. Calibration results with the images shown in Figure 1.6

image set	α	β	ϑ	u_0	v_0	k_1	k_2
A	834.01	839.86	89.95°	305.51	240.09	-0.2235	0.3761
B	836.17	841.08	89.92°	301.76	241.51	-0.2676	1.3121
A+B	834.64	840.32	89.94°	304.77	240.59	-0.2214	0.3643

In total 10 images of the plane were taken (6 of them are shown in Fig. 1.6). Five of them (called Set A) were taken at close range, while the other five (called Set B) were taken at a larger distance. We applied our calibration algorithm to Set A, Set B, and also to the whole set (called Set A+B). The results are shown in Table 1.1. For intuitive understanding, we show the estimated angle between the image axes, ϑ, instead of the skew factor γ. We can see that the angle ϑ is very close to 90°, as expected with almost all modern CCD cameras. The cameras parameters were estimated consistently for all three sets of images, except the distortion parameters with Set B. The reason is that the calibration pattern only occupies the central part of the image in Set B, where lens distortion is not significant and therefore cannot be estimated reliably.

1.4.9 Related Work

Almost at the same time, Sturm and Maybank [31], independent from us, developed the same technique. They assumed the pixels are square (i.e., $\gamma = 0$) and have studied the degenerate configurations for plane-based camera calibration.

Gurdjos et al. [14] have re-derived the plane-based calibration technique from the center line constraint.

My original implementation (only the executable) is available at
`http://research.microsoft.com/~zhang/calib/`.
Bouguet has re-implemented my technique in Matlab, which is available at
`http://www.vision.caltech.edu/bouguetj/calib_doc/`.

In many applications such as stereo, multiple cameras need to be calibrated simultaneously in order to determine the relative geometry between cameras. In 2000, I have extended (not published) this plane-based technique to stereo calibration for my stereo-based gaze-correction project [40, 39]. The formulation is similar to (1.4.12). Consider two cameras, and denote the quantity related to the second camera by $'$. Let $(\mathbf{R}_s, \mathbf{t}_s)$ be the rigid transformation between the two cameras such that $(\mathbf{R}', \mathbf{t}') = (\mathbf{R}, \mathbf{t}) \circ (\mathbf{R}_s, \mathbf{t}_s)$ or more precisely: $\mathbf{R}' = \mathbf{R}\mathbf{R}_s$ and $\mathbf{t}' = \mathbf{R}\mathbf{t}_s + \mathbf{t}$. Stereo calibration is then to solve $\mathbf{A}, \mathbf{A}', k_1, k_2, k_1', k_2', \{(\mathbf{R}_i, \mathbf{t}_i) | i = 1, \ldots, n\}$, and

$(\mathbf{R}_s, \mathbf{t}_s)$ by minimizing the following functional:

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\left[\delta_{ij}\|\mathbf{m}_{ij} - \breve{\mathbf{m}}(\mathbf{A}, k_1, k_2, \mathbf{R}_i, \mathbf{t}_i, \mathtt{M}_j)\|^2 + \delta'_{ij}\|\mathbf{m}'_{ij} - \breve{\mathbf{m}}(\mathbf{A}', k'_1, k'_2, \mathbf{R}'_i, \mathbf{t}'_i, \mathtt{M}_j)\|^2\right] \tag{1.4.13}$$

subject to

$$\mathbf{R}'_i = \mathbf{R}_i\mathbf{R}_s \quad \text{and} \quad \mathbf{t}'_i = \mathbf{R}_i\mathbf{t}_s + \mathbf{t}_i \ .$$

In the above formulation, $\delta_{ij} = 1$ if point j is visible in the first camera, and $\delta_{ij} = 0$ otherwise. Similarly, $\delta'_{ij} = 1$ if point j is visible in the second camera. This formulation thus does not require the same number of feature points to be visible over time or across cameras. Another advantage of this formulation is that the number of extrinsic parameters to be estimated has been reduced from $12n$ if the two cameras are calibrated independently to $6n + 6$. This is a reduction of 24 dimensions in parameter space if 5 planes are used.

Obviously, this is a nonlinear optimization problem. To obtain the initial guess, we run first single-camera calibration independently for each camera, and compute $\mathbf{R}_s$ through SVD from $\mathbf{R}'_i = \mathbf{R}_i\mathbf{R}_s$ $(i = 1, \ldots, n)$ and $\mathbf{t}_s$ through least-squares from $\mathbf{t}'_i = \mathbf{R}_i\mathbf{t}_s + \mathbf{t}_i$ $(i = 1, \ldots, n)$. Recently, a closed-form initialization technique through factorization of homography matrices is proposed in [34].

1.5 Solving Camera Calibration With 1D Objects

In this section, we describe in detail how to solve the camera calibration problem from a number of observations of a 1D object consisting of 3 collinear points moving around one of them [43, 44]. We only consider this minimal configuration, but it is straightforward to extend the result if a calibration object has four or more collinear points.

1.5.1 Setups With Free-Moving 1D Calibration Objects

We now examine possible setups with 1D objects for camera calibration. As already mentioned in the introduction, we need to have several observations of the 1D objects. Without loss of generality, we choose the camera coordinate system to define the 1D objects; therefore, $\mathbf{R} = \mathbf{I}$ and $\mathbf{t} = \mathbf{0}$ in (1.2.1).

Two points with known distance. This could be the two endpoints of a stick, and we take a number of images while waving freely the stick. Let `A` and `B` be the two 3D points, and $\mathbf{a}$ and $\mathbf{b}$ be the observed image points. Because the distance between `A` and `B` is known, we only need 5 parameters to define `A` and `B`. For example, we need 3 parameters to specify the coordinates of `A` in the camera coordinate system, and 2 parameters to define the orientation of the line `AB`. On the other hand, each image point provides two equations according to (1.2.1), giving in total 4 equations. Given N observations of the stick, we have 5 intrinsic parameters and $5N$ parameters for the point positions to estimate, i.e., the total number of unknowns is $5 + 5N$. However, we only have $4N$ equations. Camera calibration is thus impossible.

Three collinear points with known distances. By adding an additional point, say C, the number of unknowns for the point positions still remains the same, i.e., $5 + 5N$, because of known distances of C to A and B. For each observation, we have three image points, yielding in total $6N$ equations. Calibration seems to be plausible, but is in fact not. This is because the three image points for each observation must be collinear. Collinearity is preserved by perspective projection. We therefore only have 5 independent equations for each observation. The total number of independent equations, $5N$, is always smaller than the number of unknowns. Camera calibration is still impossible.

Four or more collinear points with known distances. As seen above, when the number of points increases from two to three, the number of independent equations (constraints) increases by one for each observation. If we have a fourth point, will we have in total $6N$ independent equations? If so, we would be able to solve the problem because the number of unknowns remains the same, i.e., $5 + 5N$, and we would have more than enough constraints if $N \geq 5$. The reality is that the addition of the fourth point or even more points does not increase the number of independent equations. It will always be $5N$ for any four or more collinear points. This is because the cross ratio is preserved under perspective projection. With known cross ratios and three collinear points, whether they are in space or in images, other points are determined exactly.

1.5.2 Setups With 1D Calibration Objects Moving Around a fixed Point

From the above discussion, calibration is impossible with a free moving 1D calibration object, no matter how many points on the object. Now let us examine what happens if one point is fixed. In the sequel, without loss of generality, point A is the fixed point, and $\mathbf{a}$ is the corresponding image point. We need 3 parameters, which are unknown, to specify the coordinates of A in the camera coordinate system, while image point $\mathbf{a}$ provides two scalar equations according to (1.2.1).

Two points with known distance. They could be the endpoints of a stick, and we move the stick around the endpoint that is fixed. Let B be the free endpoint and $\mathbf{b}$, its corresponding image point. For each observation, we need 2 parameters to define the orientation of the line AB and therefore the position of B because the distance between A and B is known. Given N observations of the stick, we have 5 intrinsic parameters, 3 parameters for A and $2N$ parameters for the free endpoint positions to estimate, i.e., the total number of unknowns is $8 + 2N$. However, each observation of $\mathbf{b}$ provides two equations, so together with $\mathbf{a}$ we only have in total $2 + 2N$ equations. Camera calibration is thus impossible.

Three collinear points with known distances. As already explained in the last subsection, by adding an additional point, say C, the number of unknowns for the point positions still remains the same, i.e., $8 + 2N$. For each observation, $\mathbf{b}$ provides two equations, but $\mathbf{c}$ only provides one additional equation because of

the collinearity of **a**, **b** and **c**. Thus, the total number of equations is $2 + 3N$ for N observations. By counting the numbers, we see that if we have 6 or more observations, we should be able to solve camera calibration, and this is the case as we shall show in the next section.

Four or more collinear points with known distances. Again, as already explained in the last subsection, The number of unknowns and the number of independent equations remain the same because of invariance of cross-ratios. This said, the more collinear points we have, the more accurate camera calibration will be in practice because data redundancy can combat the noise in image data.

1.5.3 Basic Equations

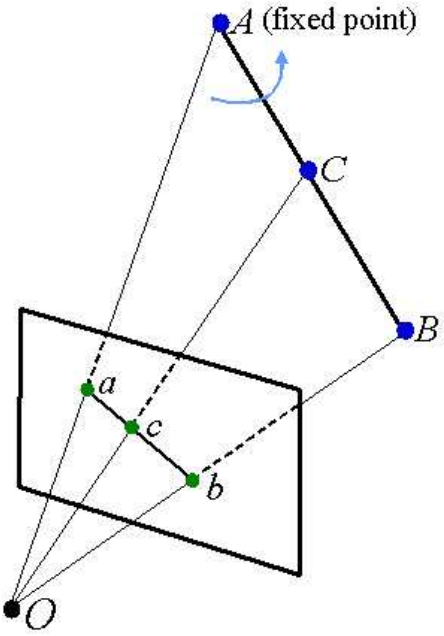


Figure 1.7. Illustration of 1D calibration objects

Refer to Figure 1.7. Point $\mathtt{A}$ is the fixed point in space, and the stick AB moves around $\mathtt{A}$. The length of the stick AB is known to be L, i.e.,

$$\|\mathtt{B} - \mathtt{A}\| = L \; . \tag{1.5.1}$$

The position of point $\mathtt{C}$ is also known with respect to $\mathtt{A}$ and $\mathtt{B}$, and therefore

$$\mathtt{C} = \lambda_A \mathtt{A} + \lambda_B \mathtt{B} \; , \tag{1.5.2}$$

where λ_A and λ_B are known. If C is the midpoint of AB, then $\lambda_A = \lambda_B = 0.5$. Points **a**, **b** and **c** on the image plane are projection of space points A, B and C, respectively.

Without loss of generality, we choose the camera coordinate system to define the 1D objects; therefore, $\mathbf{R} = \mathbf{I}$ and $\mathbf{t} = \mathbf{0}$ in (1.2.1). Let the unknown depths for A, B and C be z_A, z_B and z_C, respectively. According to (1.2.1), we have

$$\mathtt{A} = z_A \mathbf{A}^{-1} \widetilde{\mathbf{a}} \tag{1.5.3}$$

$$\mathtt{B} = z_B \mathbf{A}^{-1} \widetilde{\mathbf{b}} \tag{1.5.4}$$

$$\mathtt{C} = z_C \mathbf{A}^{-1} \widetilde{\mathbf{c}} \; . \tag{1.5.5}$$

Substituting them into (1.5.2) yields

$$z_C \widetilde{\mathbf{c}} = z_A \lambda_A \widetilde{\mathbf{a}} + z_B \lambda_B \widetilde{\mathbf{b}} \tag{1.5.6}$$

after eliminating $\mathbf{A}^{-1}$ from both sides. By performing cross-product on both sides of the above equation with $\widetilde{\mathbf{c}}$, we have

$$z_A \lambda_A (\widetilde{\mathbf{a}} \times \widetilde{\mathbf{c}}) + z_B \lambda_B (\widetilde{\mathbf{b}} \times \widetilde{\mathbf{c}}) = \mathbf{0} \; .$$

In turn, we obtain

$$z_B = -z_A \frac{\lambda_A (\widetilde{\mathbf{a}} \times \widetilde{\mathbf{c}}) \cdot (\widetilde{\mathbf{b}} \times \widetilde{\mathbf{c}})}{\lambda_B (\widetilde{\mathbf{b}} \times \widetilde{\mathbf{c}}) \cdot (\widetilde{\mathbf{b}} \times \widetilde{\mathbf{c}})} \; . \tag{1.5.7}$$

From (1.5.1), we have

$$\| \mathbf{A}^{-1} (z_B \widetilde{\mathbf{b}} - z_A \widetilde{\mathbf{a}}) \| = L \; .$$

Substituting z_B by (1.5.7) gives

$$z_A \| \mathbf{A}^{-1} \big(\widetilde{\mathbf{a}} + \frac{\lambda_A (\widetilde{\mathbf{a}} \times \widetilde{\mathbf{c}}) \cdot (\widetilde{\mathbf{b}} \times \widetilde{\mathbf{c}})}{\lambda_B (\widetilde{\mathbf{b}} \times \widetilde{\mathbf{c}}) \cdot (\widetilde{\mathbf{b}} \times \widetilde{\mathbf{c}})} \widetilde{\mathbf{b}} \big) \| = L \; .$$

This is equivalent to

$$z_A^2 \mathbf{h}^T \mathbf{A}^{-T} \mathbf{A}^{-1} \mathbf{h} = L^2 \tag{1.5.8}$$

with

$$\mathbf{h} = \widetilde{\mathbf{a}} + \frac{\lambda_A (\widetilde{\mathbf{a}} \times \widetilde{\mathbf{c}}) \cdot (\widetilde{\mathbf{b}} \times \widetilde{\mathbf{c}})}{\lambda_B (\widetilde{\mathbf{b}} \times \widetilde{\mathbf{c}}) \cdot (\widetilde{\mathbf{b}} \times \widetilde{\mathbf{c}})} \widetilde{\mathbf{b}} \; . \tag{1.5.9}$$

Equation (1.5.8) contains the unknown intrinsic parameters $\mathbf{A}$ and the unknown depth, z_A, of the fixed point A. It is the basic constraint for camera calibration with 1D objects. Vector $\mathbf{h}$, given by (1.5.9), can be computed from image points and known λ_A and λ_B. Since the total number of unknowns is 6, we need at least six observations of the 1D object for calibration. Note that $\mathbf{A}^{-T}\mathbf{A}$ actually describes the image of the absolute conic [20].

1.5.4 Closed-Form Solution

Let

$$\mathbf{B} = \mathbf{A}^{-T}\mathbf{A}^{-1} \equiv \begin{bmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{bmatrix} \tag{1.5.10}$$

$$= \begin{bmatrix} \frac{1}{\alpha^2} & -\frac{\gamma}{\alpha^2\beta} & \frac{v_0\gamma - u_0\beta}{\alpha^2\beta} \\ -\frac{\gamma}{\alpha^2\beta} & \frac{\gamma^2}{\alpha^2\beta^2} + \frac{1}{\beta^2} & -\frac{\gamma(v_0\gamma - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} \\ \frac{v_0\gamma - u_0\beta}{\alpha^2\beta} & -\frac{\gamma(v_0\gamma - u_0\beta)}{\alpha^2\beta^2} - \frac{v_0}{\beta^2} & \frac{(v_0\gamma - u_0\beta)^2}{\alpha^2\beta^2} + \frac{v_0^2}{\beta^2} + 1 \end{bmatrix} . \tag{1.5.11}$$

Note that $\mathbf{B}$ is symmetric, and can be defined by a 6D vector

$$\mathbf{b} = [B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}]^T . \tag{1.5.12}$$

Let $\mathbf{h} = [h_1, h_2, h_3]^T$, and $\mathbf{x} = z_A^2\mathbf{b}$, then equation (1.5.8) becomes

$$\mathbf{v}^T\mathbf{x} = L^2 \tag{1.5.13}$$

with

$$\mathbf{v} = [h_1^2, 2h_1h_2, h_2^2, 2h_1h_3, 2h_2h_3, h_3^2]^T .$$

When N images of the 1D object are observed, by stacking n such equations as (1.5.13) we have

$$\mathbf{V}\mathbf{x} = L^2\mathbf{1} , \tag{1.5.14}$$

where $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_N]^T$ and $\mathbf{1} = [1, \ldots, 1]^T$. The least-squares solution is then given by

$$\mathbf{x} = L^2(\mathbf{V}^T\mathbf{V})^{-1}\mathbf{V}^T\mathbf{1} . \tag{1.5.15}$$

Once $\mathbf{x}$ is estimated, we can compute all the unknowns based on $\mathbf{x} = z_A^2\mathbf{b}$. Let $\mathbf{x} = [x_1, x_2, \ldots, x_6]^T$. Without difficulty, we can uniquely extract the intrinsic parameters and the depth z_A as

$$\begin{aligned} v_0 &= (x_2x_4 - x_1x_5)/(x_1x_3 - x_2^2) \\ z_A &= \sqrt{x_6 - [x_4^2 + v_0(x_2x_4 - x_1x_5)]/x_1} \\ \alpha &= \sqrt{z_A/x_1} \\ \beta &= \sqrt{z_Ax_1/(x_1x_3 - x_2^2)} \\ \gamma &= -x_2\alpha^2\beta/z_A \\ u_0 &= \gamma v_0/\alpha - x_4\alpha^2/z_A . \end{aligned}$$

At this point, we can compute z_B according to (1.5.7), so points `A` and `B` can be computed from (1.5.3) and (1.5.4), while point `C` can be computed according to (1.5.2).

1.5.5 Nonlinear Optimization

The above solution is obtained through minimizing an algebraic distance which is not physically meaningful. We can refine it through maximum likelihood inference.

We are given N images of the 1D calibration object and there are 3 points on the object. Point $\mathtt{A}$ is fixed, and points $\mathtt{B}$ and $\mathtt{C}$ moves around $\mathtt{A}$. Assume that the image points are corrupted by independent and identically distributed noise. The maximum likelihood estimate can be obtained by minimizing the following functional:

$$\sum_{i=1}^{N} \left(\|\mathbf{a}_i - \boldsymbol{\phi}(\mathbf{A}, \mathtt{A})\|^2 + \|\mathbf{b}_i - \boldsymbol{\phi}(\mathbf{A}, \mathtt{B}_i)\|^2 + \|\mathbf{c}_i - \boldsymbol{\phi}(\mathbf{A}, \mathtt{C}_i)\|^2 \right) , \tag{1.5.16}$$

where $\boldsymbol{\phi}(\mathbf{A}, \mathtt{M})$ ($\mathtt{M} \in \{\mathtt{A}, \mathtt{B}_i, \mathtt{C}_i\}$) is the projection of point $\mathtt{M}$ onto the image, according to equations (1.5.3) to (1.5.5). More precisely, $\boldsymbol{\phi}(\mathbf{A}, \mathtt{M}) = \frac{1}{z_M}\mathbf{A}\mathtt{M}$, where z_M is the z-component of $\mathtt{M}$.

The unknowns to be estimated are:

- 5 camera intrinsic parameters α, β, γ, u_0 and v_0 that define matrix $\mathbf{A}$;
- 3 parameters for the coordinates of the fixed point $\mathtt{A}$;
- $2N$ additional parameters to define points $\mathtt{B}_i$ and $\mathtt{C}_i$ at each instant (see below for more details).

Therefore, we have in total $8 + 2N$ unknowns. Regarding the parameterization for $\mathtt{B}$ and $\mathtt{C}$, we use the spherical coordinates ϕ and θ to define the direction of the 1D calibration object, and point $\mathtt{B}$ is then given by

$$\mathtt{B} = \mathtt{A} + L \begin{bmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{bmatrix}$$

where L is the known distance between $\mathtt{A}$ and $\mathtt{B}$. In turn, point $\mathtt{C}$ is computed according to (1.5.2). We therefore only need 2 additional parameters for each observation.

Minimizing (1.5.16) is a nonlinear minimization problem, which is solved with the Levenberg-Marquardt Algorithm as implemented in `Minpack` [23]. It requires an initial guess of $\mathbf{A}, \mathtt{A}, \{\mathtt{B}_i, \mathtt{C}_i | i = 1..N\}$ which can be obtained using the technique described in the last subsection.

1.5.6 Estimating the fixed point

In the above discussion, we assumed that the image coordinates, $\mathbf{a}$, of the fixed point $\mathtt{A}$ are known. We now describe how to estimate $\mathbf{a}$ by considering whether the fixed point $\mathtt{A}$ is visible in the image or not.

Invisible fixed point. The fixed point does not need to be visible in the image. And the camera calibration technique becomes more versatile without the visibility requirement. In that case, we can for example hang a string of small balls from the ceiling, and calibrate multiple cameras in the room by swinging the string. The fixed point can be estimated by intersecting lines from different images as described below.

Each observation of the 1D object defines an image line. An image line can be represented by a 3D vector $\mathbf{l} = [l_1, l_2, l_3]^T$, defined up to a scale factor such as a point $\mathbf{m} = [u, v]^T$ on the line satisfies $\mathbf{l}^T \widetilde{\mathbf{m}} = 0$. In the sequel, we also use $(\mathbf{n}, q)$ to denote line $\mathbf{l}$, where $\mathbf{n} = [l_1, l_2]^T$ and $q = l_3$. To remove the scale ambiguity, we normalize $\mathbf{l}$ such that $\|\mathbf{l}\| = 1$. Furthermore, each $\mathbf{l}$ is associated with an uncertainty measure represented by a 3×3 covariance matrix $\mathbf{\Lambda}$.

Given N images of the 1D object, we have N lines: $\{(\mathbf{l}_i, \mathbf{\Lambda}_i) | i = 1, \ldots, N\}$. Let the fixed point be $\mathbf{a}$ in the image. Obviously, if there is no noise, we have $\mathbf{l}_i^T \widetilde{\mathbf{a}} = 0$, or $\mathbf{n}_i^T \mathbf{a} + q_i = 0$. Therefore, we can estimate $\mathbf{a}$ by minimizing

$$\mathcal{F} = \sum_{i=1}^{N} w_i \|\mathbf{l}_i^T \widetilde{\mathbf{a}}\|^2 = \sum_{i=1}^{N} w_i \|\mathbf{n}_i^T \mathbf{a} + q_i\|^2 = \sum_{i=1}^{N} w_i (\mathbf{a}^T \mathbf{n}_i \mathbf{n}_i^T \mathbf{a} + 2 q_i \mathbf{n}_i^T \mathbf{a} + q_i^2) \quad (1.5.17)$$

where w_i is a weighting factor (see below). By setting the derivative of $\mathcal{F}$ with respect to $\mathbf{a}$ to 0, we obtain the solution, which is given by

$$\mathbf{a} = -\Big(\sum_{i=1}^{N} w_i \mathbf{n}_i \mathbf{n}_i^T \Big)^{-1} \Big(\sum_{i=1}^{N} w_i q_i \mathbf{n}_i \Big) .$$

The optimal weighting factor w_i in (1.5.17) is the inverse of the variance of $\mathbf{l}_i^T \widetilde{\mathbf{a}}$, which is $w_i = 1/(\widetilde{\mathbf{a}}^T \mathbf{\Lambda}_i \widetilde{\mathbf{a}})$. Note that the weight w_i involves the unknown $\mathbf{a}$. To overcome this difficulty, we can approximate w_i by $1/\operatorname{trace}(\mathbf{\Lambda}_i)$ for the first iteration, and by re-computing w_i with the previously estimated $\mathbf{a}$ in the subsequent iterations. Usually two or three iterations are enough.

Visible fixed point. Since the fixed point is visible, we have N observations: $\{\mathbf{a}_i | i = 1, \ldots, N\}$. We can therefore estimate $\mathbf{a}$ by minimizing $\sum_{i=1}^{N} \|\mathbf{a} - \mathbf{a}_i\|^2$, assuming that the image points are detected with the same accuracy. The solution is simply $\mathbf{a} = (\sum_{i=1}^{N} a_i)/N$.

The above estimation does not make use of the fact that the fixed point is also the intersection of the N observed lines of the 1D object. Therefore, a better technique to estimate $\mathbf{a}$ is to minimize the following function:

$$\mathcal{F} = \sum_{i=1}^{N} \left[(\mathbf{a} - \mathbf{a}_i)^T \mathbf{V}_i^{-1} (\mathbf{a} - \mathbf{a}_i) + w_i \|\mathbf{l}_i^T \widetilde{\mathbf{a}}\|^2 \right] = \sum_{i=1}^{N} \left[(\mathbf{a} - \mathbf{a}_i)^T \mathbf{V}_i^{-1} (\mathbf{a} - \mathbf{a}_i) + w_i \|\mathbf{n}_i^T \mathbf{a} + q_i\|^2 \right] \quad (1.5.18)$$

where $\mathbf{V}_i$ is the covariance matrix of the detected point $\mathbf{a}_i$. The derivative of the

above function with respect to $\mathbf{a}$ is given by

$$\frac{\partial \mathcal{F}}{\partial \mathbf{a}} = 2\sum_{i=1}^{N} \left[\mathbf{V}_i^{-1}(\mathbf{a} - \mathbf{a}_i) + w_i \mathbf{n}_i \mathbf{n}_i^T \mathbf{a} + w_i q_i \mathbf{n}_i\right] .$$

Setting it to 0 yields

$$\mathbf{a} = \Big(\sum_{i=1}^{N} (\mathbf{V}_i^{-1} + w_i \mathbf{n}_i \mathbf{n}_i^T)\Big)^{-1} \Big(\sum_{i=1}^{N} (\mathbf{V}_i^{-1}\mathbf{a}_i - w_i q_i \mathbf{n}_i)\Big) .$$

If more than three points are visible in each image, the known cross ratio provides an additional constraint in determining the fixed point.

For an accessible description of uncertainty manipulation, the reader is referred to [45, Chapter 2].

1.5.7 Experimental Results

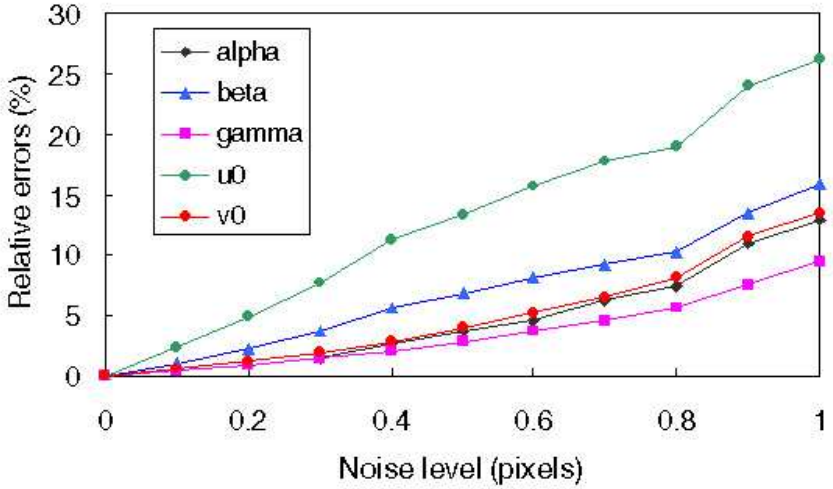
The proposed algorithm has been tested on both computer simulated data and real data.

Computer Simulations

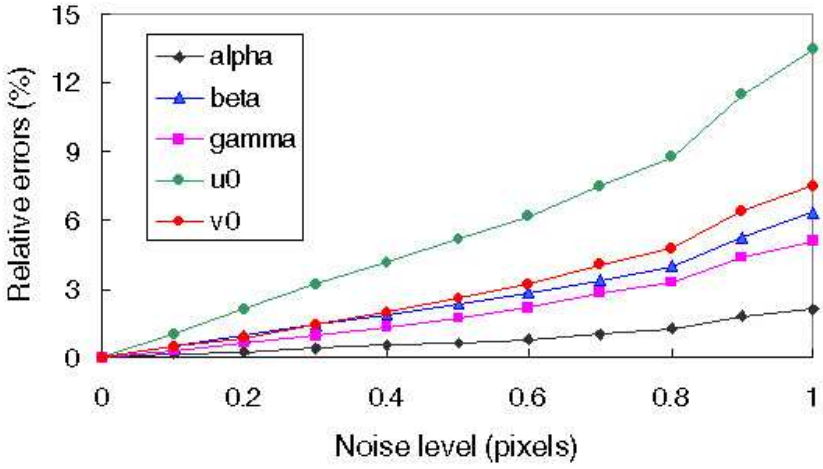
The simulated camera has the following property: $\alpha = 1000$, $\beta = 1000$, $\gamma = 0$, $u_0 = 320$, and $v_0 = 240$. The image resolution is 640×480. A stick of 70 cm is simulated with the fixed point `A` at $[0, 35, 150]^T$. The other endpoint of the stick is `B`, and `C` is located at the half way between `A` and `B`. We have generated 100 random orientations of the stick by sampling θ in $[\pi/6, 5\pi/6]$ and ϕ in $[\pi, 2\pi]$ according to uniform distribution. Points `A`, `B`, and `C` are then projected onto the image.

Gaussian noise with 0 mean and σ standard deviation is added to the projected image points $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{c}$. The estimated camera parameters are compared with the ground truth, and we measure their relative errors with respect to the focal length α. Note that we measure the relative errors in (u_0, v_0) with respect to α, as proposed by Triggs in [32]. He pointed out that the absolute errors in (u_0, v_0) is not geometrically meaningful, while computing the relative error is equivalent to measuring the angle between the true optical axis and the estimated one.

We vary the noise level from 0.1 pixels to 1 pixel. For each noise level, we perform 120 independent trials, and the results shown in Fig. 1.8 are the average. Figure 1.8a displays the relative errors of the closed-form solution while Figure 1.8b displays those of the nonlinear minimization result. Errors increase almost linearly with the noise level. The nonlinear minimization refines the closed-form solution, and produces significantly better result (with 50% less errors). At 1 pixel noise level, the errors for the closed-form solution are about 12%, while those for the nonlinear minimization are about 6%.



(a) Closed-form solution



(b) Nonlinear optimization

Figure 1.8. Calibration errors with respect to the noise level of the image points.

Table 1.2. Calibration results with real data.

Solution	α	β	γ	u_0	v_0
Closed-form	889.49	818.59	-0.1651 (90.01°)	297.47	234.33
Nonlinear	838.49	799.36	4.1921 (89.72°)	286.74	219.89
Plane-based	828.92	813.33	-0.0903 (90.01°)	305.23	235.17
Relative difference	1.15%	1.69%	0.52% (0.29°)	2.23%	1.84%

Frame 10 Frame 60

Frame 90 Frame 140

Figure 1.9. Sample images of a 1D object used for camera calibration.

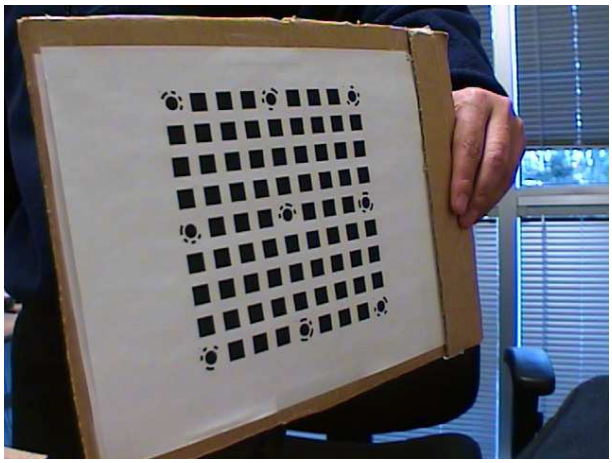


Figure 1.10. A sample image of the planar pattern used for camera calibration.

Real Data

For the experiment with real data, I used three toy beads from my kids and strung them together with a stick. The beads are approximately 14 cm apart (i.e., $L = 28$). I then moves the stick around while trying to fix one end with the aid of a book. A video of 150 frames was recorded, and four sample images are shown in Fig. 1.9. A bead in the image is modeled as a Gaussian blob in the RGB space, and the centroid of each detected blob is the image point we use for camera calibration. The proposed algorithm is therefore applied to the 150 observations of the beads, and the estimated camera parameters are provided in Table 1.2. The first row is the estimation from the closed-form solution, while the second row is the refined result after nonlinear minimization. For the image skew parameter γ, we also provide the angle between the image axes in parenthesis (it should be very close to 90°).

For comparison, we also used the plane-based calibration technique described in [42] to calibrate the same camera. Five images of a planar pattern were taken, and one of them is shown in Fig. 1.10. The calibration result is shown in the third row of Table 1.2. The fourth row displays the relative difference between the plane-based result and the nonlinear solution with respect to the focal length (we use 828.92). As we can observe, the difference is about 2%.

There are several sources contributing to this difference. Besides obviously the image noise and imprecision of the extracted data points, one source is our current rudimentary experimental setup:

- The supposed-to-be fixed point was not fixed. It slipped around on the surface.
- The positioning of the beads was done with a ruler using eye inspection.

Considering all the factors, the proposed algorithm is very encouraging.

1.6 Self-Calibration

Self-calibration is also called auto-calibration. Techniques in this category do not require any particular calibration object. They can be considered as 0D approach because only image point correspondences are required. Just by moving a camera in a static scene, the rigidity of the scene provides in general two constraints [22, 21, 20] on the cameras' internal parameters from one camera displacement by using image information alone. Absolute conic, described in Section 1.2.2, is an essential concept in understanding these constraints. Therefore, if images are taken by the same camera with fixed internal parameters, correspondences between three images are sufficient to recover both the internal and external parameters which allow us to reconstruct 3-D structure up to a similarity [20, 17]. Although no calibration objects are necessary, a large number of parameters need to be estimated, resulting in a much harder mathematical problem.

We do not plan to go further into details of this approach because two recent books [15, 7] provide an excellent recount of those techniques.

1.7 Conclusion

In this chapter, we have reviewed several camera calibration techniques. We have classified them into four categories, depending whether they use 3D apparatus, 2D objects (planes), 1D objects, or just the surrounding scenes (self-calibration). Recommendations on choosing which technique were given in the introduction section.

The techniques described so far are mostly focused on a single-camera calibration. We touched a little bit on stereo calibration in Section 1.4.9. Camera calibration is still an active research area because more and more applications use cameras. In [2], spheres are used to calibrate one or more cameras, which can be considered as a 2D approach since only the surface property is used. In [5], a technique is described to calibrate a camera network consisting of an omni-camera and a number of perspective cameras. In [24], a technique is proposed to calibrate a projector-screen-camera system.

1.A Estimating Homography Between the Model Plane and its Image

There are many ways to estimate the homography between the model plane and its image. Here, we present a technique based on maximum likelihood criterion. Let $\mathtt{M}_i$ and $\mathbf{m}_i$ be the model and image points, respectively. Ideally, they should satisfy (1.4.1). In practice, they don't because of noise in the extracted image points. Let's assume that $\mathbf{m}_i$ is corrupted by Gaussian noise with mean $\mathbf{0}$ and covariance matrix $\mathbf{\Lambda}_{\mathbf{m}_i}$. Then, the maximum likelihood estimation of $\mathbf{H}$ is obtained by minimizing the following functional

$$\sum_i (\mathbf{m}_i - \hat{\mathbf{m}}_i)^T \mathbf{\Lambda}_{\mathbf{m}_i}^{-1} (\mathbf{m}_i - \hat{\mathbf{m}}_i) \ ,$$

where $$\hat{\mathbf{m}}_i = \frac{1}{\bar{\mathbf{h}}_3^T \mathtt{M}_i} \begin{bmatrix} \bar{\mathbf{h}}_1^T \mathtt{M}_i \\ \bar{\mathbf{h}}_2^T \mathtt{M}_i \end{bmatrix} \quad \text{with } \bar{\mathbf{h}}_i, \text{ the } i^{\text{th}} \text{ row of } \mathbf{H}.$$

In practice, we simply assume $\mathbf{\Lambda}_{\mathbf{m}_i} = \sigma^2 \mathbf{I}$ for all i. This is reasonable if points are extracted independently with the same procedure. In this case, the above problem becomes a nonlinear least-squares one, i.e., $\min_{\mathbf{H}} \sum_i \|\mathbf{m}_i - \hat{\mathbf{m}}_i\|^2$. The nonlinear minimization is conducted with the Levenberg-Marquardt Algorithm as implemented in `Minpack` [23]. This requires an initial guess, which can be obtained as follows.

Let $\mathbf{x} = [\bar{\mathbf{h}}_1^T, \bar{\mathbf{h}}_2^T, \bar{\mathbf{h}}_3^T]^T$. Then equation (1.4.1) can be rewritten as

$$\begin{bmatrix} \widetilde{\mathtt{M}}^T & \mathbf{0}^T & -u\widetilde{\mathtt{M}}^T \\ \mathbf{0}^T & \widetilde{\mathtt{M}}^T & -v\widetilde{\mathtt{M}}^T \end{bmatrix} \mathbf{x} = \mathbf{0} \ .$$

When we are given n points, we have n above equations, which can be written in matrix equation as $\mathbf{L}\mathbf{x} = \mathbf{0}$, where $\mathbf{L}$ is a $2n \times 9$ matrix. As $\mathbf{x}$ is defined up to a scale factor, the solution is well known to be the right singular vector of $\mathbf{L}$ associated with

the smallest singular value (or equivalently, the eigenvector of $\mathbf{L}^T\mathbf{L}$ associated with the smallest eigenvalue). In $\mathbf{L}$, some elements are constant 1, some are in pixels, some are in world coordinates, and some are multiplication of both. This makes $\mathbf{L}$ poorly conditioned numerically. Much better results can be obtained by performing a simple data normalization prior to running the above procedure.

Bibliography

[1] Y.I. Abdel-Aziz and H.M. Karara. Direct linear transformation into object space coordinates in close-range photogrammetry. In *Proceedings of the Symposium on Close-Range Photogrammetry, University of Illinois at Urbana-Champaign, Urbana, Illinois*, pages 1–18, January 1971.

[2] M. Agrawal and L. Davis. Camera calibration using spheres: A semi-definite programming approach. In *Proceedings of the 9th International Conference on Computer Vision*, pages 782–789, Nice, France, October 2003. IEEE Computer Society Press.

[3] Duane C. Brown. Close-range camera calibration. *Photogrammetric Engineering*, 37(8):855–866, 1971.

[4] Bruno Caprile and Vincent Torre. Using Vanishing Points for Camera Calibration. *The International Journal of Computer Vision*, 4(2):127–140, March 1990.

[5] X. Chen, J. Yang, and A. Waibel. Calibration of a hybrid camera network. In *Proceedings of the 9th International Conference on Computer Vision*, pages 150–155, Nice, France, October 2003. IEEE Computer Society Press.

[6] W. Faig. Calibration of close-range photogrammetry systems: Mathematical formulation. *Photogrammetric Engineering and Remote Sensing*, 41(12):1479–1486, 1975.

[7] O. Faugeras and Q.-T. Luong. *The Geometry of Multiple Images*. The MIT Press, 2001. With contributions from T. Papadopoulo.

[8] Olivier Faugeras. *Three-Dimensional Computer Vision: a Geometric Viewpoint*. MIT Press, 1993.

[9] Olivier Faugeras, Tuan Luong, and Steven Maybank. Camera self-calibration: theory and experiments. In G. Sandini, editor, *Proc 2nd ECCV*, volume 588 of *Lecture Notes in Computer Science*, pages 321–334, Santa Margherita Ligure, Italy, May 1992. Springer-Verlag.

[10] Olivier Faugeras and Giorgio Toscani. The calibration problem for stereo. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 15–20, Miami Beach, FL, June 1986. IEEE.

[11] S. Ganapathy. Decomposition of transformation matrices for robot vision. *Pattern Recognition Letters*, 2:401–412, December 1984.

[12] D. Gennery. Stereo-camera calibration. In *Proceedings of the 10th Image Understanding Workshop*, pages 101–108, 1979.

[13] G.H. Golub and C.F. van Loan. *Matrix Computations*. The John Hopkins University Press, Baltimore, Maryland, 3 edition, 1996.

[14] P. Gurdjos, A. Crouzil, and R. Payrissat. Another way of looking at plane-based calibration: the centre circle constraint. In *Proceedings of the 7th European Conference on Computer Vision*, volume IV, pages 252–266, Copenhagen, May 2002.

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

[16] Richard Hartley. Self-calibration from multiple views with a rotating camera. In J-O. Eklundh, editor, *Proceedings of the 3rd European Conference on Computer Vision*, volume 800-801 of *Lecture Notes in Computer Science*, pages 471–478, Stockholm, Sweden, May 1994. Springer-Verlag.

[17] Richard I. Hartley. An algorithm for self calibration from several views. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 908–912, Seattle, WA, June 1994. IEEE.

[18] J. Heikkilä and O. Silvén. A four-step camera calibration procedure with implicit image correction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1106–1112, San Juan, Puerto Rico, June 1997. IEEE Computer Society.

[19] D. Liebowitz and A. Zisserman. Metric rectification for perspective images of planes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 482–488, Santa Barbara, California, June 1998. IEEE Computer Society.

[20] Q.-T. Luong and O.D. Faugeras. Self-calibration of a moving camera from point correspondences and fundamental matrices. *The International Journal of Computer Vision*, 22(3):261–289, 1997.

[21] Quang-Tuan Luong. *Matrice Fondamentale et Calibration Visuelle sur l'Environnement-Vers une plus grande autonomie des systèmes robotiques*. PhD thesis, Université de Paris-Sud, Centre d'Orsay, December 1992.

[22] S. J. Maybank and O. D. Faugeras. A theory of self-calibration of a moving camera. *The International Journal of Computer Vision*, 8(2):123–152, August 1992.

[23] J.J. More. The levenberg-marquardt algorithm, implementation and theory. In G. A. Watson, editor, *Numerical Analysis*, Lecture Notes in Mathematics 630. Springer-Verlag, 1977.

[24] T. Okatani and K. Deguchi. Autocalibration of projector-screen-camera system: Theory and algorithm for screen-to-camera homography estimation. In *Proceedings of the 9th International Conference on Computer Vision*, pages 774–781, Nice, France, October 2003. IEEE Computer Society Press.

[25] L Robert. Camera calibration without feature extraction. *Computer Vision, Graphics, and Image Processing*, 63(2):314–325, March 1995. also INRIA Technical Report 2204.

[26] J.G. Semple and G.T. Kneebone. *Algebraic Projective Geometry.* Oxford: Clarendon Press, 1952. Reprinted 1979.

[27] S.W. Shih, Y.P. Hung, and W.S. Lin. Accurate linear technique for camera calibration considering lens distortion by solving an eigenvalue problem. *Optical Engineering*, 32(1):138–149, 1993.

[28] I. Shimizu, Z. Zhang, S. Akamatsu, and K. Deguchi. Head pose determination from one image using a generic model. In *Proceedings of the IEEE Third International Conference on Automatic Face and Gesture Recognition*, pages 100–105, Nara, Japan, April 1998.

[29] C. C. Slama, editor. *Manual of Photogrammetry.* American Society of Photogrammetry, fourth edition, 1980.

[30] G. Stein. Accurate internal camera calibration using rotation, with analysis of sources of error. In *Proc. Fifth International Conference on Computer Vision*, pages 230–236, Cambridge, Massachusetts, June 1995.

[31] P. Sturm and S. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 432–437, Fort Collins, Colorado, June 1999. IEEE Computer Society Press.

[32] B. Triggs. Autocalibration from planar scenes. In *Proceedings of the 5th European Conference on Computer Vision*, pages 89–105, Freiburg, Germany, June 1998.

[33] Roger Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, 3(4):323–344, August 1987.

[34] T. Ueshiba and F. Tomita. Plane-based calibration algorithm for multi-camera systems via factorization of homography matrices. In *Proceedings of the 9th International Conference on Computer Vision*, pages 966–973, Nice, France, October 2003. IEEE Computer Society Press.

[35] G.Q. Wei and S.D. Ma. A complete two-plane camera calibration method and experimental comparisons. In *Proc. Fourth International Conference on Computer Vision*, pages 439–446, Berlin, May 1993.

[36] G.Q. Wei and S.D. Ma. Implicit and explicit camera calibration: Theory and experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):469–480, 1994.

[37] J. Weng, P. Cohen, and M. Herniou. Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):965–980, October 1992.

[38] Reg Willson. *Modeling and Calibration of Automated Zoom Lenses*. PhD thesis, Department of Electrical and Computer Engineering, Carnegie Mellon University, 1994.

[39] R. Yang and Z. Zhang. Eye gaze correction with stereovision for video teleconferencing. In *Proceedings of the 7th European Conference on Computer Vision*, volume II, pages 479–494, Copenhagen, May 2002. Also available as Technical Report MSR-TR-01-119.

[40] R. Yang and Z. Zhang. Model-based head pose tracking with stereovision. In *Proc. Fifth IEEE International Conference on Automatic Face and Gesture Recognition (FG2002)*, pages 255–260, Washington, DC, May 2002. Also available as Technical Report MSR-TR-01-102.

[41] Z. Zhang. A flexible new technique for camera calibration. Technical Report MSR-TR-98-71, Microsoft Research, December 1998. Available together with the software at `http://research.microsoft.com/~zhang/Calib/`.

[42] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[43] Z. Zhang. Camera calibration with one-dimensional objects. Technical Report MSR-TR-2001-120, Microsoft Research, December 2001.

[44] Z. Zhang. Camera calibration with one-dimensional objects. In *Proc. European Conference on Computer Vision (ECCV'02)*, volume IV, pages 161–174, Copenhagen, Denmark, May 2002.

[45] Zhengyou Zhang and Olivier D. Faugeras. *3D Dynamic Scene Analysis: A Stereo Based Approach*. Springer, Berlin, Heidelberg, 1992.