Calibration of Ultra-Wide Fisheye Lens Cameras by Eigenvalue Minimization

Kenichi Kanatani, *Fellow, IEEE,*

Abstract—We present a new technique for calibrating ultra-wide fisheye lens cameras by imposing the constraint that collinear points be rectified to be collinear, parallel lines to be parallel, and orthogonal lines to be orthogonal. Exploiting the fact that line fitting reduces to an eigenvalue problem in 3D, we do a rigorous perturbation analysis to obtain a practical calibration procedure. Doing experiments, we point out that spurious solutions exist if collinearity and parallelism alone are imposed. Our technique has many desirable properties. For example, no metric information is required about the reference pattern or the camera position, and separate stripe patterns can be displayed on a video screen to generate a virtual grid, eliminating the grid point extraction processing.

Index Terms—Fisheye lens, camera calibration, eigenvalue minimization, perturbation theorem, perspective rectification.

1 Introduction

Fisheye lens cameras are widely used for surveillance purposes because of their wide angles of view. They are also mounted on vehicles for various purposes including obstacle detection, self-localization, and bird's eye view generation [10], [13]. However, fisheye lens images have strong distortion, so that in order to apply the computer vision techniques accumulated in the past decades, one first needs to rectify the image into a perspective view. Already, there is a lot of literature for this [2], [5], [7], [8], [11], [12], [13], [14], [18], [19].

The standard approach is to place a reference grid plane and match the image with the reference, whose precise geometry is assumed to be known [2], [4], [5], [7], [19]. However, this approach is not very practical for recently popularized ultra-wide fisheye lenses, because they cover more than 180 degree angles of view and hence any (even infinite) reference plane cannot cover the entire field of view. This difficulty can be circumvented by using the *collinearity constraint* pointed out repeatedly, first by Onodera and Kanatani [15] in 1992, later by Swaminathan and Nayar [18] in 2000, and by Devernay and Faugeras [1] in 2001. They pointed out that camera calibration can be done by imposing the constraint that straight lines be rectified to be straight. This principle was applied to fisheye lenses by Nakano, et al. [12], Kase et al. [8], and Okutsu et al. [13]. Komagata et al. [9] further introduced the *parallelism constraint* and the *orthogonality constraint*, requiring that parallel lines be rectified to be parallel and orthogonal lines to be orthogonal. However, the cost function has been directly minimized by brute force means such as the Brent method and the Powell method [17].

• K. Kanatani is with the Department of Computer Science, Okayama University, Okayama, Japan 700-8530.
E-mail: kanatani@suri.cs.okayama-u.ac.jp

Manuscript received 6 Mar. 2011; revised 5 Feb. 2012; accepted 4 July 2012; published on line 11 July 2012.
Recommended for acceptance by A. Fitzgibbon.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2011-03-0417.
Digital Object Identifier no. 10.1109/TPAMI.2012.146.

In this paper, we adopt the collinearity-parallelism-orthogonality constraint of Komagata et al. [9] and optimize it by *eigenvalue minimization*. The fact that imposing collinearity implies eigenvalue minimization and that the optimization can be done by invoking the perturbation theorem was pointed out by Onodera and Kanatani [15]. Using this principle, they rectified perspective images by gradient descent. Here, we apply their principle to ultra-wide fisheye lens calibration.

The first contribution of this paper is to demonstrate the usefulness of the eigenvalue minimization principle, which has not been known in the past collinearity-based work [1], [8], [12], [13], [18]. The second contribution is to point out that the orthogonality constraint plays an essential role, showing by experiments that a *spurious solution* exists if only collinearity and parallelism are imposed. This fact has not been known in the past collinearity-based work, either.

For data acquisition, we take images of stripes of different orientations on a large-screen display by placing the camera in various positions. Like the past collinearity-based methods, our method is *non-metric* in the sense that no metric information is required about the camera position or the reference pattern. Yet, many researchers pointed out the necessity of some auxiliary information. For example, Nakano et al. [11], [12] proposed vanishing point estimation using conic fitting to straight line images (recently, Hughes et al. [5] proposed this same technique again). Okutsu et al. [14] picked out the images of antipodal points by hand. Such auxiliary information may be useful to suppress spurious solutions. As we show, however, accurate calibration is possible without any auxiliary information by our eigenvalue minimization using the collinearity-parallelism-orthogonality constraint.

This paper is organized as follows. In Section 2, we describe our imaging geometry model. Section 3 gives derivative expressions of the fundamental quantities, followed by a detailed perturbation analysis of the collinearity constraint in Section 4, of the parallelism constraint in Section 5, and of the orthogonality constraint in Section 6. Section 7 shows

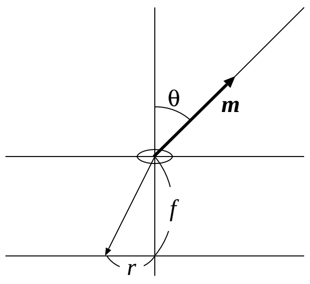


Fig. 1. The imaging geometry of a fisheye lens and the incident ray vector $\mathbf{m}$.

our experiments of the proposed non-metric technique, using stripe images on a video display. We point out that a spurious solution arises if only collinearity and parallelism are imposed and that it can be eliminated without using any auxiliary information if orthogonality is introduced. We compare our calibration results with those of an alternative and also show real scene application examples. In Section 8, we conclude.

2 Geometry of Fisheye Lens Imaging

We consider recently popularized ultra-wide fisheye lenses with the imaging geometry modeled by the *stereographic projection*

$$r = 2f \tan\frac{\theta}{2}, \tag{1}$$

where θ is the incidence angle (the angle of the incident ray of light from the optical axis) and r (in pixels) is the distance of the corresponding image point from the principal point (Fig. 1). The constant f is called the *focal length*. We consider (1) merely because our camera is as such, but the following calibration procedure is identical whatever model is used.

The value f provided by the manufacturer may not be exact, and the principal point may not be at the center of the image frame. After all, (1) is an idealization; a real lens may not exactly satisfy it. So, we generalize (1) into the form

$$\frac{r}{f_0} + a_1\Big(\frac{r}{f_0}\Big)^3 + a_2\Big(\frac{r}{f_0}\Big)^5 + \cdots = \frac{2f}{f_0}\tan\frac{\theta}{2}, \tag{2}$$

and determine the values of f, a_1, a_2, ... along with the principal point position. Here, f_0 is a scale constant to keep the powers r^k within a reasonable numerical range (in our experiment, we used the value f_0 = 150 pixels). The linear term r/f_0 has no coefficient because f on the right side is an unknown parameter. Letting $a_1 = a_2 = \cdots = 0$ corresponds to the stereographic projection. Since a sufficient number of correction terms could approximate any function, the right side of (2) could be any function of θ, e.g., the perspective projection model $(f/f_0)\tan\theta$ or the equidistance projection model $(f/f_0)\theta$. We adopt the stereographic projection model merely for the ease of initialization.

In (2), even power terms do not exist, because the lens has circular symmetry; r is an odd function of θ. We assume that the azimuthal angle of the projection is equal to that of the incident ray. In the past, these two were often assumed to be slightly different, and geometric correction of the resulting "tangential distortion" was studied. Currently, the lens manufacturing technology is of sufficiently high levels so that the tangential distortion can be safely ignored. If not, we can simply include the tangential distortion terms in (2), and the subsequent calibration procedure remains unchanged.

In the literature, the model of the form $r = c_1\theta + c_2\theta^3 + c_3\theta^5 + \cdots$ is frequently assumed [7], [8], [13], [12]. As we see shortly, however, the value of θ for a specified r is necessary in each step of the optimization iterations. So, many authors computed θ by solving a polynomial equation using a numerical means [7], [8], [13], [12], but this causes loss of accuracy and efficiency. It is more convenient to express θ in terms of r from the beginning. From (2), the expression of θ is given by

$$\theta = 2\tan^{-1}\Big(\frac{f_0}{2f}\Big(\frac{r}{f_0} + a_1\Big(\frac{r}{f_0}\Big)^3 + a_2\Big(\frac{r}{f_0}\Big)^5 + \cdots\Big)\Big). \tag{3}$$

3 Incident Ray Vector

Let $\mathbf{m}$ be the unit vector in the direction of the incident ray of light (Fig. 1); we call $\mathbf{m}$ the *incident ray vector*. In polar coordinates, it has the expression

$$\mathbf{m} = \begin{pmatrix} \sin\theta\cos\phi \\ \sin\theta\sin\phi \\ \cos\theta \end{pmatrix}, \tag{4}$$

where θ is the incidence angle from the Z-axis and ϕ is the azimuthal angle from the X-axis. Since ϕ, by our assumption, equals the azimuthal angle on the image plane, the point (x, y) on which the incident light focuses is specified by

$$x = u_0 + r\cos\phi, \qquad y = v_0 + r\sin\phi,$$
$$r = \sqrt{(x-u_0)^2 + (y-v_0)^2}, \tag{5}$$

where (u_0, v_0) is the principal point. Hence, (4) is rewritten as

$$\mathbf{m} = \begin{pmatrix} ((x-u_0)/r)\sin\theta \\ ((y-v_0)/r)\sin\theta \\ \cos\theta \end{pmatrix}. \tag{6}$$

Using $\partial r/\partial u_0$ and $\partial r/\partial v_0$ obtained from (5), we obtain the derivatives of (6) with respect to u_0 and v_0 in the form

$$\begin{aligned}
\frac{\partial \mathbf{m}}{\partial u_0} &= \begin{pmatrix} -1/r + (x-u_0)^2/r^3 \\ (x-u_0)(y-v_0)/r^3 \\ 0 \end{pmatrix}\sin\theta \\
&\quad + \begin{pmatrix} (x-u_0)/r)\cos\theta \\ ((y-v_0)/r)\cos\theta \\ -\sin\theta \end{pmatrix}\frac{\partial\theta}{\partial u_0}, \\
\frac{\partial \mathbf{m}}{\partial v_0} &= \begin{pmatrix} (x-u_0)(y-v_0)/r^3 \\ -1/r + (y-v_0)^2/r^3 \\ 0 \end{pmatrix}\sin\theta \\
&\quad + \begin{pmatrix} ((x-u_0)/r)\cos\theta \\ ((y-v_0)/r)\cos\theta \\ -\sin\theta \end{pmatrix}\frac{\partial\theta}{\partial v_0}.
\end{aligned} \tag{7}$$

Differentiating (3) with respect to u_0 and v_0 on both sides and rearranging the result, we obtain $\partial\theta/\partial u_0$ and $\partial\theta/\partial v_0$ in the form

$$\frac{\partial\theta}{\partial u_0} = -\frac{1}{f}\cos^2\frac{\theta}{2}\Big(1+\sum_{k=1}^{\infty}(2k-1)a_k\Big(\frac{r}{f_0}\Big)^{2k}\Big)\frac{x-u_0}{r},$$
$$\frac{\partial\theta}{\partial v_0} = -\frac{1}{f}\cos^2\frac{\theta}{2}\Big(1+\sum_{k=1}^{\infty}(2k-1)a_k\Big(\frac{r}{f_0}\Big)^{2k}\Big)\frac{y-v_0}{r}. \tag{8}$$

Next, we consider derivation with respect to f. Differentiating (2) with respect to f on both sides, we obtain

$$\frac{\partial\theta}{\partial f} = -\frac{2}{f}\sin\frac{\theta}{2}\cos\frac{\theta}{2} = -\frac{1}{f}\sin\theta. \tag{9}$$

It follows that the derivative of (6) with respect to f is

$$\frac{\partial\mathbf{m}}{\partial f} = -\frac{1}{f}\sin\theta\begin{pmatrix}((x-u_0)/r)\cos\theta\\((y-v_0)/r)\cos\theta\\-\sin\theta\end{pmatrix}. \tag{10}$$

Finally, we consider derivation with respect to a_k. Differentiating (2) with respect to a_k on both sides, we obtain

$$\frac{\partial\theta}{\partial a_k} = \frac{f_0}{f}\Big(\frac{r}{f_0}\Big)^{2k+1}\cos^2\frac{\theta}{2}. \tag{11}$$

It follows that the derivation of (6) with respect to a_k is

$$\frac{\partial\mathbf{m}}{\partial a_k} = \frac{f_0}{f}\Big(\frac{r}{f_0}\Big)^{2k+1}\cos^2\frac{\theta}{2}\begin{pmatrix}((x-u_0)/r)\cos\theta\\((y-v_0)/r)\cos\theta\\-\sin\theta\end{pmatrix} \tag{12}$$

As we now show, *all the cost functions in the subsequent optimization are expressed in terms of the incident ray vector* $\mathbf{m}$. Hence, the derivatives of any cost function with respect to any parameter can by evaluated simply by combining the above expressions. This fact is the *core* of our eigenvalue minimization principle.

4 Collinearity Constraint

We first show that, unlike existing collinearity-based work [1], [8], [12], [13], [18], the collinearity constraint can be imposed *without fitting lines in the image plane*; all we need to do is *3D scene analysis* in terms of the incident ray vector $\mathbf{m}$.

Suppose we observe a collinear point sequence $\mathcal{S}_\kappa$ (the subscript κ enumerates all existing sequences) consisting of N points p_1, ..., p_N, and let $\mathbf{m}_1$,..., $\mathbf{m}_N$ be their incident ray vectors. If the camera is precisely calibrated, the computed incident ray vectors should be coplanar. Hence, if $\mathbf{n}_\kappa$ is the unit normal to the plane passing through the origin O (lens center) and $\mathcal{S}_\kappa$, we should have $(\mathbf{n}_\kappa, \mathbf{m}_\alpha)$ = 0, α = 1, ..., N (Fig. 2). In the following, we denote the inner product of vectors $\mathbf{a}$ and $\mathbf{b}$ by $(\mathbf{a}, \mathbf{b})$. If the calibration is incomplete, however, $(\mathbf{n}_\kappa, \mathbf{m}_\alpha)$ may not be strictly zero. So, we adjust the parameters by minimizing

$$\sum_{\alpha\in\mathcal{S}_\kappa}(\mathbf{n}_\kappa, \mathbf{m}_\alpha)^2 = \sum_{\alpha\in\mathcal{S}_\kappa}\mathbf{n}_\kappa^\top\mathbf{m}_\alpha\mathbf{m}_\alpha^\top\mathbf{n}_\kappa$$
$$= (\mathbf{n}_\kappa, \sum_{\alpha\in\mathcal{S}_\kappa}\mathbf{m}_\alpha\mathbf{m}_\alpha^\top\mathbf{n}_\kappa) = (\mathbf{n}_\kappa, \mathbf{M}^{(\kappa)}\mathbf{n}_\kappa), \tag{13}$$

where we define

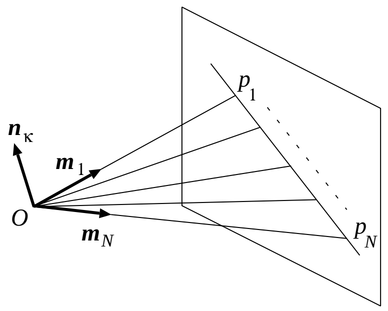


Fig. 2. The incident ray vectors $\mathbf{m}_\alpha$ of collinear points p_1, ..., p_N are coplanar.

$$\mathbf{M}^{(\kappa)} = \sum_{\alpha\in\mathcal{S}_\kappa}\mathbf{m}_\alpha\mathbf{m}_\alpha^\top. \tag{14}$$

Since (13) is a quadratic form of $\mathbf{M}^{(\kappa)}$, its minimum equals the smallest eigenvalue $\lambda^{(\kappa)}_{\min}$ of $\mathbf{M}^{(\kappa)}$, $\mathbf{n}_\kappa$ being the corresponding unit eigenvector. Thus, for enforcing the collinearity constraint for all collinear point sequences $\mathcal{S}_\kappa$, we only need to determine the parameters that minimize

$$J_1 = \sum_{\text{all }\kappa}\lambda^{(\kappa)}_{\min}, \tag{15}$$

which is the origin of the term "eigenvalue minimization."

We now evaluate the first and second derivatives of $\lambda^{(\kappa)}_{\min}$ with respect to c, which represents the parameters u_0, v_0, f, a_1, a_2, ... In the past, many authors used finite difference approximation for derivatives or resorted to brute force optimization schemes, such as the Brent method and the Powell method [17], that do not require derivatives [1], [7], [8], [9], [11], [12], [13], [14], [18]. This might perhaps be because many people thought that eigenvalues cannot be differentiated in analytical terms. As pointed out by Onodera and Kanatani [15], Kanatani [6], and Papadopoulo and Lourakis [16], however, analytical expressions of the derivatives of eigenvalues and singular values are easily obtained. We begin with first derivatives.

4.1 First derivatives

Differentiating the defining equation

$$\mathbf{M}^{(\kappa)}\mathbf{n}_\kappa = \lambda^{(\kappa)}_{\min}\mathbf{n}_\kappa \tag{16}$$

with respect to c on both sides, we have

$$\frac{\partial\mathbf{M}^{(\kappa)}}{\partial c}\mathbf{n}_\kappa + \mathbf{M}^{(\kappa)}\frac{\partial\mathbf{n}_\kappa}{\partial c} = \frac{\partial\lambda^{(\kappa)}_{\min}}{\partial c}\mathbf{n}_\kappa + \lambda^{(\kappa)}_{\min}\frac{\partial\mathbf{n}_\kappa}{\partial c}. \tag{17}$$

Computing the inner product with $\mathbf{n}_\kappa$ on both sides, we obtain

$$(\mathbf{n}_\kappa, \frac{\partial\mathbf{M}^{(\kappa)}}{\partial c}\mathbf{n}_\kappa) + (\mathbf{n}_\kappa, \mathbf{M}^{(\kappa)}\frac{\partial\mathbf{n}_\kappa}{\partial c})$$
$$= \frac{\partial\lambda^{(\kappa)}_{\min}}{\partial c}(\mathbf{n}_\kappa, \mathbf{n}_\kappa) + \lambda^{(\kappa)}_{\min}(\mathbf{n}_\kappa, \frac{\partial\mathbf{n}_\kappa}{\partial c}). \tag{18}$$

Since $\mathbf{n}_\kappa$ is a unit vector, we have $(\mathbf{n}_\kappa, \partial\mathbf{n}_\kappa/\partial c)$ = 0, because variations of a unit vector should be orthogonal to itself. The matrix $\mathbf{M}^{(\kappa)}$ is symmetric, so we have $(\mathbf{n}_\kappa, \mathbf{M}^{(\kappa)}\partial\mathbf{n}_\kappa/\partial c)$ = $(\mathbf{M}^{(\kappa)}\mathbf{n}_\kappa, \partial\mathbf{n}_\kappa/\partial c)$ = $\lambda^{(\kappa)}_{\min}(\mathbf{n}_\kappa, \partial\mathbf{n}_\kappa/\partial c)$ = 0. Thus, (18) implies

$$\frac{\partial \lambda_{\rm min}^{(\kappa)}}{\partial c} = (\mathbf{n}_\kappa, \frac{\partial \mathbf{M}^{(\kappa)}}{\partial c}\mathbf{n}_\kappa). \tag{19}$$

This result is well known as the *perturbation theorem* of eigenvalue problems [6]. From the definition of $\mathbf{M}^{(\kappa)}$ in (14), we see that

$$\begin{aligned}\frac{\partial \mathbf{M}^{(\kappa)}}{\partial c} &= \sum_{\alpha=1}^{N}\Big(\frac{\partial \mathbf{m}_\alpha}{\partial c}\mathbf{m}_\alpha^\top + \mathbf{m}_\alpha\Big(\frac{\partial \mathbf{m}_\alpha}{\partial c}\Big)^\top\Big) \\ &= 2\mathcal{S}[\sum_{\alpha=1}^{N}\frac{\partial \mathbf{m}_\alpha}{\partial c}\mathbf{m}_\alpha^\top] \equiv \mathbf{M}_c^{(\kappa)},\end{aligned} \tag{20}$$

where $\mathcal{S}[\,\cdot\,]$ denotes symmetrization ($\mathcal{S}[\mathbf{A}] = (\mathbf{A}+\mathbf{A}^\top)/2$). Thus, the first derivatives of the function J_1 with respect to $c = u_0$, v_0, f, a_1, a_2, ... are given as follows:

$$\frac{\partial J_1}{\partial c} = \sum_{\text{all }\kappa}(\mathbf{n}_\kappa, \mathbf{M}_c^{(\kappa)}\mathbf{n}_\kappa). \tag{21}$$

4.2 Second derivatives

Differentiating (19) with respect to c' (= u_0, v_0, f, a_1, a_2, ...), we obtain

$$\begin{aligned}\frac{\partial^2 \lambda_{\rm min}^{(\kappa)}}{\partial c\partial c'} &= (\frac{\partial \mathbf{n}_\kappa}{\partial c'}, \mathbf{M}_c^{(\kappa)}\mathbf{n}_\kappa) + (\mathbf{n}_\kappa, \frac{\partial^2 \mathbf{M}^{(\kappa)}}{\partial c\partial c'}\mathbf{n}_\kappa) \\ &\quad + (\mathbf{n}_\kappa, \mathbf{M}_c^{(\kappa)}\frac{\partial \mathbf{n}_\kappa}{\partial c'}) \\ &= (\mathbf{n}_\kappa, \frac{\partial^2 \mathbf{M}^{(\kappa)}}{\partial c\partial c'}\mathbf{n}_\kappa) + 2(\frac{\partial \mathbf{n}_\kappa}{\partial c'}, \mathbf{M}_c^{(\kappa)}\mathbf{n}_\kappa).\end{aligned} \tag{22}$$

First, consider the first term. Differentiation of (20) with respect to c' is

$$\frac{\partial^2 \mathbf{M}^{(\kappa)}}{\partial c\partial c'} = 2\mathcal{S}[\sum_{\alpha=1}^{N}\Big(\frac{\partial^2 \mathbf{m}_\alpha}{\partial c\partial c'}\mathbf{m}_\alpha^\top + \frac{\partial \mathbf{m}_\alpha}{\partial c}\Big(\frac{\partial \mathbf{m}_\alpha}{\partial c'}\Big)^\top\Big)]. \tag{23}$$

Hence, we have

$$\begin{aligned}(\mathbf{n}_\kappa, \frac{\partial^2 \mathbf{M}^{(\kappa)}}{\partial c\partial c'}\mathbf{n}_\kappa) &= 2\sum_{\alpha=1}^{N}\Big((\mathbf{n}_\kappa, \frac{\partial^2 \mathbf{m}_\alpha}{\partial c\partial c'})(\mathbf{m}_\alpha, \mathbf{n}_\kappa) \\ &\quad + (\mathbf{n}_\kappa, \frac{\partial \mathbf{m}_\alpha}{\partial c})(\frac{\partial \mathbf{m}_\alpha}{\partial c'}, \mathbf{n}_\kappa)\Big).\end{aligned} \tag{24}$$

If the calibration is complete, we should have $(\mathbf{m}_\alpha, \mathbf{n}_\kappa) = 0$. In the course of the optimization, we can expect that $(\mathbf{m}_\alpha, \mathbf{n}_\kappa) \approx 0$. Hence, (24) can be approximated by

$$\begin{aligned}(\mathbf{n}_\kappa, \frac{\partial^2 \mathbf{M}^{(\kappa)}}{\partial c\partial c'}\mathbf{n}_\kappa) &\approx 2\sum_{\alpha=1}^{N}(\mathbf{n}_\kappa, \frac{\partial \mathbf{m}_\alpha}{\partial c})(\frac{\partial \mathbf{m}_\alpha}{\partial c'}, \mathbf{n}_\kappa) \\ &= 2(\mathbf{n}_\kappa, \mathbf{M}_{cc'}^{(\kappa)}\mathbf{n}_\kappa),\end{aligned} \tag{25}$$

$$\mathbf{M}_{cc'}^{(\kappa)} \equiv \sum_{\alpha=1}^{N}\Big(\frac{\partial \mathbf{m}_\alpha}{\partial c}\Big)\Big(\frac{\partial \mathbf{m}_\alpha}{\partial c'}\Big)^\top. \tag{26}$$

This is a sort of the Gauss-Newton approximation.

Next, consider the second term of (22). Because $\mathbf{n}_\kappa$ is a unit vector, its variations are orthogonal to itself. Let $\lambda_1^{(\kappa)} \geq \lambda_2^{(\kappa)} \geq \lambda_{\rm min}^{(\kappa)}$ be the eigenvalues of $\mathbf{M}^{(\kappa)}$ with $\mathbf{n}_{\kappa 1}$, $\mathbf{n}_{\kappa 2}$, and $\mathbf{n}_\kappa$ the corresponding unit eigenvectors. Since the eigenvectors of a symmetric matrix are mutually orthogonal, any vector orthogonal to $\mathbf{n}_\kappa$ is expressed as a linear combination of $\mathbf{n}_{\kappa 1}$ and $\mathbf{n}_{\kappa 2}$. Hence, we can write

$$\frac{\partial \mathbf{n}_\kappa}{\partial c} = \beta_1\mathbf{n}_{\kappa 1} + \beta_2\mathbf{n}_{\kappa 2}, \tag{27}$$

for some β_1 and β_2. Substituting (19) and (27) into (17) and noting that $\mathbf{M}^{(\kappa)}\mathbf{n}_{\kappa 1} = \lambda_1^{(\kappa)}\mathbf{n}_{\kappa 1}$ and $\mathbf{M}^{(\kappa)}\mathbf{n}_{\kappa 2} = \lambda_2^{(\kappa)}\mathbf{n}_{\kappa 2}$, we obtain

$$\begin{aligned}&\beta_1(\lambda_1^{(\kappa)} - \lambda_{\rm min}^{(\kappa)})\mathbf{n}_{\kappa 1} + \beta_2(\lambda_2^{(\kappa)} - \lambda_{\rm min}^{(\kappa)})\mathbf{n}_{\kappa 2} \\ &= (\mathbf{n}_\kappa, \mathbf{M}_c^{(\kappa)}\mathbf{n}_\kappa)\mathbf{n}_\kappa - \mathbf{M}_c^{(\kappa)}\mathbf{n}_\kappa.\end{aligned} \tag{28}$$

Computing the inner product with $\mathbf{n}_{\kappa 1}$ and $\mathbf{n}_{\kappa 2}$ on both sides, we obtain

$$\begin{aligned}\beta_1(\lambda_1^{(\kappa)} - \lambda_{\rm min}^{(\kappa)}) &= -(\mathbf{n}_{\kappa 1}, \mathbf{M}_c^{(\kappa)}\mathbf{n}_\kappa), \\ \beta_2(\lambda_2^{(\kappa)} - \lambda_{\rm min}^{(\kappa)}) &= -(\mathbf{n}_{\kappa 2}, \mathbf{M}_c^{(\kappa)}\mathbf{n}_\kappa),\end{aligned} \tag{29}$$

from which β_1 and β_2 are determined. Hence, (27) is written as follows:

$$\frac{\partial \mathbf{n}_\kappa}{\partial c} = -\frac{(\mathbf{n}_{\kappa 1}, \mathbf{M}_c^{(\kappa)}\mathbf{n}_\kappa)\mathbf{n}_{\kappa 1}}{\lambda_1^{(\kappa)} - \lambda_{\rm min}^{(\kappa)}} - \frac{(\mathbf{n}_{\kappa 2}, \mathbf{M}_c^{(\kappa)}\mathbf{n}_\kappa)\mathbf{n}_{\kappa 2}}{\lambda_2^{(\kappa)} - \lambda_{\rm min}^{(\kappa)}}. \tag{30}$$

This is also a well known result of the perturbation theorem of eigenvalue problems [6]. Thus, the second term of (22) can be written as

$$\begin{aligned}2(\frac{\partial \mathbf{n}_\kappa}{\partial c'}, \mathbf{M}_c^{(\kappa)}\mathbf{n}_\kappa) &= -\frac{2(\mathbf{n}_{\kappa 1}, \mathbf{M}_c^{(\kappa)}\mathbf{n}_\kappa)(\mathbf{n}_{\kappa 1}, \mathbf{M}_{c'}^{(\kappa)}\mathbf{n}_\kappa)}{\lambda_1^{(\kappa)} - \lambda_{\rm min}^{(\kappa)}} \\ &\quad - \frac{2(\mathbf{n}_{\kappa 2}, \mathbf{M}_c^{(\kappa)}\mathbf{n}_\kappa)(\mathbf{n}_{\kappa 2}, \mathbf{M}_{c'}^{(\kappa)}\mathbf{n}_\kappa)}{\lambda_2^{(\kappa)} - \lambda_{\rm min}^{(\kappa)}}.\end{aligned} \tag{31}$$

Combining (25) and (31), we can evaluate (22). Thus, the second derivatives of the function J_1 with respect to c and c' are given by

$$\begin{aligned}\frac{\partial^2 J_1}{\partial c\partial c'} &= 2\sum_{\text{all }\kappa}\Big((\mathbf{n}_\kappa, \mathbf{M}_{cc'}^{(\kappa)}\mathbf{n}_\kappa) \\ &\quad - \sum_{i=1}^{2}\frac{(\mathbf{n}_{\kappa i}, \mathbf{M}_c^{(\kappa)}\mathbf{n}_\kappa)(\mathbf{n}_{\kappa i}, \mathbf{M}_{c'}^{(\kappa)}\mathbf{n}_\kappa)}{\lambda_i^{(\kappa)} - \lambda_{\rm min}^{(\kappa)}}\Big).\end{aligned} \tag{32}$$

5 Parallelism Constraint

We next show that the parallelism constraint can also be imposed by a direct 3D scene analysis. Let $\mathcal{G}_g$ be a group of parallel collinear point sequences (the subscript g enumerates all existing groups) with a common orientation $\mathbf{l}_g$ (unit vector). The normals $\mathbf{n}_\kappa$ to the planes passing through the origin O (lens center) and lines of $\mathcal{G}_g$ are all orthogonal to $\mathbf{l}_g$ (Fig. 3). Hence, we should have $(\mathbf{l}_g, \mathbf{n}_\kappa) = 0$, $\kappa \in \mathcal{G}_g$, if the calibration is complete. So, we adjust the parameters by minimizing

$$\begin{aligned}\sum_{\kappa\in\mathcal{G}_g}(\mathbf{l}_g, \mathbf{n}_\kappa)^2 &= \sum_{\kappa\in\mathcal{G}_g}\mathbf{l}_g^\top\mathbf{n}_\kappa\mathbf{n}_\kappa^\top\mathbf{l}_g \\ &= (\mathbf{l}_g, \sum_{\kappa\in\mathcal{G}_g}\mathbf{n}_\kappa\mathbf{n}_\kappa^\top\mathbf{l}_g) = (\mathbf{l}_g, \mathbf{N}^{(g)}\mathbf{l}_g),\end{aligned} \tag{33}$$

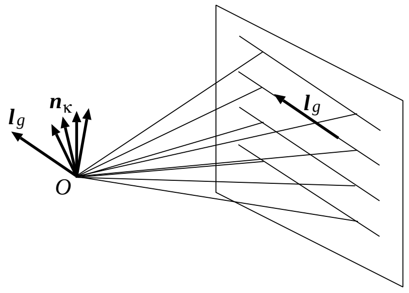


Fig. 3. The surface normals $\mathbf{n}_\kappa$ to the planes defined by parallel lines are orthogonal to the common direction $\mathbf{l}_g$ of the lines.

where we define

$$\mathbf{N}^{(g)} = \sum_{\kappa \in \mathcal{G}_g} \mathbf{n}_\kappa \mathbf{n}_\kappa^\top. \tag{34}$$

Since (33) is a quadratic form of $\mathbf{N}^{(g)}$, its minimum equals the smallest eigenvalue $\mu_{\min}^{(g)}$ of $\mathbf{N}^{(g)}$, $\mathbf{l}_g$ being the corresponding unit eigenvector. To enforce the parallelism constraint for all groups of parallel collinear sequences, we determine the parameters that minimize

$$J_2 = \sum_{\text{all } g} \mu_{\min}^{(g)}. \tag{35}$$

5.1 First derivatives

Doing the same perturbation analysis as in Section 4, we obtain the first derivatives of the function J_2 with respect to parameters c in the form

$$\frac{\partial J_2}{\partial c} = \sum_{\text{all } g} (\mathbf{l}_g, \mathbf{N}_c^{(g)} \mathbf{l}_g), \quad \mathbf{N}_c^{(g)} = 2\mathcal{S}[\sum_{\kappa \in \mathcal{G}_g} \frac{\partial \mathbf{n}_\kappa}{\partial c} \mathbf{n}_\kappa^\top], \tag{36}$$

where $\partial \mathbf{n}_\kappa / \partial c$ is given by (30).

5.2 Second derivatives

Doing the same perturbation analysis as in Section 4, we obtain the second derivatives of the function J_2 with respect to parameters c and c' in the form

$$\frac{\partial^2 J_2}{\partial c \partial c'} = 2 \sum_{\text{all } g} \Big((\mathbf{l}_g, \mathbf{N}_{cc'}^{(g)} \mathbf{l}_g) - \sum_{i=1}^{2} \frac{(\mathbf{l}_{gi}, \mathbf{N}_c^{(g)} \mathbf{l}_g)(\mathbf{l}_{gi}, \mathbf{N}_{c'}^{(g)} \mathbf{l}_g)}{\mu_i^{(g)} - \mu_{\min}^{(g)}} \Big), \tag{37}$$

$$\mathbf{N}_{cc'}^{(g)} \equiv \sum_{\kappa \in \mathcal{G}_g} \Big(\frac{\partial \mathbf{n}_\kappa}{\partial c} \Big) \Big(\frac{\partial \mathbf{n}_\kappa}{\partial c'} \Big)^\top, \tag{38}$$

where $\mu_i^{(g)}$, i = 1, 2, are the first and the second largest eigenvalues of the matrix $\mathbf{N}^{(g)}$ and $\mathbf{l}_{gi}$ are the corresponding unit eigenvectors.

6 Orthogonality Constraint

The orthogonality constraint is also easily imposed by a direct 3D scene analysis. Suppose we observe two groups $\mathcal{G}_g$ and $\mathcal{G}_{g'}$ of parallel line sequences with mutually orthogonal directions $\mathbf{l}_g$ and $\mathbf{l}_{g'}$ (Fig. 4). The orientation $\mathbf{l}_g$ of the sequences in the group $\mathcal{G}_g$ is the unit eigenvector of the matrix $\mathbf{N}^{(g)}$ in (34) for the smallest eigenvalue. The same holds for $\mathbf{l}_{g'}$, too. If the calibration is complete, we should have $(\mathbf{l}_g, \mathbf{l}_{g'})$ = 0, so we adjust the parameters by minimizing

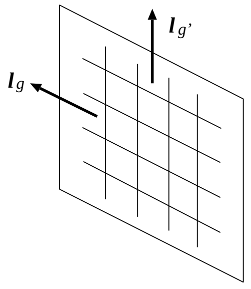


Fig. 4. If two sets of parallel lines make right angles, their directions $\mathbf{l}_g$ and $\mathbf{l}_{g'}$ are orthogonal to each other.

$$J_3 = \sum_{\substack{\text{all orthogonal} \\ \text{pairs } \{\mathcal{G}_g,\, \mathcal{G}'_g\}}} (\mathbf{l}_g, \mathbf{l}_{g'})^2. \tag{39}$$

6.1 First derivatives

The first derivatives of the function J_3 with respect to parameters c are given by

$$\frac{\partial J_3}{\partial c} = 2 \sum_{\substack{\text{all orthogonal} \\ \text{pairs } \{\mathcal{G}_g,\, \mathcal{G}'_g\}}} (\mathbf{l}_g, \mathbf{l}_{g'}) \Big((\frac{\partial \mathbf{l}_g}{\partial c}, \mathbf{l}_{g'}) + (\mathbf{l}_g, \frac{\partial \mathbf{l}_{g'}}{\partial c}) \Big). \tag{40}$$

The first derivative $\partial \mathbf{l}_g / \partial c$ is given by

$$\frac{\partial \mathbf{l}_g}{\partial c} = -\sum_{i=1}^{2} \frac{(\mathbf{l}_{gi}, \mathbf{N}_c^{(g)} \mathbf{l}_g) \mathbf{l}_{gi}}{\mu_i^{(g)} - \mu_{\min}^{(g)}}, \tag{41}$$

and $\partial \mathbf{l}_{g'} / \partial c$ similarly.

6.2 Second derivatives

Using the Gauss-Newton approximation $(\mathbf{l}_g, \mathbf{l}_{g'}) \approx 0$, we obtain the second derivatives of the function J_3 with respect to parameters c and c' in the form

$$\frac{\partial^2 J_3}{\partial c \partial c'} = 2 \sum_{\substack{\text{all orthogonal} \\ \text{pairs } \{\mathcal{G}_g,\, \mathcal{G}'_g\}}} \Big((\frac{\partial \mathbf{l}_g}{\partial c}, \mathbf{l}_{g'}) + (\mathbf{l}_g, \frac{\partial \mathbf{l}_{g'}}{\partial c}) \Big) \Big((\frac{\partial \mathbf{l}_g}{\partial c'}, \mathbf{l}_{g'}) + (\mathbf{l}_g, \frac{\partial \mathbf{l}_{g'}}{\partial c'}) \Big). \tag{42}$$

7 Experiments

7.1 Optimization procedure

To incorporate all of the collinearity, parallelism, and orthogonality constraints, we minimize

$$J = \frac{J_1}{\gamma_1} + \frac{J_2}{\gamma_2} + \frac{J_3}{\gamma_3}, \tag{43}$$

where γ_i, i = 1, 2, 3, are the weights to balance the magnitudes of the three terms. Note that $J_1 \gg J_2 \gg J_3$, since J_1 is proportional to the number of all points, J_2 to the number of all lines, and J_3 to the number of orthogonal pairs of parallel lines. In our experiment, we used as γ_i the initial value of J_i, so that J is initially $1 + 1 + 1$ = 3.

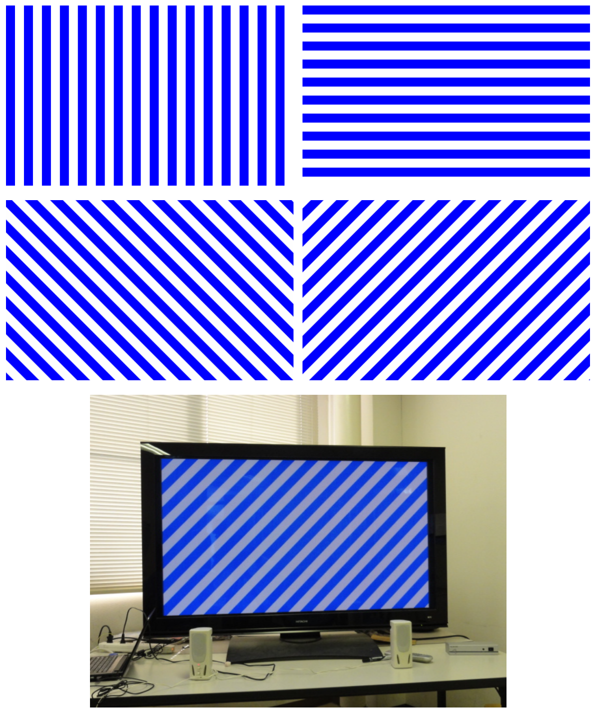


Fig. 5. Stripe patterns in four directions are displayed on a large video screen.

Now that we have derived the first and second derivatives of all J_i with respect to all the parameters, we can combine them into the Levenberg-Marquardt (LM) procedure [17], which search for the minimum of J in the $(K+4)$-dimensional parameter space of u_0, v_0, f, a_0, a_1, ..., a_K. Here, K, which we call the *correction degree*, is the number of the terms on the left hand side of (2), meaning that the left side of (2) is approximated by a $(2K+1)$th degree polynomial. In the past, the number of correction terms was limited to a small number, typically three. This is partly because optimization often involved heuristic means such as finite difference approximation of derivatives. Our eigenvalue minimization allows us to incorporate any number of correction terms in an analytical form.

7.2 Setup

The four stripe patterns shown in Fig. 5 (above) were displayed on a large video screen (Fig. 5 below). We took images by placing the camera in various positions so that the stripe pattern appears in various parts of the view (recall that the view cannot be covered by a single planar pattern

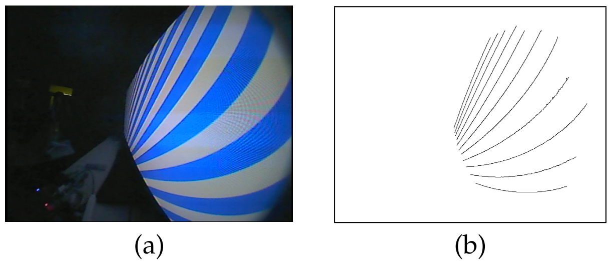


Fig. 6. (a) Fisheye lens image of a stripe pattern. (b) Detected edges.

image). The four patterns were cyclically displayed with blank frames in-between, and the camera is fixed in each position for at least one cycle to capture the four patterns; Fig. 6a shows one shot. The image size is 640×480 pixels. From each image, we detected edges; Fig. 6b shows the edges detected from the image in Fig. 6a. We manually removed those edges outside the display area. We also removed too small clusters of edge points. Then, we ran an edge segmentation algorithm to create connected edge segments. On each segment was imposed the collinearity constraint; on the segments in one frame were imposed the parallelism constraint; on the segments in one frame and the frame after the next with the same camera position were imposed the orthogonality constraint. In all, we obtained 220 segments, consisting of 20 groups of parallel segments and 10 orthogonal pairs, to which the LM procedure was applied.

The conventional approach using a reference grid board [2], [5], [7], [18], [19] would require precise localization of grid points in the image, which is a rather difficult task. Here, all we need to do is detect "continuous edges." We can use a video display instead of a specially designed grid board, *because our method is non-metric*: we need no metric information about the pattern or the camera positions. *We do not even know where each edge point corresponds to in the reference pattern.*

7.3 Results

Table 1 lists the calibration result up to the fifth correction degree. We set the frame center to be $(0,0)$ to specify the principal point (u_0, v_0). We stopped the LM iterations when the update increments in u_0, v_0, and f were less than ϵ_0 in magnitude and that in a_k less than ϵ_k in magnitude, where we let $\epsilon_0 = 10^{-3}$, $\epsilon_1 = 10^{-5}$, $\epsilon_2 = 10^{-6}$, $\epsilon_3 = 10^{-7}$, $\epsilon_4 = 10^{-8}$, and $\epsilon_5 = 10^{-9}$. Using various different initial values, we confirmed that the LM always converges to the same solution after around 10 to 20 iterations.

TABLE 1
Computed Parameters for Each Correction Degree

degree	0	1	2	3	4	5
u_0	−1.56744	−1.57819	−1.60647	−1.60103	−1.61170	−1.61145
v_0	0.529648	0.501021	0.427590	0.431905	0.432015	0.433677
f	146.648	149.567	148.110	146.724	146.793	146.499
$a_1/10^{-2}$	—	0.645886	−0.30601	−1.41625	−1.68918	−1.80625
$a_2/10^{-3}$	—	—	2.38948	7.57041	11.1085	9.34660
$a_3/10^{-4}$	—	—	—	−8.05083	−22.2515	−0.40049
$a_4/10^{-5}$	—	—	—	—	18.1185	−61.6432
$a_5/10^{-6}$	—	—	—	—	—	0.935930

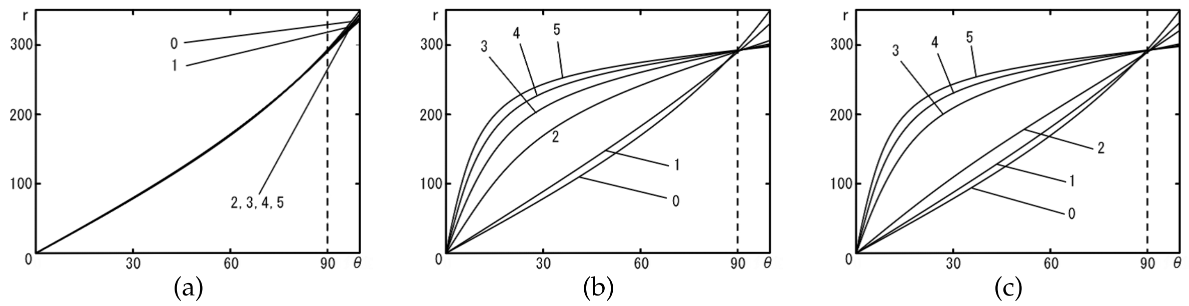


Fig. 7. The dependence of the distance r (pixels) from the focal point on the incidence angle θ (degrees) obtained by (a) using the collinearity, parallelism, and orthogonality constraints; (b) using only the collinearity constraints; (c) using the collinearity and parallelism constraints.

Fig. 7a plots the graph of (3) for different correction degrees. For the convenience of the subsequent applications, we numerically converted (3) to express the angle θ in terms of the distance r. As we see, the stereographic projection model in (1) holds fairly well even without any correction terms (degree 0). The result is almost unchanged for the degrees 3, 4, and 5, i.e., including powers up to r^7, r^9, and r^{11}. Thus, there is no need to increase the correction terms any further.

7.4 Spurious solutions

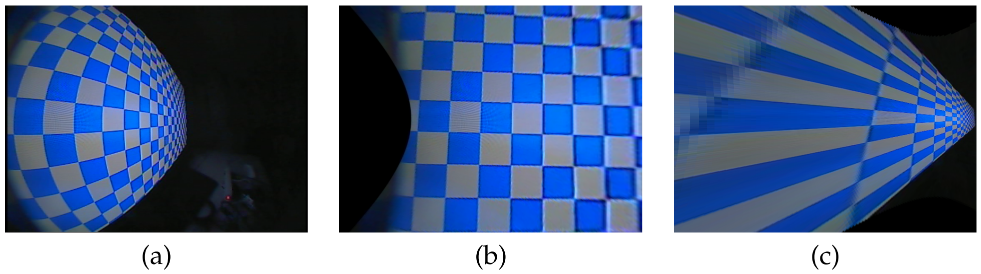
For comparison, Fig. 7b shows the same result using collinearity alone, i.e., $J = J_1/\gamma_1$ instead of (43); Fig. 7c shows the result using collinearity and parallelism, i.e., $J = J_1/\gamma_1 + J_2/\gamma_2$ instead of (43). In both cases, the graph approaches, as the degree increases, some r-θ relationship quite different from the stereographic projection. In order to see what this means, we did a rectification experiment. Fig. 8a shows a fisheye lens image viewing a square grid pattern in approximately 30 degree direction, and Fig. 8b is the rectified perspective image, using the parameters of correction degree 5 in Table 1. The image is converted to a view as if observed by rotating the camera by 60 degrees to face the pattern (see Section 7.6 for the procedure). The black area near the left boundary corresponds to 95 degrees or more from the optical axis. Thus, we can obtain a correct perspective image to the very boundary of the view.

Fig. 8c shows the result obtained by the same procedure using the spurious solution. We can see that collinear points are certainly rectified to be collinear and parallel lines to be (a skewed view of) parallel lines. We have confirmed that this spurious solution is *not* the result of a local minimum. Let us call the parameter values obtained by imposing collinearity, parallelism, and orthogonality the "correct solution." The cost functions $J = J_1/\gamma_1$ and $J = J_1/\gamma_1 + J_2/\gamma_2$ are certainly smaller at the spurious solution than at the correct solution, meaning that the spurious solution is not attributed to the minimization algorithm but is *inherent* in the formulation of the problem itself.

The fact that spurious solutions *should* exist is easily understandable if one considers perspective cameras. Suppose no image distortion exists. Then, if the assumed camera parameters, such as the focal length, the principal point, the aspect ratio, and the skew angle, are wrong, the resulting 3D interpretation is a projective transformation of the real scene, called *projective* reconstruction [3]. Projective transformations preserve collinearity but not parallelism or orthogonality. Imposing parallelism, we obtain *affine* reconstruction. Further imposing orthogonality, we obtain so called *Euclidean* reconstruction. Thus, orthogonality is essential for camera calibration; collinearity and parallelism alone are insufficient. This fact has been overlooked in the past collinearity-based work [1], [8], [12], [13], [15], [18], partly because spurious solutions can be prevented by using auxiliary information such as vanishing point estimation [2], [5], [11], [12] or antipodal point extraction [14], and partly because usually a small number, typically three, of collection terms are retained [1], [8], [12], [13], [15], [18], providing insufficient degrees of freedom to fall into a spurious solution.

7.5 Comparisons

Our fisheye lens camera was a specially designed one, and the camera manufacturer conducted calibration in a controlled environment by placing multiple grid patterns around the lens and manually detecting grid points in the image. We compared their $r(\theta)$ curve and ours shown in Fig. 7a with $K = 5$. Fig. 9 shows the difference of ours from theirs. We find that the difference in r is less than 0.5 pixels for θ between 0 and 75 degrees. However, the difference gradually



Fgi. 8. (a) Fisheye lens image viewing a square grid pattern in approximately 30 degree direction. (b) Rectified perspective image to be observed if the camera is rotated by 60 degrees to face the pattern. (c) Similarly rectified image using a spurious solution.

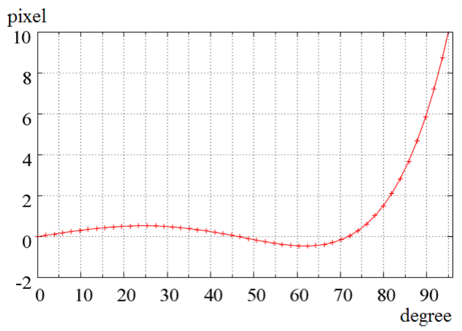


Fig. 9. The difference of our $r(\theta)$ curve from that of the camera manufacture.

increases beyond that, and 6 to 10 pixel differences arise beyond 90 degrees. Since the ground truth is unknown, we do not know which result is closer to the reality, but this comparison tells that precise calibration beyond the 90 degrees direction is rather difficult. According to the manufacture, this much difference is permissible for many practical applications in view of the saving of the cost and labor of controlled calibration.

7.6 Applications

Once the $r(\theta)$ relationship is obtained, we can not only perspectively rectify the fisheye lens image to a front view but also create a perspective view that would be observed if the camera were rotated by any given angle.

Define a world XYZ coordinate system with the origin O at the lens center and the Z-axis along the optical axis (Fig. 1). As far as the camera imaging is concerned, the outside scene can be regarded as if painted inside a sphere of a specified radius R surrounding the lens. The angle θ of the incident ray from point (X, Y, Z) on the sphere is

$$\theta = \tan^{-1}\frac{\sqrt{X^2+Y^2}}{Z} = \tan^{-1}\frac{\sqrt{R^2-Z^2}}{Z}. \tag{44}$$

The point (X, Y, Z) is projected to an image point (x, y) such that

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} u_0 \\ v_0 \end{pmatrix} + \frac{r(\theta)}{\sqrt{R^2-Z^2}}\begin{pmatrix} X \\ Y \end{pmatrix}. \tag{45}$$

Suppose we want to obtain a rectified perspective image with focal length $\bar{f}$. Then, the pixel $(\bar{x}, \bar{y})$ of the rectified image should be the projection of the 3D point (X, Y, Z) that satisfies

$$\bar{x} = \bar{f}\frac{X}{Z}, \qquad \bar{y} = \bar{f}\frac{Y}{Z}, \tag{46}$$

from which we obtain

$$Z = \frac{\bar{f}R}{\sqrt{\bar{x}^2+\bar{y}^2+\bar{f}^2}}. \tag{47}$$

Hence,

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \frac{R}{\sqrt{\bar{x}^2+\bar{y}^2+\bar{f}^2}}\begin{pmatrix} \bar{x} \\ \bar{y} \\ \bar{f} \end{pmatrix}. \tag{48}$$

Thus, the rectification to a front view can be done as follows (Fig. 10):

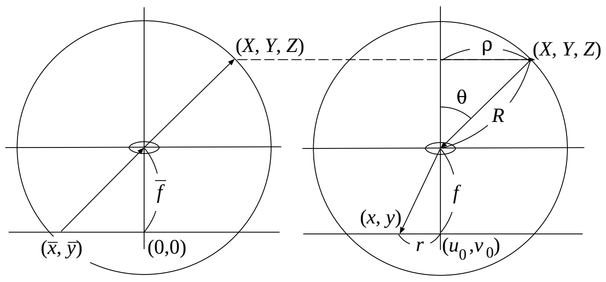


Fig. 10. The pixel $(\bar{x}, \bar{y})$ of the rectified perspective image of focal length $\bar{f}$ should be an image of the 3D point (X, Y, Z) that is actually projected to (x, y) on the fisheye lens image of focal length f.

1) For each pixel $(\bar{x}, \bar{y})$, compute the 3D coordinates (X, Y, Z) in (48), where the radius R is arbitrary (we may let $R = 1$).
2) Compute the corresponding pixel position (x, y) by (45) and copy its pixel value to $(\bar{x}, \bar{y})$. If (x, y) are not integers, interpolate its value from surrounding pixels.

Now, let us hypothetically rotate the camera by $\mathbf{R}$. Since rotation of the camera by $\mathbf{R}$ is equivalent to the rotation of the scene sphere by $\mathbf{R}^{-1}$, the 3D point (X, Y, Z) given by (48) on the rotated sphere corresponds to the 3D point

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \frac{R}{\sqrt{\bar{x}^2+\bar{y}^2+\bar{f}^2}}\mathbf{R}\begin{pmatrix} \bar{x} \\ \bar{y} \\ \bar{f} \end{pmatrix}. \tag{49}$$

on the original sphere. Its fisheye lens image should be at the pixel (x, y) given by (45). Thus, the mapping procedure goes as follows:

1) For each pixel $(\bar{x}, \bar{y})$, compute the 3D coordinates (X, Y, Z) in (49).
2) Compute the corresponding pixel position (x, y) by (45) and copy its pixel value to $(\bar{x}, \bar{y})$.

The top-left of Fig. 11 is an image of a street scene taken from a moving vehicle with a fisheye lens camera mounted below the bumper at the car front. The top-right of Fig. 11 is the rectified perspective image. The second and third rows show the rectified perspective images to be observed if the camera is rotated by 90 degrees left, right, up and down, confirming that we are really seeing more than 180 degree angles of view. Using a fisheye lens camera like this, we can detect vehicles approaching from left and/or right or create an image as if viewing the road from above the car.

8 Concluding Remarks

We have presented a new technique for calibrating ultra-wide fisheye lens cameras. Our method can be contrasted to the conventional approach of using a grid board as follows:

1) For ultra-wide fisheye lens cameras with more than 180 degrees of view, the image of any large (even infinite) plane cannot cover the image frame. Our method allows us take multiple partial images of the reference by freely moving the camera so that every

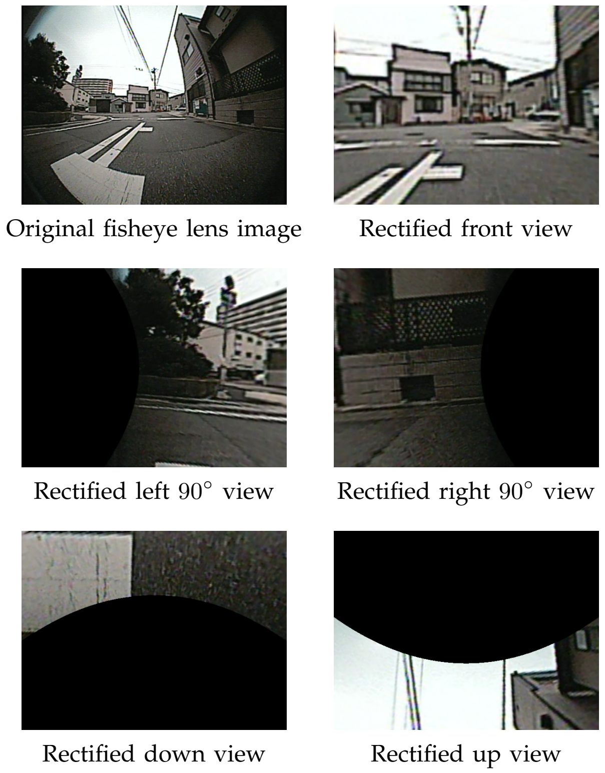


Fig. 11. Fisheye lens image of an outdoor scene taken from a moving vehicle, rectified front images, and rectified images after virtually rotating the camera to left, right, up, and down.

part of theframe is covered by some reference images.

2) The grid-based approach requires detection of “grid points” in the image, but accurate processing of a grid image is rather difficult. In our method, we only need to detect “continuous edges” of a stripe pattern.
3) The grid-based approach requires the correspondence of each detected grid point to the location on the reference. This is often difficult due to the periodicity of the grid. In our method, we need not know *where the detected edge points correspond to* in the reference.
4) In the grid-based approach, one needs to measure the camera position relative to the reference board by a mechanical means or by computing the homography from image-reference matching [2], [19]. Our method does not require any information about the camera position.
5) In the grid-based approach, one needs to create a reference pattern. This is not a trivial task. If a pattern is printed out on a sheet of paper and pasted to a planar surface, wrinkles and creases may arise and the glue may cause uneven deformations of the paper. In our method, *no metric information* is required about the pattern, so we can display it on any video screen.
6) In the grid-based approach, one can usually obtain only one image of the reference pattern from one camera position. In our method, we can fix the camera anywhere and freely change the reference pattern on the video screen.
7) The difficulty of processing a grid image with crossings and branches is circumvented by generating a “virtual grid” from separate stripe images of different orientations; each image has no crossings or branches.

The basic principle of our calibration is the imposition of the constraint that collinear points be rectified to collinear, parallel lines to parallel, and orthogonal lines to be orthogonal. Exploiting the fact that line fitting reduces to eigenvalue problems in 3D, we derived an optimization procedure in analytical terms, and did experiments, displaying a variable reference pattern on a video screen. Video screen patterns were also used by Komagata et al. [9] as a reference, but our eigenvalue minimization technique can fully exploit this scheme. We found that a *spurious solution* exists if the collinearity constraint alone is used or even combined with the parallelism constraint. However, we have shown that incorporating the orthogonality constraint allows an accurate calibration without using any auxiliary information such as vanishing point estimation [2], [5], [11], [12]. We have also shown a real image example using a vehicle-mounted fisheye lens camera. It is expected that our procedure is going to be a standard tool for fisheye lens camera calibration.

Acknowledgments

The author thank Ryota Moriyasu of ERD Corporation and Ryohei Nakamura of Kyocera Corporation for helping the experiments. This work was supported in part by JSPS Grant-in-Aid for Challenging Exploratory Research (24650086).

References

[1] F. Devernay and O. Faugeras, “Straight Lines Have to Be Straight: Automatic Calibration and Removal of Distortion from Scenes of Structured Environments,” *Machine Vision Applications*, vol. 13, no. 1, pp. 14–24, Aug. 2001.
[2] R. Hartley and S.B. Kang, “Parameter-Free Radial Distortion Correction with Center of Distortion Estimation,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 28, no. 8, pp. 1309–1321, Aug. 2007.
[3] R. Hartley and O. Zisserman, *Multiple View Geometry in Computer Vision*, second ed. Cambridge Univ. Press, 2004.
[4] J. Heikkilä, “Geometric Camera Calibration Using Circular Control Points,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1066–1077, Oct. 2000.
[5] C. Hughes, P. Denny, M. Glavin and E. Jones, “Equidistant Fish-Eye Calibration and Rectification by Vanishing Point Extraction,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2289–2296, Dec. 2010.
[6] K. Kanatani, *Statistical Optimization for Geometric Computation: Theory and Practice*. Elsevier, 1996.
[7] J. Kannala and S.S. Brandt, “A General Camera Model and Calibration Method for Conventional, Wide Angle, and Fisheye-Lenses,” *IEEE Trans. Pattern Analysis and Machine Intelligence.*, vol. 28, no. 8, pp. 1335–1340, Aug. 2006.
[8] S. Kase, H. Mitsumoto, Y. Aragaki, N. Shimomura and K. Umeda, “A Method to Construct Overhead View Images Using Multiple Fish-Eye Cameras,” *J. Japan Soc. for Precision Eng.*, vol. 75, no. 2, pp. 251–255, Feb. 2009.
[9] H. Komagata, I. Ishii, A. Takahashi, D. Wakabayashi and H. Imai, “A Geometric Calibration Method of Internal Camera Parameters for Fish-Eye Lenses,” *IEICE Trans. Information and Systems*, vol. J89-D, no. 1, pp. 64–73, Jan. 2006.
[10] Y.-C. Liu, K.-Y. Lin, and Y.-S. Chen, “Bird’s Eye View Vision System for Vehicle Surrounding Monitoring,” *Proc. Second Int’l Workshop Robt Vision*, pp. 207–218, Feb. 2008.
[11] M. Nakano, S. Li and N. Chiba, “Calibration of Fish-Eye Camera for Acquisition of Spherical Image,” *IEICE Trans. Information and Systems*, vol. J89-D-II, no. 9, pp. 1847–1856, Sept. 2005.

[12] M. Nakano, S. Li and N. Chiba, "Calibrating Fisheye Camera by Stripe Pattern Based upon Spherical Model," *IEICE Trans. Information and Systems*, vol. J89-D, no. 1, pp. 73–82, Jan. 2007.

[13] R. Okutsu, K. Terabayashi, Y. Aragaki, N. Shimomura, and K. Umeda, "Generation of Overhead View Images by Estimating Intrinsic and Extrinsic Camera Parameters of Multiple Fish-Eye Cameras," *Proc. IAPR Conf. Machine Vision Applications*, pp. 447–450, May 2009.

[14] R. Okutsu, K. Terabayashi and K. Umeda, "Calibration of Intrinsic Parameters of a Fish-Eye Camera Using a Sphere," *IEICE Trans. Information and Systems*, vol. J89-D, no. 12, pp. 2645–2653, Dec. 2010

[15] Y. Onodera and K. Kanatani, "Geometric Correction of Images without Camera Registration," *IEICE Trans. Information and Systems*, vol. J75-D-II, no. 5, pp. 1009–1013, May 1992.

[16] T. Papadopoulo and M. I. A. Lourakis, "Estimating the Jacobian of the Singular Value Decomposition: Theory and Applications," *Proc. Sixth European Conf. Computer Vision*, vol. 1, pp. 554–570, June/July 2000.

[17] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, second ed. Cambridge Univ. Press, 1992.

[18] F. Swaminathan and S. K. Nayar, "Nonmetric Calibration of Wide-Angle Lenses and Polycameras," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 10, pp. 1172–1178, Oct. 2000.

[19] Z. Zhang, "Flexible New Technique for Camera Calibration," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000.

Kenichi Kanatani received his B.E., M.S., and Ph.D. in applied mathematics from the University of Tokyo in 1972, 1974 and 1979, respectively. After serving as a orofessor of computer science at Gunma University, Gunma, Japan, he is currently a professor of computer science at Okayama University, Okayama, Japan. He is the author of many books on computer vision, including *Group-Theoretical Methods in Image Understanding* (Springer, 1990), *Geometric Computation for Machine Vision* (Oxford University Press, 1993), and *Statistical Optimization for Geometric Computation: Theory and Practice* (Elsevier 1996). He is a fellow of the IEEE.