Auto-calibration — Kruppa's equations and the intrinsic parameters of a camera

S.D. Hippisley-Cox & J. Porrill
AI Vision Research Unit
University of Sheffield
e-mail: [S.D.Hippisley-Cox,J.Porrill]@aivru.sheffield.ac.uk

Abstract

Auto-calibration may be defined as the process of finding the intrinsic parameters of a camera from real image data. Recent techniques for finding these parameters rely upon solving equations which relate the epipolar geometry of two camera positions with the intrinsic parameters, equations known as Kruppa's equations[4, 2].

These techniques involve a very time consuming numerical process, and yet only produce two of the intrinsic parameters, the focal length and the aspect ratio, to an acceptable degree of accuracy. Further processes which, for example, compute the camera's movement need to assume standard values for the other parameters[4].

In this paper, we present a method of solving Kruppa's equations for the focal length and the aspect ratio which is suitable for a real-time system, together with details of experiments using simulated noisy data which show that its accuracy is comparable with the previous method.

1 Introduction

The process of camera calibration may be described as the process whereby we obtain data about our camera setup to enable us to express it in a standard form. For example, for a static pair of cameras, this may involve computing internal camera parameters which relate physical image coordinates to world-based units in an ideal camera, together with the two camera positions. These parameters, referred to as intrinsic and extrinsic respectively, could then be used to rectify the images into parallel camera geometry for subsequent processing by a vision system. For a mobile camera, calibration may involve computing only the intrinsic parameters.

Traditional calibration systems (see e.g. [8]) involve the use of a special calibration object to facilitate this process. Auto-calibration may be defined as the process of finding the intrinsic parameters of a camera from general image data. For a linear projective pinhole camera, there are at most five intrinsic parameters:— the magnifications on the x- and y-axes (or, equivalently, the focal length in pixels and the aspect ratio), the location of the principal point, and the angle between the x- and y-axes. Recently, Hartley[3] and Faugeras, Luong & Maybank[2] presented

BMVC 1994 doi:10.5244/C.8.76

methods of finding intrinsic parameters from image data. Hartley used a one parameter camera model, and showed that, for an image pair taken with two suitable cameras, by factoring a particular matrix, one can recover the focal length, in pixels, of both cameras. Faugeras, Luong & Maybank used a four or five parameter camera model, and used three or more image pairs taken with the same camera to obtain its intrinsics. Their methods used the absolute conic, obtaining and solving Kruppa's equations. Of the values produced, only two, the x- and y-scale, were recovered with enough accuracy to pass on to further processing[4]. In addition, their method for solving Kruppa's equations was very processor intensive, requiring one minute on a network of seven Sun-4s[2].

In this paper, we present a method of solving Kruppa's equations for the x- and y-scale which is suitable for a real time system. We work with a two parameter camera model, assuming that the principal point is known and that the angle between the axes is $90°$. The method relies upon the observation that, for this camera, solving Kruppa's equations reduces to solving a quartic equation in one variable. This is, of course, quite fast. Also, each set of point matches (and hence estimate of the epipolar geometry) gives us an estimate of the intrinsics. These values can be collated or fed into a Kalman filter to produce an adaptive estimate. Further, a corollary is that we can use unusual scale values to detect "bad" epipolar geometries.

This method has been implemented in ANIT, AIVRU's parallel vision development system[1]. Preliminary experiments based on simulated data, detailed below, indicate that this method produces calibrations of the same order of accuracy as those of Faugeras, Luong & Maybank[2]. One would expect this, as they solve the same equations under the same conditions.

Advantages of this approach are:—

- it is fast;
- it is reasonably accurate;
- there an estimate of the intrinsics for every image pair processed;
- it is good for a parallel environment; and
- it produces a good starting point for systems which further refine the intrinsics.

2 Kruppa's equations

In this section, both to establish notation and for completeness, we derive the relationship between the epipolar geometry of a pair of camera positions, (represented by the coordinates of the epipoles in the images and the epipolar transformation), and the intrinsic parameters; that is, we derive Kruppa's equations.

For an alternative presentation, see [2, 4, 5].

2.1 The absolute conic and the intrinsic parameters

The *absolute conic* is the locus of points of P^3, $\mathbf{x} = (x_1, x_2, x_3, 0)^T$, which satisfy

$$\mathbf{x}^T\mathbf{x} = 0.$$

Any point satisfying this constraint will have at least one of x_1, x_2 or x_3 complex. The key thing for us here is that the absolute conic is invariant to rigid body motion. To see this, let

$$M = \begin{pmatrix} R & \mathbf{t} \\ 0_3^T & 1 \end{pmatrix}$$

be a rigid body motion, let $\xi = (\xi_1, \xi_2, \xi_3)$, and let

$$\mathbf{x} = \begin{pmatrix} \xi \\ 0 \end{pmatrix}$$

be a point on the absolute conic. Then

$$\begin{aligned} (M\mathbf{x})^T M\mathbf{x} &= (R\xi)^T R\xi \\ &= \xi^T R^T R\xi \\ &= \xi^T \xi \\ &= 0, \end{aligned}$$

showing that $M\mathbf{x}$ satisfies the equation of the absolute conic.

Now, as the absolute conic is invariant to rigid body motion, its image in a camera will be the same whatever the position of the camera. Now the projective matrix P of a camera can be decomposed into a rigid body movement M, projection, and intrinsic parameter matrix A, i.e.

$$P = ADM,$$

where

$$D = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Let $\mathbf{y}$ be a point on the image of the absolute conic. Then

$$\mathbf{y} = P\mathbf{x} = AD\mathbf{x}',$$

where $\mathbf{x}'$ is a(n other) point on the absolute conic. Writing $D\mathbf{x}'$ as ξ, we have

$$0 = \xi^T \xi = (A^{-1}\mathbf{y})^T (A^{-1}\mathbf{y}) = \mathbf{y}^T (A^{-T} A^{-1})\mathbf{y},$$

showing that the image of the absolute conic is

$$(AA^T)^{-1}.$$

The corresponding line conic, K, is the inverse of its transpose, giving

$$K = AA^T,$$

highlighting the close relationship between the image of the absolute conic and the intrinsic parameters of the camera.

With the four parameter version of the intrinsic parameters

$$A = \begin{pmatrix} a_x & 0 & c_x \\ 0 & a_y & c_y \\ 0 & 0 & 1 \end{pmatrix},$$

we have

$$K = \begin{pmatrix} {a_x}^2 + {c_x}^2 & c_x\, c_y & c_x \\ c_x\, c_y & {a_y}^2 + {c_y}^2 & c_y \\ c_x & c_y & 1 \end{pmatrix},$$

where a_x, a_y, c_x, and c_y are the x-scale, y-scale, and x and y coordinates of the principal point respectively.

2.2 The epipolar geometry and the absolute conic

Let us suppose that we have moved our camera, and, following Faugeras, Luong & Maybank[2] have calculated the F-matrix representing the epipolar geometry from point matches, and determined the epipoles $\mathbf{e}_1$, $\mathbf{e}_2$ and the epipolar transformation

$$\tau' = \frac{a\tau + b}{c\tau + d}$$

from F.

If we now restrict our attention to epipolar lines which are tangent to the image of the absolute conic, the epipolar transformation gives us a constraint on K which we can exploit to give us three equations in the five unknowns of K. These equations, (only two of which are linearly independent), are known as Kruppa's equations.

In detail:-

- First we parameterise the lines through the epipole $\mathbf{e}_1$ by their intersection $(1, \tau, 0)$ with the line at infinity. The line is therefore

$$\mathbf{y}_1 = \mathbf{e}_1 \times (1, \tau, 0).$$

- For one of these lines to be a tangent to the image of the absolute conic, we have

$$\mathbf{y}_1^T K \mathbf{y}_1 = 0.$$

  This is a quadratic in τ, and can be written as

$$\alpha_1 \tau^2 + \beta_1 \tau + \gamma_1 = 0,$$

  the roots of which give the two tangents.

- Under the epipolar transformation, each line $\mathbf{y}_1 = \mathbf{e}_1 \times (1, \tau, 0)$, is mapped to $\mathbf{y}_2 = \mathbf{e}_2 \times (1, \tau', 0)$. Now, since tangents to the image of the absolute conic remain so under the epipolar transformation, we have

$$\mathbf{y}_2^T K \mathbf{y}_2 = 0.$$

  Writing $\mathbf{y}_2$ in terms of $\mathbf{e}_2$ and τ, we have another quadratic in τ, e.g.

$$\alpha_2 \tau^2 + \beta_2 \tau + \gamma_2 = 0.$$

- As both quadratics have the same roots, we have

$$\frac{\alpha_1}{\alpha_2} = \frac{\beta_1}{\beta_2} = \frac{\gamma_1}{\gamma_2},$$

or

$$\left.\begin{array}{rcl} \alpha_1\beta_2 & = & \alpha_2\beta_1 \\ \beta_1\gamma_2 & = & \gamma_1\beta_2 \\ \alpha_1\gamma_2 & = & \alpha_2\gamma_1 \end{array}\right\} (*)$$

These are Kruppa's equations. Maple code to generate them is available from the authors. It is easy to see that $(*)$ are quadratic forms in the unknowns of K.

3 Solving Kruppa's equations

There are five parameters to find — the x-scale, the y-scale, the principal point and the angle between the axes. We assume that the angle between the axes is 90^o, and that the principal point is known.

After some algebra, two of Kruppa's equations can then be expressed in the form

$$\begin{aligned} a_1\, a_x^4 + c_1\, a_x^2 a_y^2 + d_1\, a_x^2 + e_1\, a_y^2 + f_1 &= 0 \\ b_2\, a_y^4 + c_2\, a_x^2 a_y^2 + d_2\, a_x^2 + e_2\, a_y^2 + f_2 &= 0 \end{aligned}$$

where a_i, b_i, ..., f_i, $i \in \{1, 2\}$ are functions of the epipolar geometry and principal point, constant for any particular image pair. Making the substitutions $x = a_x^2$, $y = a_y^2$, they become

$$a_1\, x^2 + c_1\, xy + d_1\, x + e_1\, y + f_1 = 0 \quad (1)$$

$$b_2\, y^2 + c_2\, xy + d_2\, x + e_2\, y + f_2 = 0 \quad (2)$$

Our problem is reduced to finding the 4 intersection points of two particular conics. Solving for y in the first equation, we obtain

$$y = -\frac{a_1\, x^2 + d_1\, x + f_1}{c_1\, x + e_1} \quad (3)$$

and substituting this into the second gives us the following quartic:—

$$\begin{aligned} &(b_2\, a_1^2 - c_2\, a_1\, c_1)\ x^4 + \\ &(d_2\, c_1^2 - c_2\, d_1\, c_1 - e_2\, a_1\, c_1 - c_2\, a_1\, e_1 + 2\, b_2\, a_1\, d_1)\ x^3 + \\ &(2\, d_2\, c_1\, e_1 + f_2\, c_1^2 + b_2\, d_1^2 - e_2\, a_1\, e_1 - e_2\, d_1\, c_1 + 2\, b_2\, a_1\, f_1 - \\ &\qquad c_2\, f_1\, c_1 - c_2\, d_1\, e_1)\ x^2 + \\ &(2\, b_2\, d_1\, f_1 + d_2\, e_1^2 - e_2\, f_1\, c_1 - c_2\, f_1\, e_1 + 2\, f_2\, c_1\, e_1 - e_2\, d_1\, e_1)\ x + \\ &(b_2\, f_1^2 + f_2\, e_1^2 - e_2\, f_1\, e_1)\ 1 = 0. \end{aligned} \quad (4)$$

This is easy to solve using a standard numerical method (see e.g. [7]). Using each of the four roots, we can find, using Equation (3), our value for y. As each (x, y)

pair clearly satifies the original conic equations, we have found the points of their intersection.

We then eliminate x-y pairs to leave those for which both x and y are real and positive. Taking the pointwise square root of those pairs which remain, we have potential candidates for our intrinsic parameters. Theoretically, there is only one set of intrinsics compatible with each epipolar geometry, and so one may expect there generally to be only one possibility offered. However, this was not borne out by our simulation, which typically offered between zero and three sets of intrinsics. When intrinsics were offered, only set was a realistic choice.

Thus, for each F-matrix, we have found an estimate of the two intrinsic parameters, a_x and a_y.

4 Experiments

4.1 A true simulation

The following simulation was built to test the theory:—

1. Given a cloud of data points and two camera positions, calculate the positions of these points in the images.
2. Peturb these to simulate image noise, (assumed to be gaussian).
3. Pass the noisy matches to a routine which calculates the F-matrix, and this, in turn, to a routine which calculates the coordinates of the epipoles and the epipolar transformation.
4. Use this epipolar geometry to calculate and subsequently solve the quartic of Section 3 for the intrinsic parameters a_x and a_y.

With a_x and a_y set to 1000 and 800 respectively, data clouds of between 25 and 45 points were passed through this system 10 times and the intrinsics offered were collated. We give the results for three pairs of camera positions with four levels of pixel noise α in Figure 1.

A typical run of 10, for the first camera position with $\alpha = 0.2$ is given in Figure 5.

4.2 Sensitivity to location of principal point

The algorithm requires that we provide it with an estimation of the location of the principal point. To test its sensitivity to this assumption, we passed the same data through the above simulation, (with image noise 0.2 pixel, principal point at $(256, 256)$), varying the its estimate over a square grid bounded by $(230, 230)$ and $(270, 270)$. The intrinsic parameters obtained are displayed as a cross-fusion stereo pair in Figure 3.

As can be seen, the error surfaces are approximately planar, with an $n\%$ error in the principal point leading to less than a $1.5 \times n\%$ error in the intrinsics.

4.3 Comparison with Luong

To compare our system with the results given by Luong in his thesis[4], a system was built to do the following:—

1. Given two camera positions, it calculates the epipoles and the epipolar transformation.
2. It then peturbs these to simulate image noise. (For a particular noise level α, the value of each of the five components ζ_i, $i = 1, \ldots, 5$, is taken from $N(\zeta_i, \alpha||\zeta_i||)$.)
3. This noisy epipolar geometry is then used to calculate and subsequently solve the quartic of Section 3.

With a_x and a_y set to 1000 and 800 respectively, as before, camera positions were passed through this system 10 times, and the intrinsics offered were collated. We give the results for three pairs of camera positions with three levels of noise in Figure 4.

For Luong[4], the noise on the epipolar geometry was typically in the range $[0.05, 0.10]$.

A typical run of 10, for the first camera position with $\alpha = 0.05$ is given in Figure 5.

Work is currently in progress on a system which will test the algorithm on real images.

References

[1] Cornell, S.M., Mayhew, J.E.W. and Harrison, S., *ANIT, a system for perceptual subsumption and intelligent vision systems*, AIVRU internal memo.

[2] Faugeras, O.D., Luong, Q.-T. and Maybank, S.J., Camera self-calibration: theory and experiments, *Proc. 2nd ECCV*, 321-334, Springer-Verlag, 1992.

[3] Hartley, R.I., Estimation of relative camera positions for uncalibrated cameras, *Proc. 2nd ECCV*, 579-587, Springer-Verlag, 1992.

[4] Luong, Q.-T., *Matrice fondamentale et calibration visuelle sur l'environnement — vers une plus grande autonomie des systemes robotiques*, PhD. thesis, Université de Paris-sud, 1992.

[5] Kruppa, E., *Zur Ermittlung eines Objectes aus zwei Perspektiven mit innerer Orientierung*, Sitz.-Ber. Akad. Wiss., Wien. Math. Naturw. Kl. (Abt. IIa 122): 1939-1948.

[6] Maybank, S.J. and Faugeras, O.D., A theory of self-calibration of a moving camera, *International Journal of Computer Vision*, 8:2, 123-151, 1992.

[7] Press, W.H., Flannery B.P., Teukolsky, S.A. and Vetterling, W.T., *Numerical recipes in C*, C.U.P., 1988.

[8] Tsai, R., An efficient and accurate camera calibration technique for 3D machine vision, *Proc. CVPR '86*, 364-374, IEEE, 1986.

	Noise level	a_x		a_y	
		μ	σ	μ	σ
Camera position 1	0.1	998	3	797	6
	0.2	996	8	795	14
	0.5	999	33	797	47
	1.0	994	37	794	60
Camera position 2	0.1	998	18	798	21
	0.2	1008	30	806	38
	0.5	991	37	788	64
	1.0	932	107	734	138
Camera position 3	0.1	1000	3	800	4
	0.2	999	4	801	6
	0.5	1005	8	804	10
	1.0	994	19	799	26

Figure 1: Simulation — performance of the algorithm.

a_x	1003	1009	933	991	993	1001	988	986	989	1007
a_y	807	815	787	791	792	796	776	777	793	817

Figure 2: Simulation — a typical run.

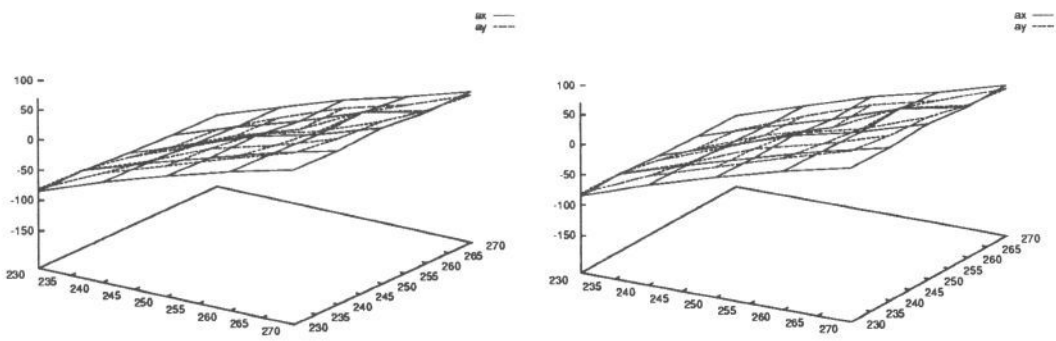

Figure 3: Sensitivity to principal point — stereo pair.

	Noise level	a_x		a_y	
		μ	σ	μ	σ
Camera position 1	0.01	1018	40	795	4
	0.05	1005	191	807	58
	0.10	1054	262	741	80
Camera position 2	0.01	1057	110	797	9
	0.05	964	457	789	63
	0.10	1000	947	816	120
Camera position 3	0.01	1012	57	796	14
	0.05	1082	126	783	51
	0.10	1008	288	748	99

Figure 4: Comparison with Luong — performance of the algorithm.

a_x	1055	688	1078	1032	1003	984	739	1273	1272	922
a_y	841	833	726	848	830	835	884	722	728	820

Figure 5: Comparison with Luong — a typical run.