Compact Algorithm for Strictly ML Ellipse Fitting

Kenichi Kanatani
Department of Computer Science, Okayama University, Okayama 700-8530 Japan
kanatani@suri.cs.okayama-u.ac.jp

Yasuyuki Sugaya
Department of Information and Computer Sciences,
Toyohashi University of Technology, Toyohashi, Aichi 441-8580 Japan
sugaya@iim.ics.tut.ac.jp

Abstract

A very compact algorithm is presented for fitting an ellipse to points in images by maximum likelihood (ML) in the strict sense. Although our algorithm produces the same solution as existing ML-based methods, it is probably the simplest and the smallest of all. By numerical experiments, we show that the strict ML solution practically coincides with the Sampson solution.

1. Introduction

Circular objects in the scene are projected onto ellipses on the image plane, and their 3-D shapes and positions can be computed from their images [11]. For this reason, fitting an ellipse to a point sequence is one of the first steps of various vision applications. Although how to select pixels that form an elliptic arc is an important issue in practice (see, e.g., [13]), we focus here on fitting an ellipse equation to pixels on an elliptic arc, assuming that such pixels are already selected.

Most of the methods presented in early days were based on heuristics combining voting and least squares in many different forms [1, 3, 16, 19, 21]. Later, the problem was formulated as statistical estimation in the presence of noise [4, 5, 10, 12, 22]. If the noise is independent isotropic Gaussian, *maximum likelihood* (*ML*) is to minimize the sum of squares of the orthogonal distances of points to the fitted ellipse, called the *reprojection error* [10].

If the noise is small, the problem reduces to minimization of an expression called the *Sampson error* [10], for which many numerical schemes exist including *FNS* [6], *HEIV* [17, 18], and the *projective Gauss-Newton iterations* [14]. There are also attempts to minimize the reprojection error directly for obtaining the strict ML solution [2, 8, 9, 20], searching a high-dimensional parameter space or solving a high-degree polynomial. However, not much is known as to how the strict ML solution differs from the Sampson solution.

In this paper, we present a new algorithm for computing the strict ML solution. Although there is no accuracy gain, since all ML-based methods minimize the same function, our algorithm is far more compact than any of existing methods. The reason why the existing strictly ML algorithms [2, 8, 9, 20] are not widely used is probably for fear of coding a complicated program and uneasiness at relying on "download". Our algorithm is simple enough to code oneself[1].

We describe our algorithm in Sec. 2 and give a derivation in Sec. 3. In Sec. 4, we compare the strict ML solution with the Sampson solution. We conclude in Sec. 5.

2. Strictly ML Ellipse Fitting

We want to fit an ellipse to N points $\{(x_\alpha, y_\alpha)\}_{\alpha=1}^N$. An ellipse is represented by

$$Ax^2 + 2Bxy + Cy^2 + 2f_0(Dx + Ey) + Ff_0^2 = 0, \quad (1)$$

where f_0 is an arbitrary scaling constant (we set f_0 = 600 pixels in our experiments). If we define the 3-D vector $\boldsymbol{x}$ and the 3×3 symmetric matrix

$$\boldsymbol{x} = \begin{pmatrix} x/f_0 \\ y/f_0 \\ 1 \end{pmatrix}, \qquad \boldsymbol{Q} = \begin{pmatrix} A & B & D \\ B & C & E \\ D & E & F \end{pmatrix}, \quad (2)$$

the ellipse equation (1) is written as $(\boldsymbol{x}, \boldsymbol{Q}\boldsymbol{x}) = 0$; throughout this paper, we denote the inner product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ by $(\boldsymbol{a}, \boldsymbol{b})$. Hereafter, the point represented by vector $\boldsymbol{x}$ is referred to simply as "point $\boldsymbol{x}$".

Let $\{\boldsymbol{x}_\alpha\}_{\alpha=1}^N$ be points supposedly on an elliptic arc. If they are assumed to be displaced from their true positions $\{\bar{\boldsymbol{x}}_\alpha\}_{\alpha=1}^N$ by independent and identical Gaussian noise in their x- and y-coordinates, maximum likelihood (ML) is equivalent to minimizing the *reprojection error*

[1]But one can try ours if one wishes:
http://www.iim.ics.tut.ac.jp/~sugaya/public-e.html

978-1-4244-2175-6/08/$25.00 ©2008 IEEE

$$E = \sum_{\alpha=1}^{N} \|\boldsymbol{x}_\alpha - \bar{\boldsymbol{x}}_\alpha\|^2, \tag{3}$$

with respect to $\bar{\boldsymbol{x}}_\alpha$ and $\boldsymbol{Q}$ subject to

$$(\bar{\boldsymbol{x}}_\alpha, \boldsymbol{Q}\bar{\boldsymbol{x}}_\alpha) = 0, \qquad \alpha = 1, ..., N. \tag{4}$$

Geometrically, we are minimizing the sum of the orthogonal distances of the points to the fitted ellipse.

Equation (1) does not necessarily describe an ellipse. Even if the points $\{\boldsymbol{x}_\alpha\}_{\alpha=1}^N$ are sampled from an ellipse, the fitted equation may define a hyperbola in the presence of large noise, and a technique for preventing this has also been proposed [7]. Here, we do not impose any constraints, regarding a fitted curve as a solution.

We now present a dramatically compact fitting algorithm. Write the matrix $\boldsymbol{Q}$ in (2) as the 6-D vector

$$\boldsymbol{u} = \begin{pmatrix} A & B & C & D & E & F \end{pmatrix}^\top. \tag{5}$$

Since the absolute scale of $\boldsymbol{Q}$ is indeterminate, we normalize $\boldsymbol{u}$ to $\|\boldsymbol{u}\| = 1$. We use the notation $\mathcal{N}[\,\cdot\,]$ to denotes normalization to unit norm.

In order to emphasize the compactness of our algorithm, we state it first and then give its derivation. The main routine of our algorithm goes as follows:

main

1. Let $\boldsymbol{u}_0 = \boldsymbol{0}$, and initialize $\boldsymbol{u}$.
2. Let $\hat{x}_\alpha = x_\alpha$, $\hat{y}_\alpha = y_\alpha$ and $\tilde{x}_\alpha = \tilde{y}_\alpha = 0$.
3. Compute the following 6-D vectors $\boldsymbol{\xi}_\alpha$ and the 6×6 matrices $V_0[\boldsymbol{\xi}_\alpha]$:

$$\boldsymbol{\xi}_\alpha = \begin{pmatrix} \hat{x}_\alpha^2 + 2\hat{x}_\alpha\tilde{x}_\alpha \\ 2(\hat{x}_\alpha\hat{y}_\alpha + \hat{y}_\alpha\tilde{x}_\alpha + \hat{x}_\alpha\tilde{y}_\alpha) \\ \hat{y}_\alpha^2 + 2\hat{y}_\alpha\tilde{y}_\alpha \\ 2f_0(\hat{x}_\alpha + \tilde{x}_\alpha) \\ 2f_0(\hat{y}_\alpha + \tilde{y}_\alpha) \\ f_0^2 \end{pmatrix}, \tag{6}$$

$$V_0[\boldsymbol{\xi}_\alpha] = \begin{pmatrix} \hat{x}_\alpha^2 & \hat{x}_\alpha\hat{y}_\alpha & 0 & f_0\hat{x}_\alpha & 0 & 0 \\ \hat{x}_\alpha\hat{y}_\alpha & \hat{x}_\alpha^2 + \hat{y}_\alpha^2 & \hat{x}_\alpha\hat{y}_\alpha & f_0\hat{y}_\alpha & f_0\hat{x}_\alpha & 0 \\ 0 & \hat{x}_\alpha\hat{y}_\alpha & \hat{y}_\alpha^2 & 0 & f_0\hat{y}_\alpha & 0 \\ f_0\hat{x}_\alpha & f_0\hat{y}_\alpha & 0 & f_0^2 & 0 & 0 \\ 0 & f_0\hat{x}_\alpha & f_0\hat{y}_\alpha & 0 & f_0^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{7}$$

4. Call *FNS* to update $\boldsymbol{u}$.
5. If $\boldsymbol{u} \approx \boldsymbol{u}_0$ up to sign, return $\boldsymbol{u}$ and stop. Else, update $\tilde{x}_\alpha$ and $\tilde{y}_\alpha$ by

$$\begin{pmatrix} \tilde{x}_\alpha \\ \tilde{y}_\alpha \end{pmatrix} \leftarrow \frac{(\boldsymbol{u}, \boldsymbol{\xi}_\alpha)}{2(\boldsymbol{u}, V_0[\boldsymbol{\xi}_\alpha]\boldsymbol{u})} \begin{pmatrix} u_1 & u_2 & u_3 \\ u_4 & u_5 & u_6 \end{pmatrix} \begin{pmatrix} \hat{x}_\alpha \\ \hat{y}_\alpha \\ f_0 \end{pmatrix}. \tag{8}$$

6. Go back to Step 3 after the following update:

$$\boldsymbol{u}_0 \leftarrow \boldsymbol{u}, \;\; \hat{x}_\alpha \leftarrow x_\alpha - \tilde{x}_\alpha, \;\; \hat{y}_\alpha \leftarrow y_\alpha - \tilde{y}_\alpha. \tag{9}$$

The initialization in Step 1 can be done by least squares or the Taubin method [14]. The FNS routine in Step 4 goes as follows:

FNS

1. Compute the following 6×6 matrix $\boldsymbol{X}$:

$$\boldsymbol{X} = \sum_{\alpha=1}^{N} \frac{\boldsymbol{\xi}_\alpha\boldsymbol{\xi}_\alpha^\top}{(\boldsymbol{u}, V_0[\boldsymbol{\xi}_\alpha]\boldsymbol{u})} - \sum_{\alpha=1}^{N} \frac{(\boldsymbol{u}, \boldsymbol{\xi}_\alpha)^2 V_0[\boldsymbol{\xi}_\alpha]}{(\boldsymbol{u}, V_0[\boldsymbol{\xi}_\alpha]\boldsymbol{u})^2}. \tag{10}$$

2. Compute the unit eigenvectors $\boldsymbol{u}'$ and of $\boldsymbol{X}$ for the smallest eigenvalues.
3. If $\boldsymbol{u}' \approx \boldsymbol{u}$ up to sign, return $\boldsymbol{u}'$ and stop. Else, let $\boldsymbol{u} \leftarrow \mathcal{N}[\boldsymbol{u} + \boldsymbol{u}']$ and go back to Step 1.

---

3. Derivation

We want to compute $\bar{\boldsymbol{x}}_\alpha$ that minimizes (3) subject to (4). Suppose we have obtained an approximate solution $\hat{\boldsymbol{x}}_\alpha \approx \bar{\boldsymbol{x}}_\alpha$ (initially, we let $\hat{\boldsymbol{x}}_\alpha = \boldsymbol{x}_\alpha$). We write

$$\bar{\boldsymbol{x}}_\alpha = \hat{\boldsymbol{x}}_\alpha - \Delta\hat{\boldsymbol{x}}_\alpha, \tag{11}$$

and compute the correction term $\Delta\hat{\boldsymbol{x}}_\alpha$ instead. Substitution of (11) into (3) yields

$$E = \sum_{\alpha=1}^{N} \|\tilde{\boldsymbol{x}}_\alpha + \Delta\hat{\boldsymbol{x}}_\alpha\|^2, \tag{12}$$

where we define

$$\tilde{\boldsymbol{x}}_\alpha = \boldsymbol{x}_\alpha - \hat{\boldsymbol{x}}_\alpha. \tag{13}$$

The ellipse equation (4) now becomes

$$(\hat{\boldsymbol{x}}_\alpha - \Delta\hat{\boldsymbol{x}}_\alpha, \boldsymbol{Q}(\hat{\boldsymbol{x}}_\alpha - \Delta\hat{\boldsymbol{x}}_\alpha)) = 0. \tag{14}$$

If $\hat{\boldsymbol{x}}_\alpha$ is a good approximation to $\bar{\boldsymbol{x}}_\alpha$, the correction term $\Delta\hat{\boldsymbol{x}}_\alpha$ is a small quantity of a high order. Hence, its quadratic term can be ignored, and we have

$$(\boldsymbol{Q}\hat{\boldsymbol{x}}_\alpha, \Delta\hat{\boldsymbol{x}}_\alpha) = \frac{1}{2}(\hat{\boldsymbol{x}}_\alpha, \boldsymbol{Q}\hat{\boldsymbol{x}}_\alpha). \tag{15}$$

Let

$$\boldsymbol{k} \equiv \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}, \quad \boldsymbol{P}_{\mathbf{k}} \equiv \boldsymbol{I} - \boldsymbol{k}\boldsymbol{k}^\top (= \mathrm{diag}(1, 1, 0)). \tag{16}$$

Introducing Lagrange multipliers to (15) and the constraints

$$(\boldsymbol{k}, \Delta\hat{\boldsymbol{x}}_\alpha) = 0, \tag{17}$$

we obtain $\Delta\hat{\boldsymbol{x}}_\alpha$ as follows (see [15] for the derivation):

$$\Delta\hat{\boldsymbol{x}}_\alpha = \frac{\Big((\hat{\boldsymbol{x}}_\alpha, \boldsymbol{Q}\hat{\boldsymbol{x}}_\alpha) + 2(\boldsymbol{Q}\hat{\boldsymbol{x}}_\alpha, \tilde{\boldsymbol{x}}_\alpha)\Big)\boldsymbol{P}_{\mathrm{k}}\boldsymbol{Q}\hat{\boldsymbol{x}}_\alpha}{2(\boldsymbol{Q}\hat{\boldsymbol{x}}_\alpha, \boldsymbol{P}_{\mathrm{k}}\boldsymbol{Q}\hat{\boldsymbol{x}}_\alpha)} - \tilde{\boldsymbol{x}}_\alpha. \tag{18}$$

On substitution of this, the reprojection error of (12) now has the following form (see [15] for the derivation):

$$E = \frac{1}{4}\sum_{\alpha=1}^{N} \frac{\Big((\hat{\boldsymbol{x}}_\alpha, \boldsymbol{Q}\hat{\boldsymbol{x}}_\alpha) + 2(\boldsymbol{Q}\hat{\boldsymbol{x}}_\alpha, \tilde{\boldsymbol{x}}_\alpha)\Big)^2}{(\boldsymbol{Q}\hat{\boldsymbol{x}}_\alpha, \boldsymbol{P}_{\mathrm{k}}\boldsymbol{Q}\hat{\boldsymbol{x}}_\alpha)}. \tag{19}$$

We compute the matrix $\boldsymbol{Q}$ that minimizes this (we describe this shortly). Writing it as $\hat{\boldsymbol{Q}}$ and substituting it into (18), we obtain from (13) the solution

$$\hat{\hat{\boldsymbol{x}}}_\alpha = \boldsymbol{x}_\alpha - \frac{\Big((\hat{\boldsymbol{x}}_\alpha, \hat{\boldsymbol{Q}}\hat{\boldsymbol{x}}_\alpha) + 2(\hat{\boldsymbol{Q}}\hat{\boldsymbol{x}}_\alpha, \tilde{\boldsymbol{x}}_\alpha)\Big)\boldsymbol{P}_{\mathrm{k}}\hat{\boldsymbol{Q}}\hat{\boldsymbol{x}}_\alpha}{4(\hat{\boldsymbol{Q}}\hat{\boldsymbol{x}}_\alpha, \boldsymbol{P}_{\mathrm{k}}\hat{\boldsymbol{Q}}\hat{\boldsymbol{x}}_\alpha)}. \tag{20}$$

The resulting $\hat{\hat{\boldsymbol{x}}}_\alpha$ are a better approximation to $\bar{\boldsymbol{x}}_\alpha$ than $\hat{\boldsymbol{x}}_\alpha$. Rewriting $\hat{\hat{\boldsymbol{x}}}_\alpha$ as $\hat{\boldsymbol{x}}_\alpha$, we repeat this until the iterations converge. In the end, $\Delta\hat{\boldsymbol{x}}_\alpha$ in (15) becomes $\mathbf{0}$.

The above algorithm is greatly simplified by using the 6-D vector encoding of (5). Note that the definition of $\boldsymbol{\xi}_\alpha$ in (6) and $V_0[\boldsymbol{\xi}_\alpha]$ in (7) implies the identities

$$(\hat{\boldsymbol{x}}_\alpha, \hat{\boldsymbol{Q}}\hat{\boldsymbol{x}}_\alpha) + 2(\hat{\boldsymbol{Q}}\hat{\boldsymbol{x}}_\alpha, \tilde{\boldsymbol{x}}_\alpha) = \frac{(\boldsymbol{u}, \boldsymbol{\xi}_\alpha)}{f_0^2}, \tag{21}$$

$$(\hat{\boldsymbol{Q}}\boldsymbol{x}_\alpha, \boldsymbol{P}_{\mathrm{k}}\hat{\boldsymbol{Q}}\boldsymbol{x}_\alpha) = \frac{(\boldsymbol{u}, V_0[\boldsymbol{\xi}_\alpha]\boldsymbol{u})}{f_0^2}. \tag{22}$$

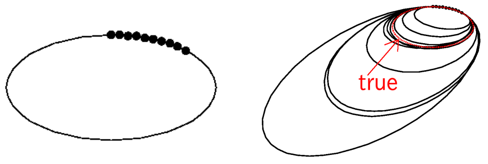
Since we define $\tilde{\boldsymbol{x}}_\alpha$ by (13), we obtain from (20) the update form in (8).

We now show how to minimize (19). Using the identities in (21) and (22), we can rewrite (19) as

$$E = \frac{1}{f_0^2}\sum_{\alpha=1}^{N} \frac{(\boldsymbol{u}, \boldsymbol{\xi}_\alpha)^2}{(\boldsymbol{u}, V_0[\boldsymbol{\xi}_\alpha]\boldsymbol{u})}, \tag{23}$$

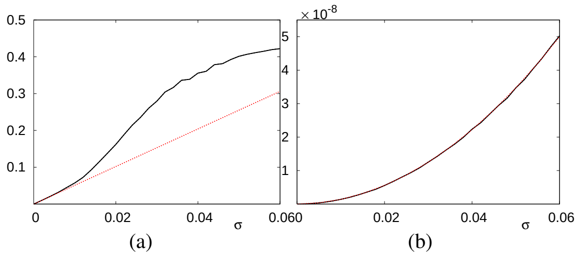
which is, aside from the constant $1/f_0^2$, the same as the function studied by Chojnacki et al. [6], who derived this as an approximation to the strict ML, also known as the *Sampson error* [10]. The only difference is that here the vector $\boldsymbol{\xi}_\alpha$ is defined by (6); the Sampson error is obtained if we let $\hat{x}_\alpha = x_\alpha$, $\hat{y}_\alpha = y_\alpha$, and $\tilde{x}_\alpha = \tilde{y}_\alpha = 0$ in (6). Nevertheless, we can apply the FNS of Chojnacki et al. [6], since the function form is identical. This is the core finding of this paper.

The slight difference between the original FNS and ours is that we replace the current $\boldsymbol{u}$ not by $\boldsymbol{u}'$ but by the "midpoint" $(\boldsymbol{u}' + \boldsymbol{u})/2$, which is normalized to $\mathcal{N}[\boldsymbol{u}' + \boldsymbol{u}]$. We have confirmed that this greatly improves the convergence of FNS, which sometimes oscillates in the presence of very large noise.



(a) (b)

Figure 1. (a) 10 points on an elliptic arc. (b) 10 samples of ellipse fitted by strict ML (σ = 0.05 pixels). The true ellipse is shown in dotted line.



(a) (b)

Figure 2. (a) Root-mean-square error. (b) Average reprojection error. The dashed lines are for the Sampson solution in both. The dotted lines show the KCR lower bound in (a) and the theoretical expectation in (b).

Here, we adopt the FNS for minimizing (23) for simplicity, but we could alternatively use the *HEIV* [17, 18], and the *projective Gauss-Newton iterations* [14], which can also minimize the Sampson error. Our contribution here is the finding that the method for minimizing the Sampson error can be used for *strict ML* if we introduce the new *intermediate variables* $\boldsymbol{\xi}_\alpha$ and $V_0[\boldsymbol{\xi}_\alpha]$ as in (6) and (7).

4. Performance Confirmation

Figure 1(a) shows 10 points on an elliptic arc. The major and minor axes of this ellipse are set to 100 and 50 pixels, respectively. We added independent Gaussian noise of mean 0 and standard deviation σ (pixels) to the x- and y-coordinates of the points and fitted an ellipse. We terminate the iterations when the change in the computed $\boldsymbol{u}$ is less than 10^{-6} in norm.

This is an extremely ill-conditioned example in the sense that even small noise will cause a large variation of the fitted ellipse. To visualize this, Fig. 1(b) shows 10 samples of the ellipse fitted by strict ML for σ = 0.05 pixels. The true ellipse is drawn in dotted line. This much deviations are inevitable even if the computation is done by strict ML.

Figure 2(a) plots, for each σ pixels, the following root-mean-square error of the fitted ellipse over 10,000 independent trials:

$$D = \sqrt{\frac{1}{10000}\sum_{a=1}^{10000}\|\boldsymbol{P}_{\mathbf{u}}\hat{\boldsymbol{u}}^{(a)}\|^2}, \quad \boldsymbol{P}_{\mathbf{u}} \equiv \boldsymbol{I} - \boldsymbol{u}\boldsymbol{u}^\top. \tag{24}$$

Here, $\hat{\boldsymbol{u}}^{(a)}$ represents the fitted ellipse in the (a)th trial, and $\boldsymbol{u}$ is its true value. Since $\boldsymbol{u}$ and $\hat{\boldsymbol{u}}^{(a)}$ are normalized to unit norm, we measure the the displacement $\|\boldsymbol{P}_{\mathbf{u}}\hat{\boldsymbol{u}}^{(a)}\|$ of $\hat{\boldsymbol{u}}^{(a)}$ orthogonal to $\boldsymbol{u}$. The dotted line is the theoretical accuracy limit called the *KCR lower bound* [5, 14]. We see that the solid lines (strict ML) and the dashed lines (Sampson) completely overlap in the graph (they differ only in the non-significant digits).

Figure 2(b) plots, for each σ (pixels), the reprojection error

$$E = \sum_{\alpha=1}^{N}\|\boldsymbol{x}_\alpha - \hat{\boldsymbol{x}}_\alpha\|^2 \tag{25}$$

averaged over 10,000 random trials. The reprojection error E is given by $\sum_{\alpha=1}^{N}\|\tilde{\boldsymbol{x}}_\alpha\|^2$ when the iterations in the main routine have converged. The reprojection error of the Sampson solution can also be computed, using the main routine, because *it can compute the orthogonal distances of the data points to a given ellipse if Step 4 is skipped*.

The dotted line in Fig. 2(b) show the theoretical expectation f_0^2E/σ^2: Since f_0^2E/σ^2 is subject to a χ^2 distribution with $N-5$ degrees of freedom to a first approximation [12], E has expectation $(N-5)\sigma^2/f_0^2$ to a first approximation. As we see, the Sampson and the strict ML solutions have practically the same reprojection error, and both almost coincides with the theoretical expectation, evidencing that the reprojection error is minimized indeed.

5. Concluding Remarks

We have presented a new algorithm for fitting an ellipse to a sequence of points based on the strict ML principle. We omitted the details here, but we also implemented alternative methods following existing strict ML principles [2, 8, 9, 20] and confirmed that our method indeed produces identical results. Although there is no accuracy gain, our algorithm is probably the simplest and the smallest. By numerical experiments, we have shown that the strict ML solution is practically identical to the Sampson solution, implying that strict ML solution also minimizes the Sampson error.

References

[1] F. J. Bookstein, Fitting conic sections to scattered data, *Comput. Graphics Image Process.*, **9** (1979) 56–71.

[2] A. Björck, *Numerical Methods for Least Squares Problems*, SIAM, Philadelphia, PA, U.S.A., 1996.

[3] J. Cabrera and P. Meer, Unbiased estimation of ellipses by bootstrapping, *IEEE Trans. Patt. Mach. Intell.*, **18**-7 (1996-7), 752–756.

[4] N. Chernov, On the convergence of fitting algorithms in computer vision, *J. Math. Imaging Vision.*, **27**-3 (2007-4), 231–239.

[5] N. Chernov and C. Lesort, Statistical efficiency of curve fitting algorithms, *Comput. Stat. Data Anal.*, **47**-4 (2004-11), 713–728.

[6] W. Chojnacki, M. J. Brooks, M.J., A. van den Hengel and D. Gawley, On the fitting of surfaces to data with covariances, *IEEE Trans. Patt. Anal. Mach. Intell.*, **22**-11 (2000-11), 1294–1303.

[7] A. Fitzgibbon, M. Pilu and R. B. Fisher, Direct least square fitting of ellipses, *IEEE Trans. Patt. Anal. Mach. Intell.*, **21**-5 (1999-5), 476–480.

[8] W. Gander, H. Golub, and R. Strebel, Least-squares fitting of circles and ellipses, *BIT*, **34**-4 (1994-12), 558–578.

[9] M. Harker and P. O'Leary, First order geometric distance (The myth of Sampsonus), *Proc. 17th Brit. Mach. Vis. Conf.*, September 2006, Edinburgh, U.K., Vol. 1, pp. 87–96.

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

[11] K. Kanatani, *Geometric Computation for Machine Vision*, Oxford University Press, Oxford, U.K., 1993.

[12] K. Kanatani, *Statistical Optimization for Geometric Computation: Theory and Practice*, Elsevier Science, Amsterdam, The Netherlands, 1996; reprinted, Dover, New York (2005)

[13] K. Kanatani and N. Ohta, Automatic detection of circular objects by ellipse growing, *Int. J. Image Graphics*, **4**-1 (2004-1), 35–50.

[14] K. Kanatani, K., Sugaya, Performance evaluation of iterative geometric fitting algorithms, *Comp. Stat. Data Anal.*, **52**-2 (2007-10), 1208–1222.

[15] K. Kanatani and Y. Sugaya, Compact algorithm for truly optimal ellipse fitting *Proc. 10th Symp. Sensing via Imaging Information*, June 2008, Yokohama, Japan, pp. IN1-12-1–IN1-12-7.

[16] D. Keren, D. Cooper and J. Subrahmonia, Describing complicated objects by implicit polynomials, *IEEE Trans. Patt. Anal. Mach. Intell.*, **16**-1 (1994-1), 38–53.

[17] Y. Leedan and P. Meer, Heteroscedastic regression in computer vision: Problems with bilinear constraint, *Int. J. Comput. Vis.*, **37**-2 (2000-6), 127–150.

[18] J. Matei and P. Meer, Estimation of nonlinear errors-in-variables models for computer vision applications, *IEEE Trans. Patt. Anal. Mach. Intell.*, **28**-10 (2006-10), 1537–1552.

[19] P. D. Sampson, Fitting conic sections to "very scattered" data: An iterative refinement of the Bookstein algorithm, *Comput. Graphics Image Process.*, **18** (1982), 97–108.

[20] P. Sturm and P. Gargallo, Conic fitting using the geometric distance, *Proc. 8th Asian Conf. Comput. Vision*, November 2007, Tokyo, Japan, Vol. 2, pp. 784–795.

[21] G. Taubin, Estimation of planar curves, surfaces, and non-planar space curves defined by implicit equations with applications to edge and rage image segmentation, *IEEE Trans. Patt. Anal. Mach. Intell.*, **13**-11 (1991-11), 1115–1138.

[22] Z. Zhang, Parameter estimation techniques: A tutorial with application to conic fitting, *Image Vision Comput.*, **15**-No. 1 (1997-1), 59–76.