Guaranteed Ellipse Fitting with the Sampson Distance

Zygmunt L. Szpak, Wojciech Chojnacki, and Anton van den Hengel

School of Computer Science, The University of Adelaide, SA 5005, Australia
{zygmunt.szpak,wojciech.chojnacki,anton.vandenhengel}@adelaide.edu.au

Abstract. When faced with an ellipse fitting problem, practitioners frequently resort to algebraic ellipse fitting methods due to their simplicity and efficiency. Currently, practitioners must choose between algebraic methods that guarantee an ellipse fit but exhibit high bias, and geometric methods that are less biased but may no longer guarantee an ellipse solution. We address this limitation by proposing a method that strikes a balance between these two objectives. Specifically, we propose a fast stable algorithm for fitting a guaranteed ellipse to data using the Sampson distance as a data-parameter discrepancy measure. We validate the stability, accuracy, and efficiency of our method on both real and synthetic data. Experimental results show that our algorithm is a fast and accurate approximation of the computationally more expensive orthogonal-distance-based ellipse fitting method. In view of these qualities, our method may be of interest to practitioners who require accurate and guaranteed ellipse estimates.

1 Introduction

In computer vision, image processing, and pattern recognition, the fitting of ellipses to data is a frequently encountered challenge. For example, ellipse fitting is used in the calibration of catadioptric cameras [9], segmentation of touching cells [3, 28] or grains [30], in strain analysis [20], and in the study of galaxies in astrophysics [26]. Given the rich field of applications, it is not surprising that numerous ellipse fitting algorithms have been reported in the literature [1, 10–12, 16, 21, 24, 25, 27, 29, 31]. What then is the purpose of yet another ellipse fitting algorithm? To address this question and motivate the principal contribution of this work, we make the following observations.

Existing ellipse fitting algorithms can be cast into two categories: (1) methods which minimise a *geometric* error, and (2) methods which minimise an *algebraic* error. Under the assumption of identical independent homogeneous Gaussian noise in the data, minimising the geometric error corresponds to the maximum likelihood estimation of the ellipse parameters, and is equivalent to minimising the sum of the orthogonal distances between data points and the ellipse.[1] Due

[1] Minimisation of the sum of the orthogonal distance between data points and a geometric primitive is frequently referred to as orthogonal distance regression.

A. Fitzgibbon et al. (Eds.): ECCV 2012, Part V, LNCS 7576, pp. 87–100, 2012.

to the non-linear nature of the ellipse fitting problem, the traditional iterative scheme for minimising the geometric error is rather complicated. It requires, for each iteration, computing the roots of a quartic polynomial as an intermediate step [18,31]. While with the choice of a suitable parametrisation the root finding step can be avoided [11,27], the iterative schemes, nonetheless, remain elaborate.

On the other hand, ellipse fitting methods based on minimizing an algebraic error are conceptually simple, easy to implement, and efficient. Unfortunately, they are plagued by two problems: they either do not guarantee that the estimate will be an ellipse (sometimes producing hyperbolas or parabolas instead), or they exhibit high bias resulting in a poor fit in some instances.

The first algebraic method to guarantee an ellipse fit was presented by Fitzgibbon *et al.* [10]. Due to its simplicity and stability, it has quickly become de facto standard for ellipse fitting. However, as the authors noted in their original work, the method suffered from substantial bias problems. Other algebraic fitting methods have focused on minimizing the bias, but no longer guaranteed that the resulting estimate will be an ellipse [2, 16].

Our contribution lies in devising an iterative guaranteed ellipse fitting method that strikes a balance between the accuracy of a geometric fit, and the stability and simplicity of an algebraic fit. For a data-parameter discrepancy measure, it utilises the Sampson distance which has generally been accepted as an accurate approximation of the geometric error that leads to good estimation results in moderate noise scenarios [15]. Remarkably, our proposed algorithm is fast and stable, and always returns an ellipse as a fit to data.

2 Background

A *conic* is the locus of solutions $\mathbf{x} = [m_1, m_2]^\mathsf{T}$ in the Euclidean plane $\mathbb{R}^2$ of a quadratic equation

$$am_1^2 + bm_1m_2 + cm_2^2 + dm_1 + em_2 + f = 0, \tag{1}$$

where a, b, c, d, e, f are real numbers such that $a^2 + b^2 + c^2 > 0$. With $\boldsymbol{\theta} = [a, b, c, d, e, f]^\mathsf{T}$ and $\mathbf{u}(\mathbf{x}) = [m_1^2, m_1m_2, m_2^2, m_1, m_2, 1]^\mathsf{T}$, equation (1) can equivalently be written as

$$\boldsymbol{\theta}^T\mathbf{u}(\mathbf{x}) = 0. \tag{2}$$

Any multiple of $\boldsymbol{\theta}$ by a non-zero number corresponds to one and the same conic. A conic is either an *ellipse*, or a *parabola*, or a *hyperbola* depending on whether the *discriminant* $\Delta = b^2 - 4ac$ is negative, zero, or positive. The condition $\Delta < 0$ characterising the ellipses can alternatively be written as

$$\boldsymbol{\theta}^\mathsf{T}\mathbf{F}\boldsymbol{\theta} > 0, \tag{3}$$

where

$$\mathbf{F} = \begin{bmatrix} 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Of all conics, here we shall specifically be concerned with ellipses.

The task of fitting an ellipse to a set of points $\mathbf{x}_1, \ldots, \mathbf{x}_N$ requires a meaningful *cost function* that characterises the extent to which any particular $\boldsymbol{\theta}$ fails to satisfy the system of copies of equation (2) associated with $\mathbf{x} = \mathbf{x}_n$, $n = 1, \ldots, N$. Once a cost function is selected, the corresponding ellipse fit results from minimising the cost function subject to the constraint (3).

Effectively, though not explicitly, Fitzgibbon *et al.* [10] proposed to use for ellipse fitting the *direct ellipse fitting* cost function

$$J_{\mathrm{DIR}}(\boldsymbol{\theta}) = \frac{\boldsymbol{\theta}^{\mathsf{T}}\mathbf{A}\boldsymbol{\theta}}{\boldsymbol{\theta}^{\mathsf{T}}\mathbf{F}\boldsymbol{\theta}},$$

where $\mathbf{A} = \sum_{n=1}^{N} \mathbf{u}(\mathbf{x}_n)\mathbf{u}(\mathbf{x}_n)^{\mathsf{T}}$. The minimiser $\widehat{\boldsymbol{\theta}}_{\mathrm{DIR}}$ of J_{DIR} is the same as the solution of the problem

$$\min_{\boldsymbol{\theta}} \boldsymbol{\theta}^{\mathsf{T}}\mathbf{A}\boldsymbol{\theta} \quad \text{subject to} \quad \boldsymbol{\theta}^{\mathsf{T}}\mathbf{F}\boldsymbol{\theta} = 1, \tag{4}$$

and it is in this form that $\widehat{\boldsymbol{\theta}}_{\mathrm{DIR}}$ was originally introduced. The representation of $\widehat{\boldsymbol{\theta}}_{\mathrm{DIR}}$ as a solution of the problem (4) makes it clear that $\widehat{\boldsymbol{\theta}}_{\mathrm{DIR}}$ is always an ellipse. Extending the work of Fitzgibbon *et al.*, Halíř and Flusser [12] introduced a numerically stable algorithm for calculating $\widehat{\boldsymbol{\theta}}_{\mathrm{DIR}}$.

Another cost function for ellipse fitting, and more generally for conic fitting, was first proposed by Sampson [25] and next popularised, in a broader context, by Kanatani [14]. It is variously called the *Sampson*, *gradient-weighted*, and *approximated maximum likelihood* (AML) distance or cost function, and takes the form

$$J_{\mathrm{AML}}(\boldsymbol{\theta}) = \sum_{n=1}^{N} \frac{\boldsymbol{\theta}^{\mathsf{T}}\mathbf{A}_n\boldsymbol{\theta}}{\boldsymbol{\theta}^{\mathsf{T}}\mathbf{B}_n\boldsymbol{\theta}}$$

with $\mathbf{A}_n = \mathbf{u}(\mathbf{x}_n)\mathbf{u}(\mathbf{x}_n)^{\mathsf{T}}$ and $\mathbf{B}_n = \partial_{\mathbf{x}}\mathbf{u}(\mathbf{x}_n)\boldsymbol{\Lambda}_{\mathbf{x}_n}\partial_{\mathbf{x}}\mathbf{u}(\mathbf{x}_n)^{\mathsf{T}}$ for each $n = 1, \ldots, N$. Here, for any length 2 vector $\mathbf{y}$, $\partial_{\mathbf{x}}\mathbf{u}(\mathbf{y})$ denotes the 6×2 matrix of the partial derivatives of the function $\mathbf{x} \mapsto \mathbf{u}(\mathbf{x})$ evaluated at $\mathbf{y}$, and, for each $n = 1, \ldots, N$, $\boldsymbol{\Lambda}_{\mathbf{x}_n}$ is a 2×2 symmetric *covariance matrix* describing the uncertainty of the data point $\mathbf{x}_n$ [4,7,14]. The function J_{AML} is a first-order approximation of a genuine maximum likelihood cost function J_{ML} which can be evolved based on the *Gaussian model of errors* in data in conjunction with the *principle of maximum likelihood*. In the case when identical independent homogeneous Gaussian noise corrupts the data points, J_{ML} reduces to—up to a numeric constant depending on the noise level—to the sum of orthogonal distances of the data points and an

ellipse. With a point $\boldsymbol{\theta}$ in the search space of all length-6 vectors termed *feasible* if $\boldsymbol{\theta}$ satisfies the ellipticity condition (3), the *approximated maximum likelihood estimate* $\widehat{\boldsymbol{\theta}}_{\mathrm{AML}}$ is, by definition, the minimiser of J_{AML} selected from among all feasible points.

3 Constrained Optimisation

3.1 Merit Function

For optimising J_{AML} so that the ellipticity constraint is satisfied, we develop a variant of the *barrier method*. We form a *merit function*

$$P(\boldsymbol{\theta}, \alpha) = J_{\mathrm{AML}}(\boldsymbol{\theta}) + \alpha g(\boldsymbol{\theta})$$

in which the *barrier function* g multiplied by the positive *homotopy parameter* α is added to the cost function J_{AML} to penalise the violation of the ellipticity condition. The barrier function is taken in the form

$$g(\boldsymbol{\theta}) = \frac{\|\boldsymbol{\theta}\|^4}{(\boldsymbol{\theta}^{\mathsf{T}}\mathbf{F}\boldsymbol{\theta})^2},$$

with $\|\boldsymbol{\theta}\| = (\sum_{j=1}^{6} \theta_j^2)^{1/2}$. This function tends to infinity as $\boldsymbol{\theta}$ approaches the parabolic boundary $\{\boldsymbol{\theta} \mid \boldsymbol{\theta}^{\mathsf{T}}\mathbf{F}\boldsymbol{\theta} = 0\}$ between the feasible elliptic region $\{\boldsymbol{\theta} \mid \boldsymbol{\theta}^{\mathsf{T}}\mathbf{F}\boldsymbol{\theta} > 0\}$ and the infeasible hyperbolic $\{\boldsymbol{\theta} \mid \boldsymbol{\theta}^{\mathsf{T}}\mathbf{F}\boldsymbol{\theta} < 0\}$ region of the search space. The significance of g is that it ensures that if P is optimised in sufficiently short steps starting from a feasible point, then any local miminiser reached on the way is feasible; and if the homotopy parameter is small enough, this local minimiser is a good approximation of a local minimiser of J_{AML}.

The standard mode of use of P, and the likes of it, is as follows. Initially, a relatively large value of α, α_1, is chosen and, starting from some feasible point $\boldsymbol{\theta}_0$, $P(\cdot, \alpha_1)$ is optimised by an iterative process; this results in a feasible local minimiser $\boldsymbol{\theta}_1$. Next α_1 is replaced by a smaller value of α, α_2, and a feasible local minimiser $\boldsymbol{\theta}_2$ of $P(\cdot, \alpha_2)$ is derived iteratively starting from $\boldsymbol{\theta}_1$. By continuing this process with a whole sequence of values of α, (α_k), chosen so that $\lim_{k\to\infty} \alpha_k = 0$, a sequence $(\boldsymbol{\theta}_k)$ of feasible points is generated. The limit of this sequence is a feasible local minimiser of J_{AML}.

Here we shall use P in a different mode. We take α to be a very small number (of the order of 10^{-15}) from the outset and keep it fixed. We then optimise $P(\cdot, \alpha)$ by a specific iterative process that we shall describe in detail later, using a direct ellipse estimate as a seed. If the minimiser turns out to be feasible (representing an ellipse), then we declare it to be a solution of our problem. If the minimiser is infeasible (representing a hyperbola), then we go back to the immediate predecessor of the iterate that constitutes the minimiser and, by running another iterative process, correct it to a new feasible point, which we subsequently take for a solution of our problem. The additional iterative process in this last step could in theory be avoided and replaced by a refined version of the original process, but at the expense of slower speed—with it, the overall procedure converges very quickly.

3.2 Optimisation Algorithm

To avoid possible problems with convergence for large data noise, we adopt the Levenberg–Marquardt (LM) algorithm as a main ingredient of our algorithm for optimising the merit function. To describe the specifics of the LM scheme, we introduce $\mathbf{r}(\boldsymbol{\theta}) = [r_1(\boldsymbol{\theta}), \ldots, r_{N+1}(\boldsymbol{\theta})]^\mathsf{T}$ such that

$$r_n(\boldsymbol{\theta}) = \left(\frac{\boldsymbol{\theta}^\mathsf{T}\mathbf{A}_n\boldsymbol{\theta}}{\boldsymbol{\theta}^\mathsf{T}\mathbf{B}_n\boldsymbol{\theta}}\right)^{1/2}$$

for each $n = 1, \ldots, N$, and

$$r_{N+1}(\boldsymbol{\theta}) = \alpha^{1/2}\frac{\|\boldsymbol{\theta}\|^2}{\boldsymbol{\theta}^\mathsf{T}\mathbf{F}\boldsymbol{\theta}}.$$

With this definition in place, the function P can be given the convenient least-squares form

$$P(\boldsymbol{\theta}) = \mathbf{r}(\boldsymbol{\theta})^\mathsf{T}\mathbf{r}(\boldsymbol{\theta}) = \|\mathbf{r}(\boldsymbol{\theta})\|^2.$$

A straightforward computation shows that the first-derivative row vectors of the components of $\mathbf{r}(\boldsymbol{\theta})$ take the form

$$(\partial_{\boldsymbol{\theta}} r_n(\boldsymbol{\theta}))^\mathsf{T} = r_n^{-1}(\boldsymbol{\theta})\mathbf{X}_n(\boldsymbol{\theta})\boldsymbol{\theta},$$
$$\mathbf{X}_n(\boldsymbol{\theta}) = \frac{\mathbf{A}_n}{\boldsymbol{\theta}^\mathsf{T}\mathbf{B}_n\boldsymbol{\theta}} - \frac{\boldsymbol{\theta}^\mathsf{T}\mathbf{A}_n\boldsymbol{\theta}}{(\boldsymbol{\theta}^\mathsf{T}\mathbf{B}_n\boldsymbol{\theta})^2}\mathbf{B}_n$$

for each $n = 1, \ldots, N$, and

$$(\partial_{\boldsymbol{\theta}} r_{N+1}(\boldsymbol{\theta}))^\mathsf{T} = 2\alpha^{1/2}\mathbf{Y}(\boldsymbol{\theta})\boldsymbol{\theta},$$
$$\mathbf{Y}(\boldsymbol{\theta}) = \frac{\mathbf{I}_6}{\boldsymbol{\theta}^\mathsf{T}\mathbf{F}\boldsymbol{\theta}} - \frac{\|\boldsymbol{\theta}\|^2}{(\boldsymbol{\theta}^\mathsf{T}\mathbf{F}\boldsymbol{\theta})^2}\mathbf{F},$$

where $\mathbf{I}_6$ denotes the 6×6 identity matrix. With the Jacobian matrix $\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta})$ of $\mathbf{r}(\boldsymbol{\theta})$ expressed as $\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}) = [(\partial_{\boldsymbol{\theta}} r_1(\boldsymbol{\theta}))^\mathsf{T}, \ldots, (\partial_{\boldsymbol{\theta}} r_{N+1}(\boldsymbol{\theta}))^\mathsf{T}]^\mathsf{T}$ and with $\mathbf{B}(\boldsymbol{\theta}) = \sum_{n=1}^{N} r_n(\boldsymbol{\theta})\partial^2_{\boldsymbol{\theta}\boldsymbol{\theta}} r_n(\boldsymbol{\theta})$, the first-derivative row vector of $P(\boldsymbol{\theta})$ and Hessian matrix $\partial^2_{\boldsymbol{\theta}\boldsymbol{\theta}}P(\boldsymbol{\theta})$ of $P(\boldsymbol{\theta})$ are given by $\partial_{\boldsymbol{\theta}}P(\boldsymbol{\theta}) = 2(\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}))^\mathsf{T}\mathbf{r}(\boldsymbol{\theta})$ and $\partial^2_{\boldsymbol{\theta}\boldsymbol{\theta}}P(\boldsymbol{\theta}) = 2\big((\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}))^\mathsf{T}\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}) + \mathbf{B}(\boldsymbol{\theta})\big)$. The LM scheme uses the Gauss–Newton approximation $2(\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}))^\mathsf{T}\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta})$ of $\partial^2_{\boldsymbol{\theta}\boldsymbol{\theta}}P(\boldsymbol{\theta})$ with the second-derivative matrix $\mathbf{B}(\boldsymbol{\theta})$ neglected. The algorithm iteratively improves on a starting estimate $\boldsymbol{\theta}_0$ by constructing new approximations with the aid of the update rule

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \boldsymbol{\eta}_k, \tag{5}$$
$$\boldsymbol{\eta}_k = -\big((\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}_k))^\mathsf{T}\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}_k) + \lambda_k\mathbf{I}_6\big)^{-1}(\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}_k))^\mathsf{T}\mathbf{r}(\boldsymbol{\theta}_k), \tag{6}$$

where λ_k is a non-negative scalar that dynamically changes from step to step [23].

On rare occasions LM can overshoot and yield an infeasible point $\boldsymbol{\theta}_{k+1}$ representing a hyperbola. When this happens, our ultimate algorithm, part of which

is LM, backtracks to the previous iterate $\boldsymbol{\theta}_k$, which represents an ellipse close the parabolic boundary, and improves on it by initiating another iterative process. This new process ensures that the estimate never leaves the feasible region. The modified update rule is

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \delta_l \boldsymbol{\xi}_k,$$

where δ_l plays the role of a step length chosen so that, on the one hand, $\boldsymbol{\theta}_{k+1}$ still represents an ellipse, and, on the other hand, the objective function is sufficiently reduced. A suitable value for δ_l is found using an inexact line search. The direction $\boldsymbol{\xi}_k$ is taken to be

$$\boldsymbol{\xi}_k = -\big((\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}_k))^{\mathsf{T}}\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}_k)\big)^{+}(\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}_k))^{\mathsf{T}}\mathbf{r}(\boldsymbol{\theta}_k),$$

where, for a given matrix $\mathbf{A}$, $\mathbf{A}^{+}$ denotes the Moore–Penrose pseudo-inverse of $\mathbf{A}$. The choice of $\boldsymbol{\xi}_k$ is motivated by the formula

$$\boldsymbol{\xi}_k = -\lim_{\lambda\to 0}\big((\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}_k))^{\mathsf{T}}\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}_k) + \lambda\mathbf{I}_6\big)^{-1}(\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}_k))^{\mathsf{T}}\mathbf{r}(\boldsymbol{\theta}_k)$$

showing that $\boldsymbol{\xi}_k$ is the limit of LM increments as the damping parameter λ approaches zero, in the scenario whereby LM acts venturously. The modified update rule is a fast replacement for LM, which if it were not to overshoot, would have to operate conservatively, taking many small steps in gradient-descent-like mode.

The details of our overall optimisation procedure are given in Algorithms 3.1, 3.2, and 3.3.

We close this section by remarking that the implementation of (6) requires some care. To explain the underlying problem, we partition $\mathbf{r}(\boldsymbol{\theta})$ as $\mathbf{r}(\boldsymbol{\theta}) = [\mathbf{r}'(\boldsymbol{\theta})^{\mathsf{T}}, r_{N+1}(\boldsymbol{\theta})]^{\mathsf{T}}$, where $\mathbf{r}'(\boldsymbol{\theta})$ is the length-N vector comprising the first N components of $\mathbf{r}(\boldsymbol{\theta})$. Then, accordingly, we write $(\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}))^{\mathsf{T}}\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta})$ as

$$(\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}))^{\mathsf{T}}\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}) = (\partial_{\boldsymbol{\theta}}\mathbf{r}'(\boldsymbol{\theta}))^{\mathsf{T}}\partial_{\boldsymbol{\theta}}\mathbf{r}'(\boldsymbol{\theta}) + (\partial_{\boldsymbol{\theta}}r_{N+1}(\boldsymbol{\theta}))^{\mathsf{T}}\partial_{\boldsymbol{\theta}}r_{N+1}(\boldsymbol{\theta})$$

If $\boldsymbol{\theta}_k$ is close to the parabolic boundary, then $\|\partial_{\boldsymbol{\theta}}r_{N+1}(\boldsymbol{\theta}_k)\|$ assumes large values and the rank-1 matrix $(\partial_{\boldsymbol{\theta}}r_{N+1}(\boldsymbol{\theta}_k))^{\mathsf{T}}\partial_{\boldsymbol{\theta}}r_{N+1}(\boldsymbol{\theta}_k)$, with norm of the order of $(\boldsymbol{\theta}_k^{\mathsf{T}}\mathbf{F}\boldsymbol{\theta}_k)^{-4}$, increasingly dominates $(\partial_{\boldsymbol{\theta}}\mathbf{r}'(\boldsymbol{\theta}_k))^{\mathsf{T}}\partial_{\boldsymbol{\theta}}\mathbf{r}'(\boldsymbol{\theta}_k) + \lambda_k\mathbf{I}_6$ so that $(\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}_k))^{\mathsf{T}}\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}_k) + \lambda_k\mathbf{I}_6$ has a large condition number. This translates into the computation of $\boldsymbol{\eta}_k$ becoming ill-conditioned. To remedy the situation, we introduce a new dummy length-6 variable $\boldsymbol{\xi}_k$ and reformulate (6) as follows

$$\begin{bmatrix} (\partial_{\boldsymbol{\theta}}\mathbf{r}'(\boldsymbol{\theta}_k))^{\mathsf{T}}\partial_{\boldsymbol{\theta}}\mathbf{r}'(\boldsymbol{\theta}_k) + \lambda_k\mathbf{I}_6 & (\boldsymbol{\theta}_k^{\mathsf{T}}\mathbf{F}\boldsymbol{\theta}_k)^4(\partial_{\boldsymbol{\theta}}r_{N+1}(\boldsymbol{\theta}_k))^{\mathsf{T}}\partial_{\boldsymbol{\theta}}r_{N+1}(\boldsymbol{\theta}_k) \\ \mathbf{I}_6 & -(\boldsymbol{\theta}_k^{\mathsf{T}}\mathbf{F}\boldsymbol{\theta}_k)^4\mathbf{I}_6 \end{bmatrix} \begin{bmatrix} \boldsymbol{\eta}_k \\ \boldsymbol{\xi}_k \end{bmatrix} = \begin{bmatrix} -(\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}_k))^{\mathsf{T}}\mathbf{r}(\boldsymbol{\theta}_k) \\ \mathbf{0} \end{bmatrix}. \quad (7)$$

The advantage of this approach is that the system (7) is much better conditioned and leads to numerical results of higher precision.

Algorithm 3.1: GuaranteedEllipseFit($\widehat{\boldsymbol{\theta}}_{\text{DIR}}$, data_points)

main
comment: initialise variables
keep_going $\leftarrow$ **true** ; use_pseudo-inverse $\leftarrow$ **false** ;
$\lambda \leftarrow 0.01$; $\mathtt{k} \leftarrow 1$; $\nu \leftarrow 1.2$; $\boldsymbol{\theta} \leftarrow \widehat{\boldsymbol{\theta}}_{\text{DIR}}$; $\gamma \leftarrow 0.00005$

comment: a data structure $\Omega = \{\mathbf{r}(\boldsymbol{\theta}), \partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}), \mathbf{H}(\boldsymbol{\theta}), \mathtt{cost}_{\boldsymbol{\theta}}, \boldsymbol{\theta}, \lambda, \nu, \gamma, \boldsymbol{\theta}\_\mathtt{wasUpdated}\}$ is used to pass parameters between functions
while keep_going **and** $k <$ maxIter **do**
- $\Omega_k.\mathbf{r}(\boldsymbol{\theta}) \leftarrow [\mathtt{r}_1(\boldsymbol{\theta}), \ldots, \mathtt{r}_{\mathtt{N}+1}(\boldsymbol{\theta})]^{\mathsf{T}}$ .. (residuals computed on data_points)
- $\Omega_k.\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}) \leftarrow [\partial_{\boldsymbol{\theta}}\mathtt{r}_1(\boldsymbol{\theta}), \ldots, \partial_{\boldsymbol{\theta}}\mathtt{r}_{\mathtt{N}+1}(\boldsymbol{\theta})]^{\mathsf{T}}$ ..... (Jacobian matrix)
- $\Omega_k.\mathbf{H}(\boldsymbol{\theta}) \leftarrow (\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta}))^{\mathsf{T}}\partial_{\boldsymbol{\theta}}\mathbf{r}(\boldsymbol{\theta})$ .... (approximate halved Hessian matrix)
- $\Omega_k.\mathtt{cost}_{\boldsymbol{\theta}} \leftarrow \mathbf{r}(\boldsymbol{\theta})^{\mathsf{T}}\mathbf{r}(\boldsymbol{\theta})$ ..... (current cost)
- $\Omega_k.\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}$ ..... (current parameter estimate)
- $\Omega_k.\lambda \leftarrow \lambda$ ..... (damping parameter)
- $\Omega_k.\nu \leftarrow \nu$ ..... (damping parameter multiplier)
- $\Omega_k.\gamma \leftarrow \gamma$ ..... (used in line search)
- $\Omega_k.\boldsymbol{\theta}\_\mathtt{wasUpdated} \leftarrow$ **false** ..... (indicates if $\boldsymbol{\theta}$ was modified or not)
- **if not** use_pseudo-inverse
  - **then** $\Omega_{k+1} \leftarrow$ LevenbergMarquardtStep(Ω_k)
  - **else** $\Omega_{k+1} \leftarrow$ LineSearchStep(Ω_k)
- **if** $\Omega_{\mathtt{k}+1}.\boldsymbol{\theta}^{\mathsf{T}}\mathbf{F}\Omega_{\mathtt{k}+1}.\boldsymbol{\theta} \leq 0$
  - **then**
    - **comment:** revert to previous iterate which was an ellipse
    - use_pseudo-inverse $\leftarrow$ **true**
    - $\Omega_{\mathtt{k}+1}.\boldsymbol{\theta} \leftarrow \Omega_{\mathtt{k}}.\boldsymbol{\theta}$
    - **if** $\mathtt{k} > 1$ **then** $\Omega.\boldsymbol{\theta}_{\mathtt{k}} \leftarrow \Omega_{\mathtt{k}-1}.\boldsymbol{\theta}$
  - **else if** $\min_{\varepsilon=\pm 1} \left\| \frac{\Omega_{\mathtt{k}+1}.\boldsymbol{\theta}}{\|\Omega_{\mathtt{k}+1}.\boldsymbol{\theta}\|} + \varepsilon \frac{\Omega_{\mathtt{k}}.\boldsymbol{\theta}}{\|\Omega_{\mathtt{k}}.\boldsymbol{\theta}\|} \right\| < \mathtt{tol}_{\boldsymbol{\theta}}$ **and** $\Omega_{k+1}.\boldsymbol{\theta}\_\mathtt{wasUpdated}$ **then** keep_going $\leftarrow$ **false**
  - **else if** $|\Omega_{\mathtt{k}+1}.\mathtt{cost}_{\boldsymbol{\theta}} - \Omega_{\mathtt{k}}.\mathtt{cost}_{\boldsymbol{\theta}}| < \mathtt{tol}_{\mathtt{cost}}$ **and** $\Omega_{k+1}.\boldsymbol{\theta}\_\mathtt{wasUpdated}$ **then** keep_going $\leftarrow$ **false**
  - **else if** $\|\Omega_{\mathtt{k}+1}.\Delta_{\boldsymbol{\theta}}\| < \mathtt{tol}_{\Delta}$ **then** keep_going $\leftarrow$ **false**
- $\mathtt{k} \leftarrow \mathtt{k} + 1$

output ($\Omega_{\mathtt{k}}.\boldsymbol{\theta}$)

4 Experimental Results

To investigate the stability and accuracy of our algorithm, we conducted experiments on both synthetic and real data. We compared our results with the maximum likelihood [31] and direct ellipse fitting [10, 12] estimates, which represent the gold standard and baseline methods, respectively. Both the maximum likelihood and our proposed approximate maximum likelihood method were seeded with the result of the direct ellipse fit. All estimation methods operated on Hartley-normalised data points [6].

4.1 Synthetic Data

For our synthetic experiments, we fixed the image size to 600×600 pixels and generated ellipses with random centers, major/minor axes, and orientations of the axes. Figure 1 shows a sample of ten such randomly generated ellipses. We then conducted four simulation experiments, which focused on different segments

Algorithm 3.2: LevenbergMarquardtStep(Ω)

comment: compute two potential updates based on different weightings of the identity matrix

$\Delta_{\mathrm{a}} \leftarrow [\Omega.\mathbf{H}(\boldsymbol{\theta}) + \Omega.\lambda \mathbf{I}_6)]^{-1} (\Omega.\partial_{\boldsymbol{\theta}} \mathbf{r}(\boldsymbol{\theta}))^{\mathsf{T}} \Omega.\mathbf{r}(\boldsymbol{\theta})$
$\boldsymbol{\theta}_{\mathrm{a}} \leftarrow \Omega.\boldsymbol{\theta} - \Delta_{\mathrm{a}}$
$\Delta_{\mathrm{b}} \leftarrow [\Omega.\mathbf{H}(\boldsymbol{\theta}) + (\Omega.\lambda)(\Omega.\nu)^{-1} \mathbf{I}_6)]^{-1} (\Omega.\partial_{\boldsymbol{\theta}} \mathbf{r}(\boldsymbol{\theta}))^{\mathsf{T}} \Omega.\mathbf{r}(\boldsymbol{\theta})$
$\boldsymbol{\theta}_{\mathrm{b}} \leftarrow \Omega.\boldsymbol{\theta} - \Delta_{\mathrm{b}}$

comment: compute new residuals and costs based on these updates

$\mathtt{cost}_{\boldsymbol{\theta}_{\mathrm{a}}} \leftarrow \mathbf{r}(\boldsymbol{\theta}_{\mathrm{a}})^{\mathsf{T}} \mathbf{r}(\boldsymbol{\theta}_{\mathrm{a}})$
$\mathtt{cost}_{\boldsymbol{\theta}_{\mathrm{b}}} \leftarrow \mathbf{r}(\boldsymbol{\theta}_{\mathrm{b}})^{\mathsf{T}} \mathbf{r}(\boldsymbol{\theta}_{\mathrm{b}})$

comment: determine appropriate damping and if possible select an update

if $\mathtt{cost}_{\boldsymbol{\theta}_{\mathrm{a}}} \geq \Omega.\mathtt{cost}_{\boldsymbol{\theta}}$ **and** $\mathtt{cost}_{\boldsymbol{\theta}_{\mathrm{b}}} \geq \Omega.\mathtt{cost}_{\boldsymbol{\theta}}$

then
- $\Omega.\boldsymbol{\theta}\_\mathtt{wasUpdated} \leftarrow$ **false** ..... (neither potential update reduced the cost)
- $\Omega.\boldsymbol{\theta} \leftarrow \Omega.\boldsymbol{\theta}$ ..... (no change in parameters)
- $\Omega.\Delta_{\boldsymbol{\theta}} \leftarrow \Omega.\Delta_{\boldsymbol{\theta}}$ ..... (no change in step direction)
- $\Omega.\mathtt{cost}_{\boldsymbol{\theta}} \leftarrow \Omega.\mathtt{cost}_{\boldsymbol{\theta}}$ ..... (no change in cost)
- $\Omega.\lambda \leftarrow (\Omega.\lambda)(\Omega.\nu)$ ..... (next iteration add more of the identity matrix)

else if $\mathtt{cost}_{\boldsymbol{\theta}_{\mathrm{b}}} < \Omega.\mathtt{cost}_{\boldsymbol{\theta}}$

then
- $\Omega.\boldsymbol{\theta}\_\mathtt{wasUpdated} \leftarrow$ **true** ..... (update 'b' reduced the cost function)
- $\Omega.\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_{\mathrm{b}}$ ..... (choose update 'b')
- $\Omega.\Delta_{\boldsymbol{\theta}} \leftarrow \Delta_{\mathrm{b}}$ ..... (store the step direction)
- $\Omega.\mathtt{cost}_{\boldsymbol{\theta}} \leftarrow \mathtt{cost}_{\boldsymbol{\theta}_{\mathrm{b}}}$ ..... (store the current cost)
- $\Omega.\lambda \leftarrow (\Omega.\lambda)(\Omega.\nu)^{-1}$ ..... (next iteration add less of the identity matrix)

else
- $\Omega.\boldsymbol{\theta}\_\mathtt{wasUpdated} \leftarrow$ **true** ..... (update 'a' reduced the cost function)
- $\Omega.\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_{\mathrm{a}}$ ..... (choose update 'a')
- $\Omega.\Delta_{\boldsymbol{\theta}} \leftarrow \Delta_{\mathrm{a}}$ ..... (store the step direction)
- $\Omega.\mathtt{cost}_{\boldsymbol{\theta}} \leftarrow \mathtt{cost}_{\boldsymbol{\theta}_{\mathrm{a}}}$ ..... (store the current cost)
- $\Omega.\lambda \leftarrow \Omega.\lambda$ ..... (keep the same damping for the next iteration)

comment: return a data structure containing all the updates

return (Ω)

Algorithm 3.3: LineSearchStep(Ω)

comment: determine a step-size that still guarantees an ellipse

$\delta \leftarrow 0.5$
repeat
- $\Delta_{\mathrm{a}} \leftarrow [\Omega.\mathbf{H}(\boldsymbol{\theta})]^{+} (\Omega.\partial_{\boldsymbol{\theta}} \mathbf{r}(\boldsymbol{\theta}))^{\mathsf{T}} \Omega.\mathbf{r}(\boldsymbol{\theta})$ ..... (compute update)
- $\boldsymbol{\theta}_{\mathrm{a}} \leftarrow \Omega.\boldsymbol{\theta} - \delta \Delta_{\mathrm{a}}$ ..... (apply update)
- $\delta \leftarrow \delta/2$ ..... (halve the step-size)
- $\mathtt{cost}_{\boldsymbol{\theta}_{\mathrm{a}}} \leftarrow \mathbf{r}(\boldsymbol{\theta}_{\mathrm{a}})^{\mathsf{T}} \mathbf{r}(\boldsymbol{\theta}_{\mathrm{a}})$ ..... (compute new residual)

until $\boldsymbol{\theta}_{\mathrm{a}}^{\mathsf{T}} \mathbf{F} \boldsymbol{\theta}_{\mathrm{a}} > 0$ **and** $(\mathtt{cost}_{\boldsymbol{\theta}_{\mathrm{a}}} < (1 - \delta\gamma)(\Omega.\mathtt{cost}_{\boldsymbol{\theta}})$ **or** $\|\Delta_{\boldsymbol{\theta}_{\mathrm{a}}}\| < \mathtt{tol}_{\Delta})$
$\Omega.\boldsymbol{\theta}\_\mathtt{wasUpdated} \leftarrow$ **true** ..... (update reduced the cost function)
$\Omega.\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_{\mathrm{a}}$ ..... (choose update)
$\Omega.\Delta_{\boldsymbol{\theta}} \leftarrow \Delta_{\mathrm{a}}$ ..... (store the current search direction)
$\Omega.\mathtt{cost}_{\boldsymbol{\theta}} \leftarrow \mathtt{cost}_{\boldsymbol{\theta}_{\mathrm{a}}}$ ..... (store the current cost)

comment: return a data structure containing all the updates

return (Ω)

of the ellipses. We characterised a point on an ellipse by its angle with respect to the canonical Cartesian coordinate system, and formed four groups, namely 0°–180°, 180°–225°, 180°–360°, and 270°–315°. For each simulation, we generated 200 ellipses and for each ellipse sampled 50 points from the angle groups 0°–180°

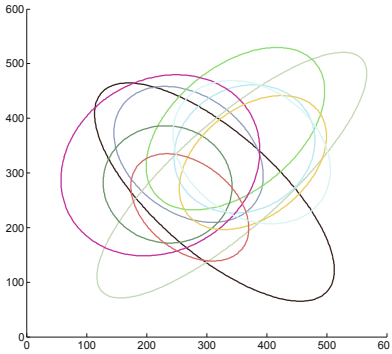
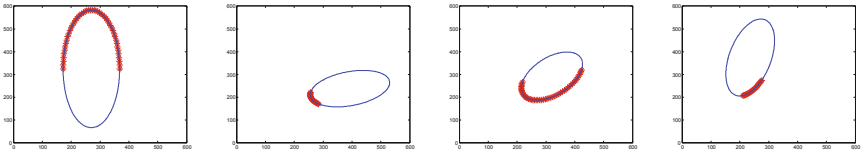


Fig. 1. An example of ten randomly generated ellipses that were sampled in simulations



(a) 0°–180° (b) 180°–225° (c) 180°–360° (d) 270°–315°

Fig. 2. An example of four different regions of an ellipse that were sampled in simulations

and 180°–360°, and 25 points from the angle groups 180°–225° and 270°–315° (see Figure 2 for an example). Every point on the ellipse segment was perturbed by identical independent homogeneous Gaussian noise at a pre-set level. For different series of experiments, different noise levels were applied.

The performance of the estimation methods was measured in terms of the *mean-root-square (RMS) orthogonal distance*

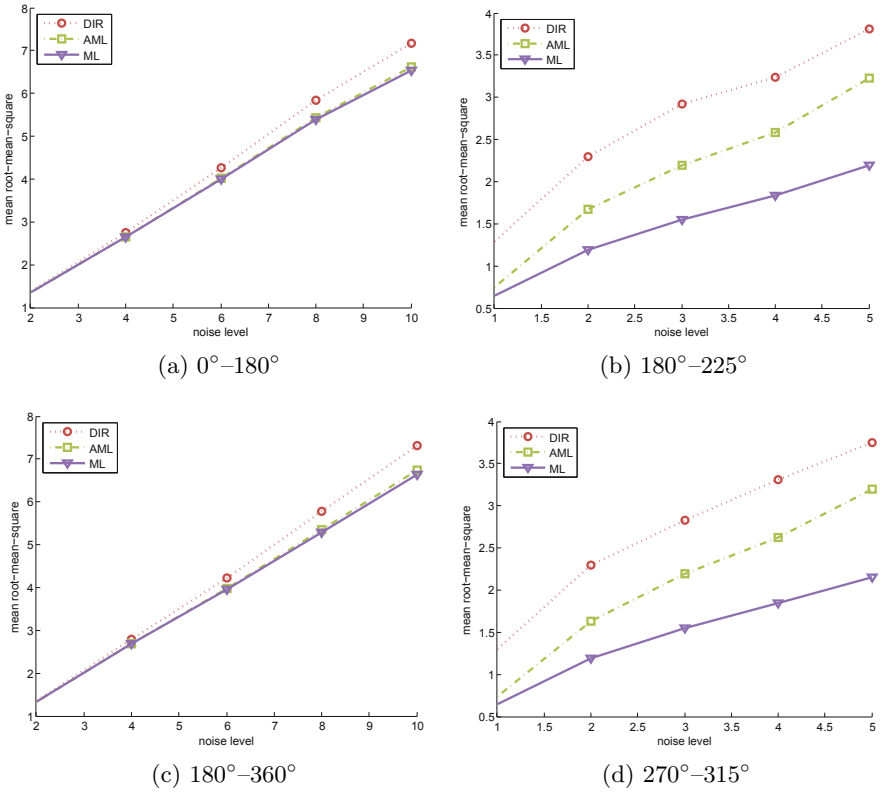
$$\sqrt{\frac{1}{2N}\sum_{n=1}^{N} d_n^2},$$

with d_n being the orthogonal distance between the nth data point and an ellipse, which measures the *geometric* error of the ellipse with respect to the data points. The process of computing the orthogonal distances d_n is rather involved—the detailed formulae can be found in [31].

The results of the four simulations are presented in Figure 3.

4.2 Real Data

It is known that the most challenging scenario for ellipse fitting is when an ellipse needs to be fit to data coming from a short segment with high eccentricity [17].



(a) 0°–180° (b) 180°–225°

(c) 180°–360° (d) 270°–315°

Fig. 3. Mean root-mean-square orthogonal distance error for varying noise levels. The results are based on 200 simulations.

Such a situation may arise in the case of fitting an ellipse to a face region, where the data lie on the chin line. The chin is a prominent and distinctive area of the face which plays an important role in biometric identification and other applications [8]. Hence, various chin-line detection methods have been reported in the literature [13,19]. To investigate how our method compares with the gold standard and baseline on real data, we selected three subjects and manually fit an ellipse to their face. Only the segment of the ellipse corresponding to the chin was used as input to the estimation methods. The results of the estimation methods are presented in Figure 4.

4.3 Stability and Efficiency

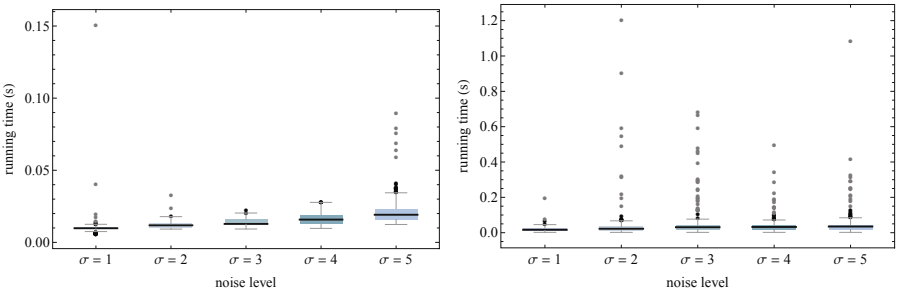
For every experiment, we verified that our algorithm was indeed producing an ellipse fit. We also studied the average running time of a MATLAB implementation of our algorithm [22] for varying noise levels. Figure 5 reports the running times associated with the results in Figures 3a and 3b.

(a) Ground Truth (b) ML (c) AML (d) Direct Fit

Fig. 4. Comparison of ellipse estimation methods on face fitting. Column (a) depicts the manually fitted ground truth ellipses. Only the portion of the ground truth ellipses corresponding to the chin of the subjects was used as input data for the estimation methods. In columns (b), (c) and (d), the ellipses have a black border on the chin indicating the input data.

5 Discussion

The experiments on synthetic data show that on average our algorithm produces results that are indeed a good approximation of the maximum likelihood method. For ill-posed problems, when the data points are sampled from a very short ellipse segment (Figures 3b and 3d), the accuracy of our approximate maximum likelihood estimate deteriorates more rapidly as noise increases. Nevertheless, it is still substantially better than the direct ellipse fit. The observed accuracy of the Sampson approximation is in agreement with the findings of Kanatani



(a) 0°–180° (see Figure 3a) (b) 180°–225° (see Figure 3b)

Fig. 5. Boxplots of the number of seconds that elapsed before our algorithm converged for (a) well-posed and (b) ill-posed problems

and Sugaya [15]. However, even though the accuracy of the Sampson distance was acknowledged by other researchers, we believe it has not found its way into mainstream use for two important reasons: (1) it was unclear how to impose the ellipse constraint in a straightforward manner, and (2) the numerical scheme used to minimize the Sampson distance would break down even with moderate noise [5, 16]. Our algorithm addresses both of these shortcomings.

The results on real data serve to further confirm the utility of our method. It was noted in [10] that the direct ellipse fit produces unreliable results when the data is coming from a short segment of an ellipse with high eccentricity. This was confirmed on the real data set used in our experimentation—the proposed method performs much better in the specific short-arc scenarios considered in this case.

Another benefit of our technique is that it is very fast and, compared to the maximum likelihood schemes [11,27,31], is easy to implement. When seeded with the direct ellipse fit estimate and applied to a well-posed problem, it converges rapidly (see Figure 5a). For ill-posed problems, the convergence of the method is typically still very fast (see Figure 5b), although occasionally more than a hundred iterations may be required to reach a solution.

6 Conclusion

We have demonstrated how the Sampson distance can be exploited to produce a fast and stable numerical scheme that guarantees generation of an elliptic fit to data. Our experiments have validated the stability and efficiency of the proposed scheme, and have affirmed the superiority of our technique over the baseline algebraic method. Experimental results also showed that our algorithm produces on average accurate approximations of the results generated with the use of the gold standard orthogonal-distance-based fitting method. The source code for the algorithm can be found at [22]. On that website we have also made

an interactive web-based application available, which can be used to explore in more detail how our guaranteed ellipse fitting method compares with the direct ellipse fit. The application gives the user control over the shape of the ellipse, the arc from which data points are sampled, and the noise level. Practitioners are now offered an ellipse-fitting technique that strikes a balance between simplicity, accuracy, and efficiency.

The mechanism for guaranteeing generation of an ellipse estimate can be extended to other ellipse fitting methods, and this will be the subject of our future research.

Acknowledgements. This work was partially supported by the Australian Research Council.

References

1. Ahn, S.J., Rauh, W., Warnecke, H.: Least-squares orthogonal distances fitting of circle, sphere, ellipse, hyperbola, and parabola. Pattern Recognition 34(12), 2283–2303 (2001)
2. Al-Sharadqah, A., Chernov, N.: A doubly optimal ellipse fit. Comput. Statist. Data Anal. 56(9), 2771–2781 (2012)
3. Bai, X., Sun, C., Zhou, F.: Splitting touching cells based on concave points and ellipse fitting. Pattern Recognition 42(11), 2434–2446 (2009)
4. Brooks, M.J., Chojnacki, W., Gawley, D., van den Hengel, A.: What value covariance information in estimating vision parameters? In: Proc. Eighth Int. Conf. Computer Vision, vol. 1, pp. 302–308 (2001)
5. Chernov, N.: On the convergence of fitting algorithms in computer vision. J. Math. Imaging and Vision 27(3), 231–239 (2007)
6. Chojnacki, W., Brooks, M.J.: Revisiting Hartley's normalized eight-point algorithm. IEEE Trans. Pattern Anal. Mach. Intell. 25(9), 1172–1177 (2003)
7. Chojnacki, W., Brooks, M.J., van den Hengel, A., Gawley, D.: On the fitting of surfaces to data with covariances. IEEE Trans. Pattern Anal. Mach. Intell. 22(11), 1294–1303 (2000)
8. Ding, L., Martinez, A.M.: Features versus context: An approach for precise and detailed detection and delineation of faces and facial features. IEEE Trans. Pattern Anal. Mach. Intell. 32(11), 2022–2038 (2010)
9. Duan, F., Wang, L., Guo, P.: RANSAC Based Ellipse Detection with Application to Catadioptric Camera Calibration. In: Wong, K.W., Mendis, B.S.U., Bouzerdoum, A. (eds.) ICONIP 2010, Part II. LNCS, vol. 6444, pp. 525–532. Springer, Heidelberg (2010)
10. Fitzgibbon, A., Pilu, M., Fisher, R.B.: Direct least square fitting of ellipses. IEEE Trans. Pattern Anal. Mach. Intell. 21(5), 476–480 (1999)
11. Gander, W., Golub, G.H., Strebel, R.: Least-squares fitting of circles and ellipses. BIT 34(4), 558–578 (1994)
12. Halíř, R., Flusser, J.: Numerically stable direct least squares fitting of ellipses. In: Proc. Sixth Int. Conf. in Central Europe on Computer Graphics and Visualization, vol. 1, pp. 125–132 (1998)
13. Hu, M., Worrall, S., Sadka, A.H., Kondoz, A.A.: A fast and efficient chin detection method for 2D scalable face model design. In: Proc. Int. Conf. Visual Information Engineering, pp. 121–124 (2003)

14. Kanatani, K.: Statistical Optimization for Geometric Computation: Theory and Practice. Elsevier, Amsterdam (1996)
15. Kanatani, K., Sugaya, Y.: Compact algorithm for strictly ML ellipse fitting. In: Proc. 19th Int. Conf. Pattern Recognition, pp. 1–4 (2008)
16. Kanatani, K., Rangarajan, P.: Hyper least squares fitting of circles and ellipses. Comput. Statist. Data Anal. 55(6), 2197–2208 (2011)
17. Kanatani, K., Sugaya, Y.: Performance evaluation of iterative geometric fitting algorithms. Comput. Statist. Data Anal. 52(2), 1208–1222 (2007)
18. Kim, I.: Orthogonal distance fitting of ellipses. Commun. Korean Math. Soc. 17(1), 121–142 (2002)
19. Lee, K., Cham, W., Chen, Q.: Chin contour estimation using modified Canny edge detector. In: Proc. 7th Int. Conf. Control, Automation, Robotics and Vision, vol. 2, pp. 770–775 (2002)
20. Mulchrone, K.F., Choudhury, K.R.: Fitting an ellipse to an arbitrary shape: implications for strain analysis. J. Structural Geology 26(1), 143–153 (2004)
21. O'Leary, P., Zsombor-Murray, P.: Direct and specific least-square fitting of hyperbolæ and ellipses. J. Electronic Imaging 13(3), 492–503 (2004)
22. `http://sites.google.com/site/szpakz/`
23. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C. Cambridge University Press, Cambridge (1995)
24. Rosin, P.L.: A note on the least squares fitting of ellipses. Pattern Recognition Lett. 14(10), 799–808 (1993)
25. Sampson, P.D.: Fitting conic sections to 'very scattered' data: An iterative refinement of the Bookstein algorithm. Computer Graphics and Image Processing 18(1), 97–108 (1982)
26. Sarzi, M., Rix, H., Shields, J.C., Rudnick, G., Ho, L.C., McIntosh, D.H., Filippenko, A.V., Sargent, W.L.W.: Supermassive black holes in bulges. The Astrophysical Journal 550(1), 65–74 (2001)
27. Sturm, P., Gargallo, P.: Conic Fitting Using the Geometric Distance. In: Yagi, Y., Kang, S.B., Kweon, I.S., Zha, H. (eds.) ACCV 2007, Part II. LNCS, vol. 4844, pp. 784–795. Springer, Heidelberg (2007)
28. Yu, D., Pham, T.D., Zhou, X.: Analysis and recognition of touching cell images based on morphological structures. Computers in Biology and Medicine 39(1), 27–39 (2009)
29. Yu, J., Kulkarni, S.R., Poor, H.V.: Robust ellipse and spheroid fitting. Pattern Recognition Lett. 33(5), 492–499 (2012)
30. Zhang, G., Jayas, D.S., White, N.D.: Separation of touching grain kernels in an image by ellipse fitting algorithm. Biosystems Engineering 92(2), 135–142 (2005)
31. Zhang, Z.: Parameter estimation techniques: a tutorial with application to conic fitting. Image and Vision Computing 15(1), 59–76 (1997)