Conic Fitting Using the Geometric Distance

Peter Sturm, Pau Gargallo

Conic Fitting Using the Geometric Distance

Peter Sturm and Pau Gargallo

INRIA Rhône-Alpes and Laboratoire Jean Kuntzmann, France

Abstract. We consider the problem of fitting a conic to a set of 2D points. It is commonly agreed that minimizing geometrical error, i.e. the sum of squared distances between the points and the conic, is better than using an algebraic error measure. However, most existing methods rely on algebraic error measures. This is usually motivated by the fact that point-to-conic distances are difficult to compute and the belief that non-linear optimization of conics is computationally very expensive. In this paper, we describe a parameterization for the conic fitting problem that allows to circumvent the difficulty of computing point-to-conic distances, and we show how to perform the non-linear optimization process efficiently.

1 Introduction

Fitting of ellipses, or conics in general, to edge or other data is a basic task in computer vision and image processing. Most existing works concentrate on solving the problem using linear least squares formulations [3, 4, 16]. Correcting the bias introduced by the linear problem formulation, is often aimed at by solving iteratively reweighted linear least squares problems [8–10, 12, 16], which is equivalent to non-linear optimization.

In this paper, we propose a non-linear optimization approach for fitting a conic to 2D points, based on minimizing the sum of squared geometric distances between the points and the conic. The arguments why most of the algorithms proposed in literature do not use the sum of squared geometrical distances as explicit cost function, are:

- non-linear optimization is required, thus the algorithms will be much slower.
- computation of a point's distance to a conic requires the solution of a 4th order polynomial [13, 18], which is time-consuming and does not allow analytical derivation (for optimization methods requiring derivatives), thus leading to the use of numerical differentiation, which is again time-consuming.

The main goal of this paper is to partly contradict these arguments. This is mainly achieved by parameterizing the problem in a way that allows to replace point-to-conic distance computations by point-to-point distance computations, thus avoiding the solution of 4th order polynomials. The problem formulation remains non-linear though. However, we show how to solve our non-linear optimization problem efficiently, in a manner routinely used in bundle adjustment.

2 Problem Formulation

2.1 Cost Function

Let $\mathbf{q}_p = (x_p, y_p, 1)^\mathsf{T}, p = 1 \dots n$ be the homogeneous coordinates of n measured 2D points. The aim is to fit a conic to these points. Many methods have been proposed for this task, often based on minimizing the sum of algebraic distances [3, 4, 12] (here, C is the usual symmetric 3×3 matrix representing the conic):

$$\sum_{p=1}^{n} \left(C_{11}x_p^2 + C_{22}y_p^2 + 2C_{12}x_p y_p + 2C_{13}x_p + 2C_{23}y_p + C_{33} \right)^2$$

This is a linear least squares problem, requiring some constraint on the unknowns in order to avoid the trivial solution. For example, Bookstein proposes the constraint $C_{11}^2 + 2C_{12}^2 + C_{22}^2 = 1$, which allows to make the solution invariant to Euclidean transformations of the data [3]. Fitzgibbon, Pilu and Fisher impose $4(C_{11}C_{22} - C_{12}^2) = 1$ in order to guarantee that the fitted conic will be an ellipse [4]. In both cases, the constrained linear least squares problem can be solved by solving a 3×3 symmetric generalized eigenvalue problem.

The cost function we want to minimize (cf. section 5), is

$$\sum_{p=1}^{n} dist\left(\mathbf{q}_p, \mathsf{C}\right)^2 \tag{1}$$

where $dist(\mathbf{q}, \mathsf{C})$ is the geometric distance between a point $\mathbf{q}$ and a conic C, i.e. the distance between $\mathbf{q}$ and the point on C, that is closest to $\mathbf{q}$. Determining $dist(\mathbf{q}, \mathsf{C})$ requires in general to compute the roots of a 4th order polynomial.

2.2 Transformations and Types of Conics

Let P be a projective transformation acting on 2D points (i.e. P is a 3×3 matrix). A conic C is transformed by P according to ($\sim$ means equality up to scale):

$$\mathsf{C}' \sim \mathsf{P}^{-\mathsf{T}} \mathsf{C} \mathsf{P}^{-1} \tag{2}$$

In this work we are only interested in real conics, i.e. that do not only contain imaginary points. These can be characterized using the eigendecomposition of the conic's 3×3 matrix: imaginary conics are exactly those whose eigenvalues have all the same sign [2]. We are thus only interested in conics with eigenvalues of different signs. This constraint will be explicitly imposed, as shown in the following section. In addition, we are only interested in *proper* conics, i.e. non-degenerate ones, with only non-zero eigenvalues. Concerning different types of real conics, we distinguish the projective and affine classes:

- all proper real conics are projectively equivalent, i.e. for any two conics, there exists at least one projective transformation relating them according to (2).
- affine classes: ellipses, hyperbolae, parabolae.

In the following, we formulate the optimization problem for general conics, i.e. the corresponding algorithm may find the correct solution even if the initial guess is of the "wrong type". Specialization of the method to the 3 affine cases of interest, is relatively straightforward; details are given in [15].

3 Minimizing the Geometrical Distance

In this section, we describe our method for minimizing the geometrical distance based cost function. The key of the method is the parameterization of the problem. In the next paragraph, we will first describe the parameterization, before showing that it indeed allows to minimize geometrical distance. After this, we explain how to initialize the parameters and describe how to solve the non-linear optimization problem in a computationally efficient way.

3.1 Parameterization

The parameterization explained in the following, is illustrated in figure 1.

For each of the n measured points $\mathbf{q}_p$, we parameterize a point $\hat{\mathbf{q}}_p$, such that all $\hat{\mathbf{q}}_p$ lie on a conic. The simplest way to do so is to choose the unit circle as support, in which case we may parameterize the $\hat{\mathbf{q}}_p$ by an angle α_p each:

$$\hat{\mathbf{q}}_p = \begin{pmatrix} \cos\alpha_p \\ \sin\alpha_p \\ 1 \end{pmatrix}$$

Furthermore, we include in our parameterization a 2D projective transformation, or, homography, P.

We then want to solve the following optimization problem:

$$\min_{\mathsf{P},\alpha_1 \cdots \alpha_n} \sum_{p=1}^{n} dist\left(\mathbf{q}_p, \mathsf{P}\hat{\mathbf{q}}_p\right)^2 \tag{3}$$

In section 3.2, we show that this parameterization indeed allows to minimize the desired cost function based on point-to-conic distances.

At first sight, this parameterization has the drawback of a much larger number of parameters than necessary: $n + 8$ (the n angles α_p and 8 parameters for P) instead of 5 that would suffice to parameterize a conic. We will show however, in section 3.4, that the optimization can nevertheless be carried out in a computationally efficient manner, due to the sparsity of the normal equations associated to our least squares cost function.

Up to now, we have considered P as a general 2D homography, which is clearly an overparameterization. We do actually parameterize P minimally:

$$\mathsf{P} \sim \mathsf{R}\,\Sigma = \mathsf{R}\,\mathrm{diag}(a, b, c)$$

where R is an orthonormal matrix and a, b and c are scalars. We show in the following section that this parameterization is sufficient, i.e. it allows to express all proper real conics.

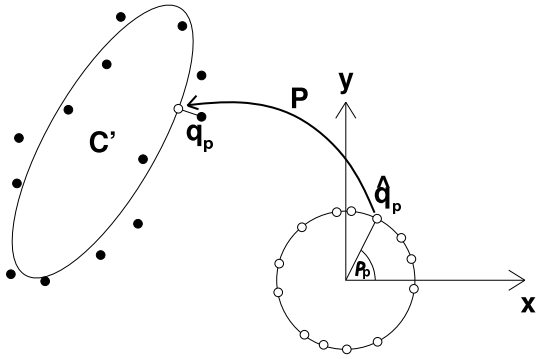

Fig. 1. Illustration of our parameterization.

We may thus parameterize P using 6 parameters (3 for R and the 3 scalars). Since the scalars are only relevant up to a global scale factor, we may fix one and thus reduce the number of parameters for P to the minimum of 5. More details on the parameterization of the orthonormal matrix R are given in section 3.4.

3.2 Completeness of the Parameterization

We first show that the above parameterization allows to "reach" all proper real conics and then, that minimizing the associated cost function (3) is equivalent to minimizing the desired cost function (1).

For any choice of R, a, b and c ($a, b, c \neq 0$), the associated homography will map the points $\hat{\mathbf{q}}_p$ to a set of points that lie on a conic C. This is obvious since point-conic incidence is invariant to projective transformations and since the $\hat{\mathbf{q}}_p$ lie on a conic at the outset (the unit circle). The resulting conic C is given by:

$$\mathsf{C} \sim \mathsf{P}^{-\mathsf{T}} \underbrace{\begin{pmatrix} 1 & & \\ & 1 & \\ & & -1 \end{pmatrix}}_{\text{unit circle}} \mathsf{P}^{-1} \sim \mathsf{R} \begin{pmatrix} 1/a^2 & & \\ & 1/b^2 & \\ & & -1/c^2 \end{pmatrix} \mathsf{R}^{\mathsf{T}}$$

We now show that any proper real conic C′ can be "reached" by our parameterization, i.e. that there exist an orthonormal matrix R and scalars a, b and c such that $\mathsf{C} \sim \mathsf{C}'$. To do so, we consider the eigendecomposition of C′:

$$\mathsf{C}' = \mathsf{R}' \begin{pmatrix} a' & & \\ & b' & \\ & & c' \end{pmatrix} \mathsf{R}'^{\mathsf{T}}$$

where R′ is an orthonormal matrix, whose rows are eigenvectors of C′, and a', b' and c' are its eigenvalues (any symmetric matrix may be decomposed in this way). The condition for C′ being a proper conic is that its three eigenvalues

are non-zero, and the condition that it is a proper **and** real conic is that one eigenvalue's sign is opposed to that of the two others.

If for example, c' is this eigenvalue, then with $\mathsf{R} = \mathsf{R}', a = 1/\sqrt{|a'|}, b = 1/\sqrt{|b'|}$ and $c = 1/\sqrt{|c'|}$, we have obviously $\mathsf{C} \sim \mathsf{C}'$. If the "individual" eigenvalue is a' instead, the following solution holds:

$$a = \frac{1}{\sqrt{|c'|}} \quad b = \frac{1}{\sqrt{|b'|}} \quad c = \frac{1}{\sqrt{|a'|}} \quad \mathsf{R} = \begin{pmatrix} & & 1 \\ & -1 & \\ 1 & & \end{pmatrix} \mathsf{R}'$$

and similarly for b' being the "individual" eigenvalue. Hence, our parameterization of an homography via an orthonormal matrix and three scalars, is complete.

We now show that the associated cost function (3), is equivalent to the desired cost function (1), i.e. that the global minima of both cost functions correspond to the same conic (if a unique global minimum exists of course). Let C' be the global minimum of the cost function (1). Let, for any measured point $\mathbf{q}_p$, $\hat{\mathbf{v}}_p$ be the closest point on C'. If more than one point on C' are equidistant from $\mathbf{q}_p$, pick any one of them.

We have shown above that there exist R, a, b and c, such that P maps the unit circle to C'. Let $\hat{\mathbf{w}}_p = \mathsf{P}^{-1}\hat{\mathbf{v}}_p$. Since $\hat{\mathbf{v}}_p$ lies on C', it follows that $\hat{\mathbf{w}}_p$ lies on the unit circle. Hence, there exists an angle α_p such that $\hat{\mathbf{w}}_p \sim (\cos\alpha_p, \sin\alpha_p, 1)^\mathsf{T}$. Consequently, there exists a set of parameters $\mathsf{R}, a, b, c, \alpha_1, \ldots, \alpha_n$ for which the value of the cost function (3) is the same as that of the global minimum of (1). Hence, our parameterization and cost function are equivalent to minimizing the desired cost function based on geometrical distance between points and conics.

3.3 Initialization

Minimizing the cost function (3) requires a non-linear, thus iterative, optimization method. Initial values for the parameters may be taken from the result of any other (linear) method. Let C' be the initial guess for the conic. The initial values for R, a, b and c (thus, for P) are obtained in the way outlined in the previous section, based on the eigendecomposition of C'.

As for the angles α_p, we determine the closest points on C' to the measured $\mathbf{q}_p$, by solving the 4th order polynomial mentioned in the introduction or an equivalent problem (see [15] for details). We then map these to the unit circle using P and extract the angles α_p, as described in the previous section.

3.4 Optimization

We now describe how we optimize the cost function (3). Any non-linear optimization method may be used, but since we deal with a non-linear least squares problem, we use the Levenberg-Marquardt method [7]. In the following, we describe how we deal with the rotational components of our parameterization (the orthonormal matrix R and the angles α_p), we then explicitly give the Jacobian of the cost function, and show how the normal equations' sparsity may be used to solve them efficiently.

Update of Rotational Parameters. To avoid singularities in the parameterization of the orthonormal matrix R, we estimate, as is typical practice e.g. in photogrammetry [1], a first order approximation of an orthonormal "update" matrix at each iteration, as follows:

1. Let R_0 be the estimation of R after the previous iteration.
2. Let $\mathsf{R}_1 = \mathsf{R}_0\Delta$ be the estimation to be obtained after the current iteration. Here, we only allow the update matrix Δ to vary, i.e. R_0 is kept fixed. Using the Euler angles α, β, γ, we may parameterize Δ as follows:

$$\Delta = \begin{pmatrix} \cos\beta\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \cos\beta\sin\gamma & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma \\ -\sin\beta & \sin\alpha\cos\beta & \cos\alpha\cos\beta \end{pmatrix} \tag{4}$$

3. The update angles α, β and γ will usually be very small, i.e. we have $\cos\alpha \approx 1$ and $\sin\alpha \approx \alpha$. Instead of optimizing directly over the angles, we thus use the first order approximation of Δ:

$$\Delta' = \begin{pmatrix} 1 & -\gamma & \beta \\ \gamma & 1 & -\alpha \\ -\beta & \alpha & 1 \end{pmatrix}$$

4. In the cost function (3), we thus replace R by $\mathsf{R}_0\Delta'$, and estimate α, β and γ. At the end of the iteration, we update the estimation of R. In order to keep R orthonormal, we do of course not update it using the first order approximation, i.e. as $\mathsf{R}_1 = \mathsf{R}_0\Delta'$. Instead, we compute an exact orthonormal update matrix Δ using equation (4) and the estimated angles, and update the rotation via $\mathsf{R}_1 = \mathsf{R}_0\Delta$.
5. It is important to note that at the next iteration, R_1 will be kept fixed on its turn, and new (small) update angles will be estimated. Thus, the initial values of the update angles at each iteration, are always zero, which greatly simplifies the analytical computation of cost function's Jacobian.

The points $\hat{\mathbf{q}}_p$ on the unit circle, are updated in a similar manner, using a 1D rotation matrix each for the update:

$$\Psi_p = \begin{pmatrix} \cos\rho_p & -\sin\rho_p & 0 \\ \sin\rho_p & \cos\rho_p & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

and its first order approximation:

$$\Psi'_p = \begin{pmatrix} 1 & -\rho_p & 0 \\ \rho_p & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

Points are thus updated as follows: $\hat{\mathbf{q}}_p \rightarrow \Psi_p\hat{\mathbf{q}}_p$ (where, as for R, the update angles are estimated using the first order approximations Ψ'_p).

Cost Function and Jacobian. Let the measured points be given by $\mathbf{q}_p = (x_p, y_p, 1)^\mathsf{T}$, and the current estimate of the $\hat{\mathbf{q}}_p$ by $\hat{\mathbf{q}}_p = (\hat{x}_p, \hat{y}_p, 1)^\mathsf{T}$. At each iteration, we have to solve the problem:

$$\min_{a,b,c,\alpha,\beta,\gamma,\rho_1 \cdots \rho_n} \sum_{p=1}^{n} d^2 \left(\mathbf{q}_p, \mathsf{R}\Delta'\Sigma\Psi'_p \hat{\mathbf{q}}_p \right) \tag{5}$$

This has a least squares form, i.e. we may formulate the cost function using $2n$ residual functions:

$$\sum_{j=1}^{2n} r_j^2 \quad \text{with} \quad r_{2i-1} = x_i - \frac{(\mathsf{R}\Delta'\Sigma\Psi'_i\hat{\mathbf{q}}_i)_1}{(\mathsf{R}\Delta'\Sigma\Psi'_i\hat{\mathbf{q}}_i)_3} \quad \text{and} \quad r_{2i} = y_i - \frac{(\mathsf{R}\Delta'\Sigma\Psi'_i\hat{\mathbf{q}}_i)_2}{(\mathsf{R}\Delta'\Sigma\Psi'_i\hat{\mathbf{q}}_i)_3}$$

As for the Jacobian of the cost function, it is defined as:

$$\mathsf{J} = \begin{pmatrix} \frac{\partial r_1}{\partial a} & \frac{\partial r_1}{\partial b} & \frac{\partial r_1}{\partial c} & \frac{\partial r_1}{\partial \alpha} & \frac{\partial r_1}{\partial \beta} & \frac{\partial r_1}{\partial \gamma} & \frac{\partial r_1}{\partial \rho_1} & \frac{\partial r_1}{\partial \rho_2} & \cdots & \frac{\partial r_1}{\partial \rho_n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial r_{2n}}{\partial a} & \frac{\partial r_{2n}}{\partial b} & \frac{\partial r_{2n}}{\partial c} & \frac{\partial r_{2n}}{\partial \alpha} & \frac{\partial r_{2n}}{\partial \beta} & \frac{\partial r_{2n}}{\partial \gamma} & \frac{\partial r_{2n}}{\partial \rho_1} & \frac{\partial r_{2n}}{\partial \rho_2} & \cdots & \frac{\partial r_{2n}}{\partial \rho_n} \end{pmatrix}$$

It can be computed analytically, as follows. Due to the fact that before each iteration, the update angles $\alpha, \beta, \gamma, \rho_1, \ldots, \rho_n$ are all zero, the entries of the Jacobian, evaluated at each iteration, have the following very simple form:

$$\begin{pmatrix} \hat{x}_1 u_{11} & \hat{y}_1 u_{12} & u_{13} & (b\hat{y}_1 u_{13} - c u_{12}) & (c u_{11} - a\hat{x}_1 u_{13}) & (a\hat{x}_1 u_{12} - b\hat{y}_1 u_{11}) & u_{14} & \cdots & 0 \\ \hat{x}_1 v_{11} & \hat{y}_1 v_{12} & v_{13} & (b\hat{y}_1 v_{13} - c v_{12}) & (c v_{11} - a\hat{x}_1 v_{13}) & (a\hat{x}_1 v_{12} - b\hat{y}_1 v_{11}) & v_{14} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \hat{x}_n u_{n1} & \hat{y}_n u_{n2} & u_{n3} & (b\hat{y}_n u_{n3} - c u_{n2}) & (c u_{n1} - a\hat{x}_n u_{n3}) & (a\hat{x}_n u_{n2} - b\hat{y}_n u_{n1}) & 0 & \cdots & u_{n4} \\ \hat{x}_n v_{n1} & \hat{y}_n v_{n2} & v_{n3} & (b\hat{y}_n v_{n3} - c v_{n2}) & (c v_{n1} - a\hat{x}_n v_{n3}) & (a\hat{x}_n v_{n2} - b\hat{y}_n v_{n1}) & 0 & \cdots & v_{n4} \end{pmatrix}$$

with

$$\mathbf{u}_i = s_i^2 \begin{pmatrix} bR_{23}\hat{y}_i - cR_{22} \\ cR_{21} - aR_{23}\hat{x}_i \\ aR_{22}\hat{x}_i - bR_{21}\hat{y}_i \\ c(bR_{21}\hat{x}_i + aR_{22}\hat{y}_i) - abR_{23} \end{pmatrix} \quad \mathbf{v}_i = s_i^2 \begin{pmatrix} cR_{12} - bR_{13}\hat{y}_i \\ aR_{13}\hat{x}_i - cR_{11} \\ bR_{11}\hat{y}_i - aR_{12}\hat{x}_i \\ abR_{13} - c(bR_{11}\hat{x}_i + aR_{12}\hat{y}_i) \end{pmatrix}$$

and $s_i = (aR_{31}\hat{x}_i + bR_{32}\hat{y}_i + cR_{33})^{-1}$.

As for the residual functions themselves, with $\alpha, \beta, \gamma, \rho_1, \ldots, \rho_n$ being zero before each iteration, they evaluate to:

$$\begin{aligned} r_{2i-1} &= x_i - s_i \left(aR_{11}\hat{x}_i + bR_{12}\hat{y}_i + cR_{13} \right) \\ r_{2i} &= y_i - s_i \left(aR_{21}\hat{x}_i + bR_{22}\hat{y}_i + cR_{23} \right) \end{aligned}$$

With the explicit expressions for the Jacobian and the residual functions, we have given all ingredients required to optimize the cost function using e.g. the Levenberg-Marquardt or Gauss-Newton methods. In the following paragraph, we show how to benefit from the sparsity of the Jacobian (nearly all derivatives with respect to the ρ_p are zero).

Hessian. The basic approximation to the Hessian matrix used in least squares optimizers such as Gauss-Newton is $\mathsf{H} = \mathsf{J}^\mathsf{T}\,\mathsf{J}$. Each iteration of such a non-linear method comes down to solving a linear equation system of the following form:

$$\begin{pmatrix} \mathsf{A}_{6\times 6} & \mathsf{B}_{6\times n} \\ \mathsf{B}^\mathsf{T} & \mathsf{D}_{n\times n} \end{pmatrix} \begin{pmatrix} \mathbf{x}_6 \\ \mathbf{y}_n \end{pmatrix} = \begin{pmatrix} \mathbf{a}_6 \\ \mathbf{b}_n \end{pmatrix}$$

where D is, due to sparsity of the Jacobian (see previous paragraph) a diagonal matrix. The right-hand side is usually the negative gradient of the cost function, which for least squares problems can also be computed as $-\mathsf{J}^\mathsf{T}\mathbf{r}$, $\mathbf{r}$ being the vector of the $2n$ residuals defined above. As suggested in [14], we may reduce this $(6+n)\times(6+n)$ problem to a 6×6 problem, as follows:

1. The lower set of equations give:

$$\mathbf{y} = \mathsf{D}^{-1}\left(\mathbf{b} - \mathsf{B}^\mathsf{T}\mathbf{x}\right) \tag{6}$$

2. Replacing this in the upper set of equations, we get:

$$\left(\mathsf{A} - \mathsf{B}\mathsf{D}^{-1}\mathsf{B}^\mathsf{T}\right)\mathbf{x} = \mathbf{a} - \mathsf{B}\mathsf{D}^{-1}\mathbf{b} \tag{7}$$

3. Since D is diagonal, its inversion is trivial, and thus the coefficients of the equation system (7) may be computed efficiently (in time and memory).
4. Once $\mathbf{x}$ is computed by solving the 6×6 system (7), $\mathbf{y}$ is obtained using (6).

Hence, the most complex *individual* operation at each iteration is the same as that in iterative methods minimzing algebraic distance – inverting a 6×6 symmetric matrix or, equivalently, solving a linear equation system of the same size. In practice, we reduce the original problem to $(5+n)\times(5+n)$, respectively 5×5, by fixing one of the scalars a, b, c (the one with the largest absolute value). However, most of the computation time is actually spent on computing the partial derivatives required to compute the coefficients of the above equation systems. Overall, the computational complexity is linear in the number of data points. A detailed complexity analysis is given in [15].

With a non-optimized implementation, measured computation times for one iteration were about 10 times those required for the standard linear method (`Linear` in the next section). This may seem much but note that e.g. with 200 data points, one iteration requires less than 2 milli-secs on a 2.8GHz Pentium 4.

4 Experimental Results

Points were simulated on a unit circle, equally distributed over an arc of varying length, to simulate occluded data. Each point was subjected to Gaussian noise (in x and y). Six methods were used to fit conics:

- `Linear`: least-square solution based on the algebraic distance, using the constraint of unit norm on the conic's coefficients.
- `Bookstein`: the method of [3].

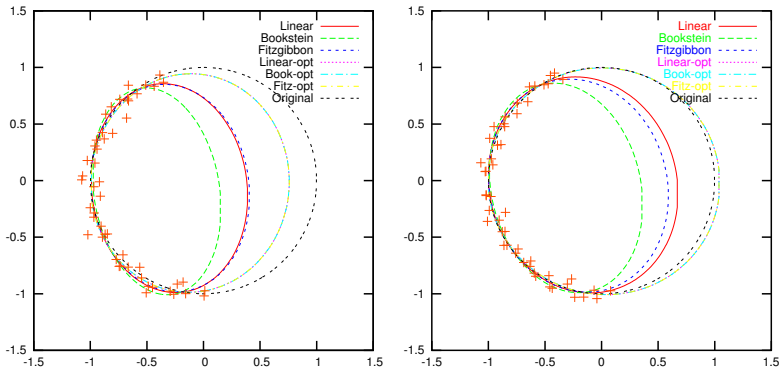

Fig. 2. Two examples: 50 points were distributed over an arc of 160°, and were subjected to a Gaussian noise of a standard deviation of 5 percent the radius of the circle.

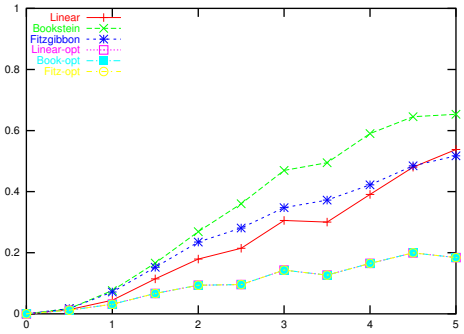

Fig. 3. Relative error on estimated minor axis length, as a function of noise (the unit of the y-axis is 100%). The graphs for the three non-linear optimization methods are superimposed.

- `Fitzgibbon`: the method of [4].
- Non-linear optimization using our method, using the results of the above methods as initialization: `Linear-opt`, `Book-opt` and `Fitz-opt`.

We performed experiments for many combinations of noise level, amount of occlusion and number of points on the conic, see [15] for a detailled account. Figure 2 shows two typical examples. With all three initializations, the optimization method converged to the same conic in a few iterations each (2 or 3 typically).

It is not obvious how to quantitatively compare the methods. Displaying residual geometrical point-to-conic distances for example would be unfair, since our method is designed for minimizing this. Instead, we compute an error measure on the estimated conic. Figure 3 shows the relative error on the length of the estimated conic's minor axis (one indicator of how well the conic's shape has been recovered), relative to the amount of noise. Each point in the graph represents the median value of the results of 50 simulation runs. All methods

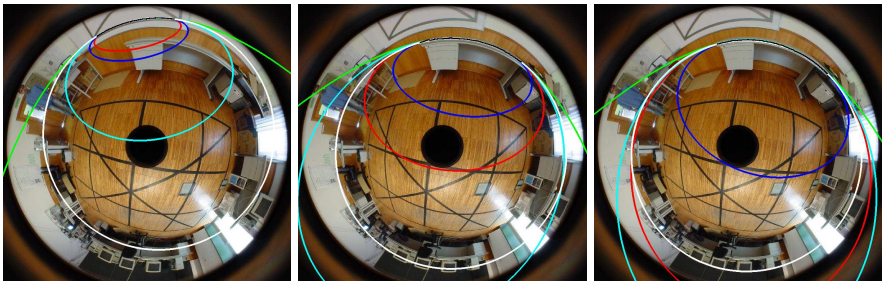

Fig. 4. Sample results on real data: fitting conics to catadioptric line images. Colors are as in figure 2; reference conics are shown in white and data points in black, in the common portion of the estimated conics.

degrade rather gracefully, the non-linear optimization results being by far the best (the three graphs are superimposed). We also tested our approach with the results of a hyperbola-specific version of [4] as initialization. In most cases, the optimization method is capable of switching from an hyperbola to an ellipse, and to reach the same solution as when initialized with an ellipse.

Figure 4 shows sample results on real data, fitting conics to edge points of catadioptric line images (same color code as in figure 2). Reference conics are shown in white; they were fitted using calibration information on the catadioptric camera (restricting the problem to 2 dof) and serve here as "ground truth". The data points are shown by the black portion common to all estimated conics. They cover very small portions of the conics, making the fitting rather ill-posed. The ill-posedness shows e.g. in the fact that in most cases, conics with widely varying shape have similar residuals. Nevertheless, our approach gives results that are clearly more consistent than for any of the other methods; also note that in the shown examples, the three non-linear optimizations converged to the same conic each time. More results are given in [15].

5 Discussion on Choice of Cost Function

Let us briefly discuss the cost function used. A usual choice, and the one we adopted here, is the sum of squared geometrical distances of the points to the conic. Minimizing this cost function gives the optimal conic in the maximum likelihood sense, under the assumption that data points are generated from points on the true conic, by displacing them along the normal direction by a random distance that follows a zero mean Gaussian distribution, the same for all points. Another choice [17] is based on the assumption that a data point could be generated from *any* point on the true conic, by displacing it possibly in other directions than the normal to the conic. There may be other possibilities, taking into account the different densities of data points along the conic in areas with different curvatures. Which cost function to choose depends on the underlying application but of course also on the complexity of implementation and computation.

In this work we use the cost function based on the geometrical distance between data points and the conic; it is analytically and computationally more tractable than e.g. [17]. Further, if data points are obtained by edge detection, i.e. if they form a contour, then it is reasonable to assume that the order of the data points along the contour is the same as that of the points on the true conic that were generating them. Hence, it may not be necessary here to evaluate the probability of all points on the conic generating all data points and it seems reasonable to stick with the geometric distance between data points and the conic, i.e. the distance between data points and the closest points on the conic. A more detailed discussion is beyond the scope of this paper though.

A final comment is that it is straightforward to embed our approach in any M-estimator, in order to make it robust to outliers.

6 Conclusions and Perspectives

We have proposed a method for fitting conics to points, minimizing geometrical distance. The method avoids the solution of 4th order polynomials, often considered to be one of the main reasons for using algebraic distances. We have described in as much detail as possible how to perform the non-linear optimization computationally efficiently. A few simulation results are presented that suggest that the optimization of geometrical distance may correct bias present in results of linear methods, as expected. However, the main motivation for this paper was not to measure absolute performance, but to show that conic fitting by minimization of geometrical distance, is feasible.

Recently, we became aware of the work [5], that describes an ellipse-specific method very similar in spirit and formulation to ours. Our method, as presented, is not specific to any affine conic type. This is an advantage if the type of conic is not known beforehand (e.g. line-based camera calibration of omnidirectional cameras is based on fitting conics of possibly different types [6]), and switching between different types is indeed completely natural for the method. However, we have also implemented ellipse-, hyperbola- and parabola-specific versions of the method [15].

The proposed approach for conic fitting can be adapted to other problems. This is rather direct for e.g. the reconstruction of a conic's shape from multiple calibrated images or the optimization of the pose of a conic with known shape, from a single or multiple images. Equally straightforward is the extension to the fitting of quadrics to 3D point sets. Generally, the approach may be used for fitting various types of surfaces or curves to sets of points or other primitives.

Another application is plumb-line calibration, where points would have to be parameterized on lines instead of the unit circle. Besides this, we are currently investigating an extension of our approach to the estimation of the shape and/or pose of quadrics, from silhouettes in multiple images. The added difficulty is that points on quadrics have to be parameterized such as to lie on occluding contours. This may be useful for estimating articulated motions of objects modelled by quadric-shaped parts, similar to [11] which considered cone-shaped parts.

Other current work is to make a MATLAB implementation of the proposed approach publicly available, on the first author's website and to study cases when the Gauss-Newton approximation of the Hessian may become singular.

Acknowledgements. We thank Pascal Vasseur for the catadioptric image and the associated calibration data and the reviewers for very useful comments.

References

1. Atkinson, K.B. (Editor): Close Range Photogrammetry and Machine Vision. Whittles Publishing (1996)
2. Boehm, W., Prautzsch, H.: Geometric Concepts for Geometric Design. A.K. Peters (1994)
3. Bookstein, F.L.: Fitting Conic Sections to Scattered Data. Computer Graphics and Image Processing **9** (1979) 56–71
4. Fitzgibbon, A., Pilu, M., Fisher, R.B.: Direct Least Square Fitting of Ellipses. IEEE–PAMI **21 (5)** (1999) 476–480
5. Gander, W., Golub, G.H., Strebel, R.: Fitting of Circles and Ellipses. BIT **34** (1994) 556–577
6. Geyer, C., Daniilidis, K.: Catadioptric Camera Calibration. ICCV (1999) 398–404
7. Gill, P.E., Murray, W., Wright, M.H.: Practical Optimization. Academic Press (1981)
8. Halír, R.: Robust Bias-Corrected Least Squares Fitting of Ellipses. Conf. in Central Europe on Computer Graphics, Visualization and Interactive Digital Media (2000)
9. Kanatani, K.: Statistical Bias of Conic Fitting and Renormalization. IEEE–PAMI **16 (3)** (1994) 320–326
10. Kanazawa, Y., Kanatani, K.: Optimal Conic Fitting and Reliability Evaluation. IEICE Transactions on Information and Systems **E79-D (9)** (1996) 1323–1328
11. Knossow, D., Ronfard, R., Horaud, R., Devernay, F.: Tracking with the Kinematics of Extremal Contours. ACCV (2006) 664–673
12. Rosin, P.L.: Analysing Error of Fit Functions for Ellipses. Pattern Recognition Letters **17** (1996) 1461–1470
13. Rosin, P.L.: Ellipse Fitting Using Orthogonal Hyperbolae and Stirling's Oval. Graphical Models and Image Processing **60 (3)** (1998) 209–213
14. Slama C.C. (Editor): Manual of Photogrammetry, Fourth Edition. American Society of Photogrammetry and Remote Sensing (1980)
15. Sturm, P.: Conic Fitting Using the Geometric Distance. Rapport de Recherche, INRIA (2007)
16. Taubin, G.: Estimation of Planar Curves, Surfaces, and Nonplanar Space Curves Defined by Implicit Equations with Applications to Edge and Range Image Segmentation. IEEE–PAMI, **13 (11)** (1991) 1115–1138
17. Werman, M., Keren, D.: A Bayesian Method for Fitting Parametric and Nonparametric Models to Noisy Data. IEEE–PAMI, **23 (5)** (2001) 528–534
18. Zhang, Z.: Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting. Rapport de Recherche No. 2676, INRIA (1995)