Chapter 1

LEAST SQUARES FITTING OF QUADRATIC CURVES AND SURFACES

N. Chernov and H. Ma[*]
University of Alabama at Birmingham
Birmingham, AL 35294, USA

Key Words: Least squares, orthogonal regression, fitting ellipses, conics, quadrics.

AMS Subject Classification: 62J.

Abstract

In computer vision one often fits ellipses and other conics to observed points on a plane or ellipsoids/quadrics to spacial point clouds. The most accurate and robust fit is obtained by minimizing geometric (orthogonal) distances, but this problem has no closed form solution and most known algorithms are prohibitively slow. We revisit this issue based on recent advances by S. J. Ahn, D. Eberly, and our own. Ahn has sorted out various approaches and identified the most efficient one. Eberly has developed a fast method of projecting points onto ellipses/ellipsoids (and gave a proof of its convergence). We extend Eberly's projection algorithm to other conics, as well as quadrics in space. We also demonstrate that Eberly's projection method combined with Ahn's most efficient approach (and using Taubin's algebraic fit for initialization) makes a highly efficient fitting scheme working well for all quadratic curves and surfaces.

[*]E-mail address: chernov@math.uab.edu; hma@uab.edu

1. Introduction

Fitting simple contours (primitives) to observed image data is one of the basic tasks in pattern recognition and computer vision. The most popular contours are lines, circles, and ellipses (recall that round objects appear as elliptic ovals on photos). In 3D space, one often fits planes, spheres, or more complex surfaces (such as ellipsoids) to point clouds. We review the most advanced fitting methods and extend them to all quadratic curves and surfaces.

We begin with the 2D fitting problem. Let $(x_1, y_1), \dots, (x_n, y_n)$ denote the observed points. Let $P(x, y; \Theta) = 0$ be the equation of the fitting contour, where Θ represents the vector of unknown parameters. For example, lines can be defined by equation

$$x \cos\theta + y \sin\theta + d = 0, \tag{1}$$

so $\Theta = (\theta, d)$. Circles can be described by

$$(x-a)^2 + (y-b)^2 - R^2 = 0, \tag{2}$$

so $\Theta = (a, b, R)$. Ellipses can be defined by

$$\frac{\breve{x}^2}{a^2} + \frac{\breve{y}^2}{b^2} - 1 = 0 \tag{3}$$

in the canonical coordinates $\breve{x}, \breve{y}$, which can be related to x, y by translation and rotation, i.e., $\breve{x} = (x - c_1)\cos\theta - (y - c_2)\sin\theta$ and $\breve{y} = (x - c_1)\sin\theta + (y - c_2)\cos\theta$; now $\Theta = (a, b, c_1, c_2, \theta)$, where (c_1, c_2) is the center, a, b are the semiaxes, and θ is the angle of tilt.

The classical least squares fit minimizes geometric distances from the observed points to the fitting curve:

$$\mathcal{F}(\Theta) = \sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} (x_i - x_i')^2 + (y_i - y_i')^2 \quad \to \quad \min. \tag{4}$$

Here d_i denotes the geometric distance from the observed point (x_i, y_i) to the fitting contour, and (x_i', y_i') the (orthogonal) projection of (x_i, y_i) onto the contour.

The least squares fit (4) has many nice features. It is invariant under translations, rotations, and scaling, i.e., the fitting contour does not depend on the choice of the coordinate system. It provides the maximum likelihood estimate of Θ under standard statistical assumptions (where points are observed with an independent isotropic gaussian noise [9, 12, 15]). The minimization (4) is always regarded as the most desirable solution[1] of the fitting problem, albeit hard to compute in most cases.

Fig. 1 shows a sample of eight points (their coordinates are given in Table 1; they are borrowed from [19]) and the best fitting ellipse obtained by (4). We will explore this example at the end of Section 2.

[1] In particular, (4) has been prescribed by a recently ratified standard for testing the data processing software for coordinate metrology [20].

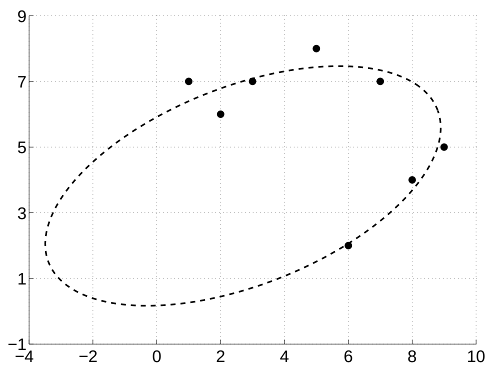

Figure 1. A sample of eight points and the best fitting ellipse.

When one fits lines (1), the problem (4) has a closed form solution, and its properties have been studied deeply [10, 15]. When one fits circles (2), the problem (4) has no closed form solution, but one has an explicit formula for the distances,

$$d_i = \sqrt{(x_i - a)^2 + (y_i - b)^2} - R, \tag{5}$$

hence one can easily compute the objective function (4), as well as its derivatives with respect to a, b, R. This makes the minimization of $\mathcal{F}$ rather straightforward – the standard Levenberg-Marquardt algorithm (the most popular and reputable scheme for solving least-squares problems [15, 19]) works well. Algebraic circle fits (say, the one by Taubin; see Appendix) provide good initial guesses.

When one fits ellipses, no simple explicit formulas are available for the distances d_i's or the projections (x_i', y_i'). Theoretically, the projection (x_i', y_i') can be found by solving a polynomial equation of degree four [28], but such a solution is complicated and inconvenient (it involves complex numbers), and even worse – it is numerically unstable [4]. Besides it is not clear how to differentiate d_i with respect to Θ. Since one cannot easily compute the objective function (4) or its derivatives, it appears that the Levenberg-Marquardt scheme is impractical (see [25] that mentioned such a point of view).

The first thorough investigation of the ellipse fitting problem was done in the middle 1990's by Gander, Golub, and Strebel [19]. They developed a roundabout way of minimizing $\mathcal{F}$ by using auxiliary parameters ω_i, $i = 1, \dots, n$, describing the location of the projected points (x_i', y_i') on the ellipse; the latter, in the canonical coordinates (3) were expressed by $\breve{x}_i' = a\cos\omega_i$ and $\breve{y}_i' = b\sin\omega_i$. Thus the objective function becomes

$$\begin{aligned}\mathcal{F} = \sum_{i=1}^{n} &(x_i - c_1 - a\cos\omega_i\cos\theta - b\sin\omega_i\sin\theta)^2 \\ &+ (y_i - c_2 + a\cos\omega_i\sin\theta - b\sin\omega_i\cos\theta)^2.\end{aligned}$$

In [19], $\mathcal{F}$ was minimized with respect to $a, b, c_1, c_2, \theta, \omega_1, \dots, \omega_n$ *simultaneously*. This procedure avoids the calculation of d_i's, but the minimization in the $(n+5)$-dimensional parameter space is predictably cumbersome and slow. So the authors of [19] demonstrated that the minimization of geometric distances for ellipses was a prohibitively difficult task.

Recently Sturm and Gargallo [25] modified the above method in several ways. In particular, they used a projective matrix that allowed them to describe conics of all types (ellipses, hyperbolas, parabolas) with the same set of 5 parameters. Thus their method could freely switch between types during iterations. But they still work in an $(n+5)$-dimensional parameter space, in that sense their method is similar to that of [19].

In the late 1990s, in computer vision applications various alternative fitting schemes were developed, where so called algebraic distances were minimized; we review them in Appendix. They produce ellipses that fit less accurately than those minimizing geometric distances (4). Some authors say that "the performance gap between algebraic fitting and geometric fitting is wide..." (see [7, p. 12]).

In the early 2000's another approach to the minimization of geometric distances (4) emerged, due to Ahn et. al. [4, 6, 7], that turned out to be very efficient. We describe it in a general form.

2. Fitting implicit curves and surfaces

Least squares problems are commonly solved by the Gauss-Newton (GN) method or its Levenberg-Marquardt (LM) correction. If one minimizes a sum of squares $\mathcal{F}(\Theta) = \sum f_i^2$, then both GM and LM would use the values of f_i's and their first derivatives with respect to Θ, which we denote by $(f_i)_\Theta$.

Now suppose, as before, that we fit a curve defined by implicit equation $P(x,y;\Theta) = 0$ to observed points (x_i, y_i). Our goal is to minimize the sum of squares (4). The GN and LM methods can be applied here in two ways: we either treat $\mathcal{F}$ as a sum of n squares, $\mathcal{F} = \sum d_i^2$, or a sum of $2n$ squares, $\mathcal{F} = \sum g_i^2 + \sum h_i^2$, where $g_i = x_i - x_i'$ and $h_i = y_i - y_i'$. In the former case we will need d_i's and their derivatives $(d_i)_\Theta$. In the latter we need f_i's and g_i's, as well as their derivatives $(g_i)_\Theta$ and $(f_i)_\Theta$. The resulting algorithm is said to be *distance-based* if one uses d_i's, or *coordinate-based* if one uses g_i's and h_i's; see Ahn et al. [6, 7].

Obviously, it is enough to know the projections (x_i', y_i') in order to compute d_i's, g_i's, and h_i's. But their derivatives $(d_i)_\Theta, (g_i)_\Theta, (f_i)_\Theta$ present a more challenging problem. Sometimes finite differences are used to approximate these derivatives, which reduces the accuracy of the fit. But there are surprisingly simple formulas for these derivatives:

Proposition. *Let (x,y) be a given a point and (x',y') denote its projection onto the curve $P(x,y;\Theta) = 0$ (then x', y' depend on Θ). Denote $g = x - x'$, $h = y - y'$, and $d^2 = g^2 + h^2$. Then we have*

$$g_\Theta = \frac{P_\Theta P_x^2 - gP_y(P_xP_{y\Theta} - P_yP_{x\Theta})}{P_x(P_x^2 + P_y^2)}, \tag{6}$$

$$h_\Theta = \frac{P_\Theta P_y^2 + hP_x(P_xP_{y\Theta} - P_yP_{x\Theta})}{P_y(P_x^2 + P_y^2)}, \tag{7}$$

and

$$d_\Theta = \frac{P_\Theta}{\sqrt{P_x^2 + P_y^2}}, \tag{8}$$

where P_Θ, P_x, P_y denote the first order partial derivatives of P with respect to Θ, x, y, respectively, and $P_{x\Theta}$ and $P_{y\Theta}$ the corresponding second order partial derivatives; all the derivatives are taken at the projection point (x', y').

Proof. Since the vector $(x - x', y - y')$ is orthogonal to the curve,

$$g = x - x' = tP_x \qquad \text{and} \qquad h = y - y' = tP_y \tag{9}$$

for some scalar t. This immediately gives

$$d^2 = g^2 + h^2 = t^2(P_x^2 + P_y^2). \tag{10}$$

Next we use differentiation with respect to Θ. Differentiating the identity $P(x', y'; \Theta) = 0$ gives

$$P_\Theta = -P_x x'_\Theta - P_y y'_\Theta = (gg_\Theta + hh_\Theta)/t, \tag{11}$$

and differentiating the identity $d^2 = g^2 + h^2$ gives

$$dd_\Theta = gg_\Theta + hh_\Theta = tP_\Theta. \tag{12}$$

Now (8) follows from (12) and (10). Differentiation of (9) gives

$$g_\Theta = t_\Theta P_x + tP_{x\Theta}, \qquad h_\Theta = t_\Theta P_y + tP_{y\Theta}.$$

Eliminating t_Θ from these two equations yields

$$g_\Theta P_y - h_\Theta P_x = -t(P_x P_{y\Theta} - P_y P_{x\Theta}). \tag{13}$$

Solving (12) and (13) for g_Θ and h_Θ we obtain (6) and (7). $\square$

The formulas (6)–(8) were essentially obtained by Ahn et al. [6, 7]. Independently the formula (8) was derived in [3] (see eq. (24) there).

Practically, the calculation of the derivatives $d_\Theta, g_\Theta, h_\Theta$ by (6)–(8) is easy once the projection point (x', y') is located. The differentiation of $P(x, y; \Theta)$ with respect to Θ is usually straightforward; for example, one can easily find the derivatives of (3) with respect to a, b, c_1, c_2, θ.

Alternatively, one can try to change the parametrization scheme in order to simplify differentiation. For example, instead of (3) one can define an ellipse by equation

$$\sqrt{(x - p_1)^2 + (y - p_2)^2} + \sqrt{(x - q_1)^2 + (y - q_2)^2} - 2a = 0 \tag{14}$$

where (p_1, p_2) and (q_1, q_2) denote its foci and $2a$, as before, the major axis. These are sometimes called *Kepler's parameters* of an ellipse; they have certain advantages [11]. In particular, differentiation of (14) with respect to p_1, p_2, q_1, q_2, and a is quite straightforward.

We note that the coordinate-based scheme using g_i's and h_i's operates with $2n$ terms and involves the second order derivatives $P_{x\Theta}$ and $P_{y\Theta}$. The distance-based scheme operates only with n terms and does not involve the second order derivatives; cf. (8). As a result, the coordinate-based scheme seems to be less efficient, and we will only use the distance-based fit in what follows.

x	1	2	5	7	9	3	6	8
y	7	6	8	7	5	7	2	4

Table 1. A benchmark example with eight points [19].

Remark. Since the distance d given must be differentiable, we have to treat it as a *signed* distance – it must be positive on one side of the curve and negative on the other, just as we had it in (5). For ellipses, one can make $d > 0$ for points outside the ellipse and $d < 0$ for points inside.

The above minimization scheme also works for surfaces in the 3D space, when they are defined by implicit equations $P(x, y, z; \Theta) = 0$. In that case the above formulas acquire an extra term corresponding to the z variable, otherwise they remain pretty much the same.

Now the minimization problem (4) can be solved by the Gauss-Newton or Levenberg-Marquardt algorithm provided one can project any given point (x, y) onto a given curve/surface $P = 0$. We will discuss the projection subproblem later. The rate of converges of the GN and LM algorithms is nearly quadratic provided one has a good initial guess (which can be found, for example, by a non-iterative algebraic fit, such as the Taubin fit, see Appendix).

Ahn et al. have applied the above minimization scheme to quadratic curves (ellipses, hyperbolas, and parabolas) and some quadratic surfaces (spheres, ellipsoids, cones); see [4, 6, 7]. They compared it with several other minimization algorithms [5, 6, 7] and concluded that this one is the fastest and most robust. We have also tested it on quadratic curves and surfaces of various types and found that it has the following two advantages over other schemes: (i) it converges in fewer iterations, and (ii) it finds a smaller value of the objective function $\mathcal{F}$ more frequently than other methods do (i.e., the other methods tend to end up in a local minimum or diverge more often that this method does).

A benchmark example introduced in [19] and used later in [4] and other papers is a simple eight point set whose coordinates are shown in Table 1. The best fitting ellipse is known to have center $(2.6996, 3.8160)$, axes $a = 6.5187$ and $b = 3.0319$, and angle of tilt $\theta = 0.3596$; see Fig. 1. We have run two fitting algorithms – the implicit fit described here and the Gander-Golub-Strebel (GGS) method, which simultaneously optimizes $n + 5$ parameters, as mentioned earlier. Both methods were initialized by randomly generated ellipses (to get an initial ellipse, we just picked 5 points randomly in the square $0 \leq x, y \leq 10$ and used the ellipse interpolating those 5 points). After running these fitting methods from 10^5 random initial guesses, we found that the implicit fit failed to converged to the best ellipse in 11% of the cases, while the GGS method failed in 26% of the cases. In those cases where both algorithms converged, the implicit method took 20 iterations, on the average, while the GGS method took 60 iterations, on the average. The cost of one iteration was also higher for the GGS method. Table 2 summarizes the results.

We note that a high number of iterations here is not unusual. The authors of [19] used a modification of the best fitting circle to initialize their GGS procedure, and it took 71 iterations (!) to converge. A coordinate-based variant of the implicit fit used in [4] took 19 iterations to converge (it was also initialized with the best fitting circle). Our distance-based

	Failure rate	Avg. iter.	Cost per iter. (flops)
Implicit	11%	20	1640
GGS	26%	60	1710

Table 2. Comparison of two ellipse fitting methods.

	Initial ellipse		
	Best Circle	"Direct fit"	Taubin fit
Implicit (G)	16	17	17
Implicit (K)	14	16	16
GGS	50	54	54

Table 3. Comparison of two ellipse fitting methods.

implicit fit converged in 16 iterations.

In our experiment we used standard geometric parameters of the ellipse, i.e., c_1, c_2, a, b, θ. With Kepler's parameters (14), things get a little faster – the implicit method converged in 14 iterations. Table 3 gives the number of iterations taken by our fitting methods (the implicit method was implemented in geometric parameters (G) and in Kepler parameters (K)), initialized with the modified best fitting circle as in [19], the "direct ellipse fit" [18], and the Taubin fit (see Appendix).

3. Projection onto conics

The fitting scheme described above will work well only if one uses an efficient and reliable solution to the projection subproblem. The latter remains the time consuming part of the fitting process [6, 7].

In practice, various heuristic projection algorithms are employed [4, 5, 6, 7, 28] that are relatively fast, but their convergence is not guaranteed (and occasionally they do fail). On the other hand, certain theoretically reliable methods were proposed [2, 27], but most of them are overly complicated and too slow for practical use. For example, the projection of a point (u, v) onto an ellipse can be found as a root of a polynomial of degree four, but, as we said, this method is quite impractical and virtually never used.

A remarkable approach to projecting points onto ellipses was found by D. Eberly in 2004 [1, Section 14.13.1]. Not only it produces the desired projection faster than anything known previously (including heuristic schemes), but it comes with a mathematical proof of converging to the correct projection point in all cases, i.e., it is completely reliable. Below we describe Eberly's method for ellipses and then adapt it to other quadratic curves and surfaces. In each case we provide a theoretical proof of convergence. We consider such proofs as an important asset of the proposed methods.

Ellipses. It will be sufficient to project a point (u, v) onto an ellipse in its canonical coordinates:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - 1 = 0. \tag{15}$$

Indeed, other ellipses can be translated and rotated to the canonical form (15), and then the projection point can be translated and rotated back to the original ellipse (the details are straightforward, we omit them).

Due to the obvious symmetry, it is enough to work in the first quadrant $u > 0, v > 0$; then the projection point (x, y) will also be in the first quadrant, i.e., $x > 0, y > 0$. (Other points can be reflected to the first quadrant about the axes, and then the projection point can be reflected back.) Also, we exclude the degenerate cases where $u = 0$ or $v = 0$; they are fairly simple and can be handled separately (see details in [1]).

Now the projection point (x, y) on the ellipse satisfies the orthogonality conditions (9), hence

$$u - x = tx/a^2 \qquad \text{and} \qquad v - y = ty/b^2 \tag{16}$$

for some real t (note that $t < 0$ for points inside the ellipse and $t > 0$ for points outside the ellipse). From (16) we find

$$x = \frac{a^2 u}{t + a^2} \qquad \text{and} \qquad y = \frac{b^2 v}{t + b^2} \tag{17}$$

Since $x, y > 0$, we have constraints $t > -a^2$ and $t > -b^2$. Assuming, as usual, that $a \geq b$ we get a single constraint $t > -b^2$. Substituting (17) into (15) we obtain a function

$$F(t) = \frac{a^2u^2}{(t + a^2)^2} + \frac{b^2v^2}{(t + b^2)^2} - 1, \tag{18}$$

whose root we need to find (because (x, y) must lie on the ellipse). Once we solve equation (18) for t, we can compute the projection point (x, y) by (17). Note that

$$\lim_{t \to -b^2+} F(t) = +\infty \qquad \text{and} \qquad \lim_{t \to \infty} F(t) = -1.$$

Taking the derivatives of F we see that

$$F'(t) = -\frac{2a^2u^2}{(t + a^2)^3} - \frac{2b^2v^2}{(t + b^2)^3} \tag{19}$$

and

$$F''(t) = \frac{6a^2u^2}{(t + a^2)^4} + \frac{6b^2v^2}{(t + b^2)^4}. \tag{20}$$

Thus on the interval $(-b^2, \infty)$ we have $F' < 0$ and $F'' > 0$, i.e., the function F is monotonically decreasing and concave; see Fig. 2. Thus standard Newton's method starting at any point t_0 where $F(t_0) > 0$ will converge to the unique root of F. Eberly suggests to start with $t_0 = bv - b^2$, because $F(t_0) > 0$ is guaranteed by (18). We found that it is more beneficial to start with

$$t_0 = \max\{au - a^2, bv - b^2\}. \tag{21}$$

Then Newton's method converges in 3-5 iterations in all practical cases and finds the root to within 7-8 significant digits. This is, on average, 2-3 times faster than solving equation of

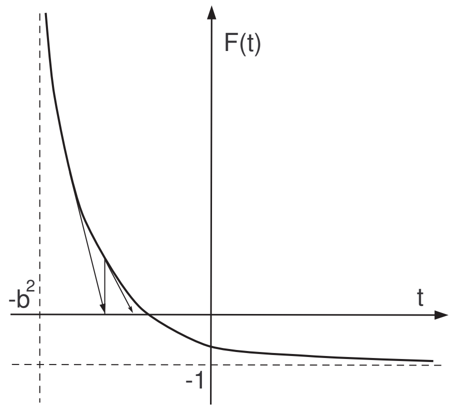

Figure 2. A typical graph of $F(t)$ for $t > -b^2$ and the progress of Newton's iterations toward the root.

degree four or using general heuristics [4, 7]. The MATLAB code for this method is posted on our web page [29].

Hyperbolas. Now let us project a point (u, v) onto a hyperbola. Again, the latter can be defined in its canonical coordinates:

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} - 1 = 0. \tag{22}$$

Due to symmetry, we restrict the method to $u > 0, v > 0$, then we also have $x > 0, y > 0$. The orthogonality conditions (9) now read

$$u - x = tx/a^2 \qquad \text{and} \qquad v - y = -ty/b^2, \tag{23}$$

from which

$$x = \frac{a^2 u}{t + a^2} \qquad \text{and} \qquad y = \frac{b^2 v}{-t + b^2} \tag{24}$$

Since $x, y > 0$, we have constraints $-a^2 < t < b^2$. Substituting (24) into (22) we obtain a function

$$F(t) = \frac{a^2 u^2}{(t + a^2)^2} - \frac{b^2 v^2}{(-t + b^2)^2} - 1, \tag{25}$$

whose root we need to find. Note that

$$\lim_{t \to -a^2+} F(t) = +\infty \qquad \text{and} \qquad \lim_{t \to b^2-} F(t) = -\infty.$$

Taking the derivatives of F we see that

$$F'(t) = -\frac{2a^2 u^2}{(t + a^2)^3} - \frac{2b^2 v^2}{(-t + b^2)^3} \tag{26}$$

hence $F' < 0$ for all $t \in (-a^2, b^2)$. Next,

$$F''(t) = \frac{6a^2 u^2}{(t + a^2)^4} - \frac{6b^2 v^2}{(-t + b^2)^4}. \tag{27}$$

Now F'' decreases from $+\infty$ (near $-a^2$) to $-\infty$ (near b^2), and it is monotonic (because $F''' < 0$, as one can easily verify). Thus F has a unique inflection point, t_*, within the interval $(-a^2, b^2)$. See Fig. 3, where two possible cases are shown: (a) the inflection point lies above the x axis, i.e., $F(t_*) > 0$ and (b) the inflection point lies below the x axis. The inflection point is found by solving $F'' = 0$, hence

$$t_* = \frac{b^2\sqrt{au} - a^2\sqrt{bv}}{\sqrt{au} + \sqrt{bv}}.$$

Now by computing $F(t_*)$ we can determine which case, (a) or (b), we have at hand. Standard Newton's method will converge to the root of $F(t) = 0$, but the starting point t_0 must be selected wisely. In the case (a) we need to choose t_0 such that $F(t_0) < 0$, and in the case (b) we need $F(t_0) > 0$. In the case (a) we can try points $t_k = b^2 - (b^2 - t_*)/2^k$ for $k = 1, 2, \dots$ until we find one where F is negative. In the case (b) we can try points $t_k = -a^2 + (t_* + a^2)/2^k$ for $k = 1, 2, \dots$ until we find one where F is positive. This process of choosing t_0 is relatively inexpensive and it does not involve the derivatives of F. This completes our projection algorithm for hyperbolas.

In practice, it works about 1.5 longer than the projection algorithm for ellipses, due to a more elaborate choice of t_0, so its speed is close to that of heuristic schemes [4, 7]. But it is guaranteed to converge, according to our analysis.

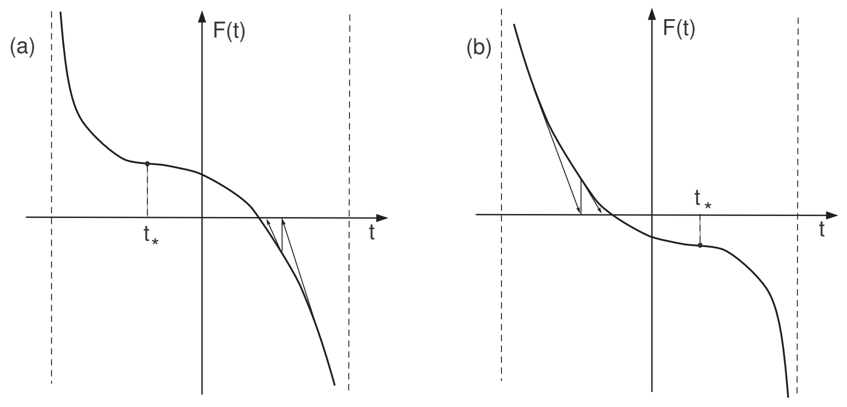

Figure 3. Two possible appearances of $F(t)$ on the interval $-a^2 < t < -b^2$. Arrows show the progress of Newton's iterations toward the root.

The MATLAB code for this method is posted on our web page [29].

Parabolas. Next let us project a point (u, v) onto a parabola. Again, the latter can be defined in its canonical coordinates:

$$y^2 - 2px = 0, \tag{28}$$

where $p > 0$ is the distance from the focus to the directrix. Due to symmetry, we restrict the method to $v > 0$, then we also have $y > 0$. The orthogonality conditions (9) now give

$$u - x = -pt \qquad \text{and} \qquad v - y = yt \tag{29}$$

from which

$$x = u + pt \qquad \text{and} \qquad y = \frac{v}{t+1} \tag{30}$$

Since $y > 0$, we have constraint $t > -1$. Substituting (30) into (28) we obtain a function

$$F(t) = \frac{v^2}{(t+1)^2} - 2pu - 2p^2 t, \tag{31}$$

whose root we need to find. Note that

$$\lim_{t\to -1+} F(t) = +\infty \qquad \text{and} \qquad \lim_{t\to\infty} F(t) = -\infty.$$

Taking the derivatives of F we see that $F'(t) = -\frac{2v^2}{(t+1)^3} - 2p^2$ and $F''(t) = \frac{6v^2}{(t+1)^4}$. Thus on the interval $(-1, \infty)$ we have $F' < 0$ and $F'' > 0$, i.e., the function F is monotonically decreasing and concave. Now standard Newton's method starting at any point t_0 where $F(t_0) > 0$ will converge to the unique root of F. We can try points $t_k = -1 + 2^{-k}$ for $k = 1, 2, \ldots$ until we find one where F is positive. The MATLAB code for this method is posted on our web page [29].

4. Projection onto quadrics

Here we describe the projection of a spacial point (u, v, w) onto quadratic surfaces of various kinds.

Ellipsoids. An ellipsoid is defined its canonical coordinates as follows:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} - 1 = 0, \tag{32}$$

where $a \geq b \geq c > 0$ are its semiaxes. (Other ellipsoids case be translated and rotated to the canonical form (32), and then the projection point can be translated and rotated back to the original ellipsoid.) Due to symmetry, we restrict the method to $u > 0, v > 0, w > 0$, then we also have $x > 0, y > 0, z > 0$. The orthogonality conditions now give

$$u - x = tx/a^2, \qquad v - y = ty/b^2, \qquad w - z = tz/c^2 \tag{33}$$

for some scalar t, from which

$$x = \frac{a^2 u}{t + a^2}, \qquad y = \frac{b^2 v}{t + b^2} \qquad z = \frac{c^2 w}{t + c^2} \tag{34}$$

Since $x, y, z > 0$, we have constraint $t > \max\{-a^2, -b^2 - c^2\} = -c^2$. Substituting (34) into (32) we obtain a function

$$F(t) = \frac{a^2 u^2}{(t+a^2)^2} + \frac{b^2 v^2}{(t+b^2)^2} + \frac{c^2 w^2}{(t+c^2)^2} - 1, \tag{35}$$

whose root we need to find. Note that

$$\lim_{t\to -c^2+} F(t) = +\infty \qquad \text{and} \qquad \lim_{t\to\infty} F(t) = -1.$$

Taking the derivatives of F we see that

$$F'(t) = -\frac{2a^2u^2}{(t+a^2)^3} - \frac{2b^2v^2}{(t+b^2)^3} - \frac{2c^2w^2}{(t+c^2)^3} \tag{36}$$

and

$$F''(t) = \frac{6a^2u^2}{(t+a^2)^4} + \frac{6b^2v^2}{(t+b^2)^4} + \frac{6c^2w^2}{(t+c^2)^4}. \tag{37}$$

Thus on the interval $(-c^2, \infty)$ we have $F' < 0$ and $F'' > 0$, i.e., the function F is monotonically decreasing and concave, just as in Fig. 2. Thus standard Newton's method starting at any point t_0 where $F(t_0) > 0$ will converge to the unique root of F, and we choose (see (21))

$$t_0 = \max\{au - a^2, bv - b^2, cw - c^2\}.$$

Hyperbolic paraboloids. Now let us project a point (u, v, w) onto a hyperbolic paraboloid ("saddle") defined in its canonical coordinates as

$$\frac{x^2}{a^2} - \frac{y^2}{b^2} - z = 0. \tag{38}$$

Due to symmetry, we restrict the method to $u > 0, v > 0$, then we also have $x > 0, y > 0$. The orthogonality conditions now give

$$u - x = tx/a^2, \qquad v - y = -ty/b^2, \qquad w - z = -t/2 \tag{39}$$

for some scalar t, from which

$$x = \frac{a^2u}{t+a^2}, \qquad y = \frac{b^2v}{-t+b^2}, \qquad z = w + \frac{t}{2}. \tag{40}$$

Since $x, y > 0$, we have constraints $-a^2 < t < b^2$. Substituting (40) into (38) we obtain a function

$$F(t) = \frac{a^2u^2}{(t+a^2)^2} - \frac{b^2v^2}{(-t+b^2)^2} - w - \frac{t}{2}, \tag{41}$$

whose root we need to find. Note that

$$\lim_{t\to -a^2+} F(t) = +\infty \qquad \text{and} \qquad \lim_{t\to b^2-} F(t) = -\infty.$$

Taking the derivatives of F we see that

$$F'(t) = -\frac{2a^2u^2}{(t+a^2)^3} - \frac{2b^2v^2}{(-t+b^2)^3} - \frac{1}{2} \tag{42}$$

hence $F' < 0$ for all $t \in (-a^2, b^2)$. Next,

$$F''(t) = \frac{6a^2u^2}{(t+a^2)^4} - \frac{6b^2v^2}{(-t+b^2)^4}. \tag{43}$$

Now F'' decreases from $+\infty$ (near $-a^2$) to $-\infty$ (near b^2), and it is monotonic (because $F''' < 0$, as one can easily verify). Thus F has a unique inflection point, t_*, within the interval $(-a^2, b^2)$. Our further analysis repeats that done for hyperbolas in the previous section.

Hyperboloids. Now let us project a point (u, v, w) onto a hyperboloid (one sheet) defined in its canonical coordinates as

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} - \frac{z^2}{c^2} - 1 = 0, \tag{44}$$

where we can assume $a \geq b$. Due to symmetry, we restrict the method to $u > 0, v > 0, w > 0$, then we also have $x > 0, y > 0, z > 0$. The orthogonality conditions now give

$$u - x = tx/a^2, \qquad v - y = ty/b^2, \qquad w - z = -tz/c^2 \tag{45}$$

for some scalar t, from which

$$x = \frac{a^2 u}{t + a^2}, \qquad y = \frac{b^2 v}{t + b^2}, \qquad z = \frac{c^2 w}{-t + c^2}. \tag{46}$$

Since $x, y, z > 0$, we have constraints $-b^2 < t < c^2$. Substituting (46) into (44) we obtain a function

$$F(t) = \frac{a^2 u^2}{(t + a^2)^2} + \frac{b^2 v^2}{(t + b^2)^2} - \frac{c^2 w^2}{(-t + c^2)^2} - 1, \tag{47}$$

whose root we need to find. Note that

$$\lim_{t \to -b^2+} F(t) = +\infty \qquad \text{and} \qquad \lim_{t \to c^2-} F(t) = -\infty.$$

Taking the derivatives of F we see that

$$F'(t) = -\frac{2a^2 u^2}{(t + a^2)^3} - \frac{2b^2 v^2}{(t + b^2)^3} - \frac{2c^2 w^2}{(-t + c^2)^3} \tag{48}$$

hence $F' < 0$ for all $t \in (-b^2, c^2)$. Next,

$$F''(t) = \frac{6a^2 u^2}{(t + a^2)^4} + \frac{6b^2 v^2}{(t + b^2)^4} - \frac{6c^2 w^2}{(-t + c^2)^4}. \tag{49}$$

Again, as before, F'' decreases from $+\infty$ (near $-b^2$) to $-\infty$ (near c^2), and it is monotonic (because $F''' < 0$, as one can easily verify). Thus F has a unique inflection point, t_*, within the interval $(-b^2, c^2)$. Its graph looks like one of those shown in Fig. 3.

But now it is not easy to determine which case we have at hand – the one shown in part (a) or in part (b) of Fig. 3 (because we cannot solve equation $F'' = 0$). However, one of the two iterative procedures described in the case of hyperbolas (i.e., Newton's method working from the left and another one – from the right), must work.

Thus we simply can choose one of these two procedures at random and follow it hoping that it converges. But if it fails, i.e., if an iteration lands outside the interval $(-b^2, c^2)$, then we switch to the other procedure, and it will surely converge. We note that even if we start on the wrong side, Newton's iteration may land on the right side and then converge.

As there is a 50% chance of choosing one of the two sides correctly at random, the current projection method is perhaps about 1.5 times slower, on average, than the previous one (a moderate price to pay for extra complications). We emphasize that our analysis still guarantees that the method converges to the correct projection point in all cases.

Other quadrics. We have covered three major types of quadratic surfaces in 3D. There are two others – elliptic paraboloid and hyperboloid of two sheets, which are treated very similarly, with small variations in each case that we leave out.

5. Conclusion

The main goal of our chapter is to put a strong argument against the customary presumption that fitting ellipses and other quadratic curves/surfaces by minimizing geometric distances is a prohibitively difficult task. It is no longer so! Three major breakthroughs have occurred in the last decade:

- Ahn et al. have designed a general fitting scheme for implicit curves and surfaces that is surprisingly simple and fast. Its most time-consuming part is the projection of data points onto the curve/surface.
- Eberly discovered a remarkably fast and totally reliable projection algorithm for ellipses (which we generalize here to other conics and quadrics).
- Taubin's algebraic fit gained the reputation of being able to provide a balanced (nearly unbiased) initialization for the subsequent iterative procedure.

Combining these three advances together gives a complete fitting scheme for fitting quadratic curves and surfaces of all kinds. This scheme is reliable and efficient.

Acknowledgement. N.C. was partially supported by National Science Foundation, grant DMS-0969187.

Appendix

Here we review non-geometric (algebraic) fits that are used to provide an initial guess, i.e., a curve that initializes an iterative procedure solving the geometric fitting problem (4).

Suppose again that one fits an implicit curve $P(x, y; \Theta) = 0$ to observed points (x_1, y_1), ..., (x_n, y_n). Perhaps the simplest non-geometric fit is the one minimizing

$$\mathcal{F}_1(\Theta) = \sum_{i=1}^{n} [P(x_i, y_i; \Theta)]^2. \tag{50}$$

To justify this method one usually notes that $P(x_i, y_i; \Theta) = 0$ if and only if the point (x_i, y_i) lies on the curve, and $[P(x_i, y_i; \Theta)]^2$ is small when the point lies near the curve. The minimization of (50) is called *algebraic fit* and $|P(x_i, y_i; \Theta)|$ is called the corresponding *algebraic distance*.

When the curve is defined by an algebraic equation, such as

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = 0, \tag{51}$$

then an unconstrained minimization of (50) produces the unwanted degenerate solution: $A = B = C = D = E = F = 0$. To avoid it, one can impose a constraint, such as $A^2 + B^2 + C^2 + D^2 + E^2 + F^2 = 1$. The resulting fit would not be invariant under rotations or translations of the data set, i.e., the resulting curve would depend on the choice of the coordinate system, which is hardly acceptable; see [8, 19].

Better constraints (i.e., those that are invariant under translations and rotations) are $A + C = 1$ (see [19]), or $A^2 + B^2/2 + C^2 = 1$ (see [8]), or $4AC - B^2 = 1$ (see [18]). The last constraint guarantees that the resulting curve will be an ellipse (rather than a hyperbola or parabola).

But practical experience shows that all algebraic fits, with or without constraints, are statistically inaccurate and biased, in one way or another. The main reason is that algebraic distances may be substantially different from geometric distances [28].

This defect can be compensated for by using linear approximation

$$\frac{|P(x_i, y_i; \Theta)|}{\|\nabla P(x_i, y_i; \Theta)\|} = d_i + \mathcal{O}(d_i^2)$$

where $\nabla P = \big(\partial P/\partial x, \partial P/\partial y\big)$ denotes the gradient vector. This leads to an 'approximate geometric fit', which minimizes

$$\mathcal{F}_2(\Theta) = \sum_{i=1}^{n} \frac{[P(x_i, y_i; \Theta)]^2}{\|\nabla P(x_i, y_i; \Theta)\|^2}. \tag{52}$$

This method is called *gradient weighted algebraic fit*. It was applied to quadratic curves by Sampson [24] and popularized by Taubin [26].

If the curve is defined by an algebraic equation, such as (51), then both numerator and denominator of each fraction in (52) are homogeneous quadratic polynomials of the parameters. As a result, $\mathcal{F}_2$ is invariant under scalings of the parameter vector (A, B, C, D, E, F), hence no additional constraints are needed anymore. The minimization of (52) produces a more accurate fit than the simple algebraic fits (50) do; see statistical analysis in [12].

But the problem of minimizing (52) has no closed form solution and must be solved by iterations. Various iterative schemes for the minimization of (52) have been developed (see [16, 17, 21]); some of them have become standard in computer vision industry. They are all too complex to be used for an initialization of the geometric fit (4) (besides, each of them needs its own initialization to get started...). In this sense, the minimization of (52) and that of (4) can be regarded as two independent approaches to the task of fitting curves to data. While (4) is called Maximum Likelihood Estimation (MLE), that of (52) is called Approximate Maximum Likelihood Estimation (AMLE); see [16, 17].

One may wonder if the AMLE (52) can be used *instead of* the MLE (4), as the minimization of (52) is technically simpler than that of (4). Most researchers, however, agree that the answer is NO, i.e., the minimization of (4) would produce a better fit than that of (52); see comments in [7, p. 12]. (Though a complete statistical analysis has yet to be done.)

Taubin [26] simplified (52) and converted it into a non-iterative fit that minimizes

$$\mathcal{F}_3(\Theta) = \frac{\sum [P(x_i, y_i; \Theta)]^2}{\sum \|\nabla P(x_i, y_i; \Theta)\|^2}. \tag{53}$$

Note that Taubin simply summed up all the numerators and all the denominators in (52) separately.

If one fits conics defined by algebraic equation (51), then both numerator and denominator of (53) are homogeneous quadratic polynomials of the components of the parameter vector $\mathbf{A} = (A, B, C, D, E, F)^T$. Thus one can rewrite (53) as

$$\mathcal{F}_4(\mathbf{A}) = \frac{\mathbf{A}^T \mathbf{M} \mathbf{A}}{\mathbf{A}^T \mathbf{N} \mathbf{A}} \quad \rightarrow \quad \min, \tag{54}$$

where $\mathbf{M}$ and $\mathbf{N}$ are some 6×6 symmetric positive semi-definite matrices. Since $\mathcal{F}_4(\mathbf{A})$ is invariant under scalings of the vector $\mathbf{A}$, one can solve (54) by minimizing $\mathcal{F}_5(\mathbf{A}) = \mathbf{A}^T \mathbf{M} \mathbf{A}$ under the constraint $\mathbf{A}^T \mathbf{N} \mathbf{A} = 1$. Introducing a Lagrange multiplier η we can minimize the function

$$\mathcal{F}_6(\mathbf{A}, \eta) = \mathbf{A}^T \mathbf{M} \mathbf{A} - \eta(\mathbf{A}^T \mathbf{N} \mathbf{A} - 1).$$

Differentiating with respect to $\mathbf{A}$ gives the first order necessary condition

$$\mathbf{M}\mathbf{A} = \eta \mathbf{N} \mathbf{A}, \tag{55}$$

thus $\mathbf{A}$ must be a generalized eigenvector of the matrix pair $(\mathbf{M}, \mathbf{N})$. Moreover, premultiplying (55) by $\mathbf{A}$ we see that $\mathbf{A}^T \mathbf{M} \mathbf{A} = \eta$, and because we are minimizing $\mathbf{A}^T \mathbf{M} \mathbf{A}$, the desired vector $\mathbf{A}$ must correspond to the *smallest* (non-negative) eigenvalue η. (We note that $\mathbf{N}$ here is singular; one usually eliminates F to reduce $\mathbf{A}$ to a 5-vector $\mathbf{A}' = (A, B, C, D, E)^T$ and the 6×6 problem (55) to a 5×5 problem $\mathbf{M}'\mathbf{A}' = \eta' \mathbf{N}' \mathbf{A}'$, where the 5×5 matrix $\mathbf{N}'$ is positive definite; see details in [22].)

Solving a generalized eigenvalue problem takes just one call of a standard matrix function (such functions are included in most modern software packages, e.g., in MATLAB). Thus Taubin's fit is regarded as a fast non-iterative procedure. In practice the Taubin's fit is only marginally slower than the simple algebraic fit minimizing (50).

Its advantage is that Taubin's fit is more balanced than any algebraic fit. It has a much smaller bias; see statistical analysis in [22]. Its disadvantage is that it produces a conic that may be of any kind – an ellipse, a hyperbola, or a parabola. If one fits conics of a certain type (e.g., ellipses), then Taubin's fit must be supplemented with another simple fit whenever it gives the wrong curve. Experimental tests show that Taubin's fit provides a better initialization of iterative procedures than simple algebraic fits do [14].

References

[1] *3D Game Engine Design*, 2nd ed., Morgan Kaufmann Publishers, San Francisco, CA, 2007. See also Internet article *Distance from a point to an ellipse in 2D*, Geometric Tools, LLC, www.geometrictools.com

[2] Aigner, M. & Jüttler, B. (2005). Robust computation of foot points on implicitly defined curves, In: Editors Daehlen, M. et al., *Mathematical Methods for Curves and Surfaces*, Tromso 2004, Nashboro Press, pp. 1–10.

[3] Aigner, M. & Jüttler, B. (2008). Gauss-Newton type techniques for robustly fitting implicitly defined curves and surfaces to unorganized data points, In: *Shape Modeling International*, pp. 121–130.

[4] Ahn, S. J., Rauh, W. & Warnecke, H. J. (2001). Least-squares orthogonal distances fitting of circle, sphere, ellipse, hyperbola, and parabola, *Pattern Recog.*, **34**, 2283–2303.

[5] Ahn, S. J., Rauh, W. & Recknagel, M. (2001). Least squares orthogonal distances fitting of implicit curves and surfaces, In: LNCS **2191**, 398–405.

[6] Ahn, S. J., Rauh, W. & Cho, H. S. (2002). Orthogonal distances fitting of implicit curves and surfaces, *IEEE trans. PAMI*, **24**, 620–638.

[7] Ahn, S. J. (2004). Least Squares Orthogonal Distance Fitting of Curves and Surfaces in Space, In: LNCS **3151**, Springer, Berlin.

[8] Bookstein, F. L. (1979). Fitting conic sections to scattered data, *Comp. Graph. Image Proc.* **9**, 56–71.

[9] Chan, N. N. (1965). On circular functional relationships, *J. R. Statist. Soc. B*, **27**, 45–56.

[10] Cheng, C.-L. & Van Ness, J. W. (1999). *Statistical Regression with Measurement Error*, Arnold, London.

[11] Chernov, N., Ososkov, G. & Silin, I. (2000). Robust fitting of ellipses to non-complete and contaminated data, *Czech. J. Phys.* **50**, 347–354.

[12] Chernov, N. & Lesort, C. (2004). Statistical efficiency of curve fitting algorithms, *Comp. Stat. Data Anal.*, **47**, pp. 713–728.

[13] Chernov, N. & Lesort, C. (2005). Least squares fitting of circles, *J. Math. Imag. Vision* **23**, 239–251.

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

[15] Chernov, N. (2010). *Circular and linear regression: Fitting circles and lines by least squares*, Chapman & Hall/CRC Monographs on Statistics and Applied Probability **117**.

[16] Chojnacki, W., Brooks, M. J. & van den Hengel, A. (2001). Rationalising the renormalisation method of Kanatani, *J. Math. Imag. Vision*, **14**, 21–38.

[17] Chojnacki, W., Brooks, M. J., van den Hengel, A. & Gawley, D. (2005). FNS, CFNS and HEIV: A unifying approach, *J. Math. Imag. Vision*, **23**, 175–183.

[18] Fitzgibbon, A. W., Pilu, M. & Fisher, R. B. (1999). Direct Least Squares Fitting of Ellipses, *IEEE Trans. PAMI* **21**, 476–480.

[19] Gander, W., Golub, G. H. & Strebel, R. (1994). Least squares fitting of circles and ellipses, *BIT*, **34**, 558–578.

[20] Geometric Product Specification (GPS) – Acceptance and representation test for coordinate measuring machines (CMM) – Part 6: Estimation of errors in computing Gaussian associated features. Int'l Standard ISO 10360-6. IS, Geneva, Switzerland (2001).

[21] Kanatani, K. (1994). Statistical bias of conic fitting and renormalization, *IEEE Trans. PAMI*, **16**, 320–326.

[22] Kanatani, K. (2008). Statistical optimization for geometric fitting: Theoretical accuracy bound and high order error analysis, *Int. J. Computer Vision* **80**, 167–188.

[23] Leedan, Y. & Meer, P. (2000). Heteroscedastic regression in computer vision: Problems with bilinear constraint, *Intern. J. Comp. Vision*, **37**, 127–150.

[24] Sampson, P. D. (1982). Fitting conic sections to very scattered data: an iterative refinement of the Bookstein algorithm, *Comp. Graphics Image Proc.* **18**, 97–108.

[25] Sturm, P., & Gargallo, P. (2007). Conic fitting using the geometric distance, *Proc. Asian Conf. Comp. Vision*, Tokyo, Japan, **2**, pp. 784–795.

[26] Taubin, G. (1991). Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation, *IEEE Trans. PAMI*, **13**, 1115–1138.

[27] Wijewickrema, S., Papliński, A. & Esson, Ch. (2006). Orthogonal distance fitting revisited, *Tech. report, Clayton School Inf. Technol.*, Monash U., Melbourne, 2006/205.

[28] Zhang, Z. (1997). Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting, *Intern. J. Image Vision Comput.*, **15**, 59–76.

[29] http://www.math.uab.edu/ chernov/cl