Improving The Stability Of Algebraic Curves For Applications

Tolga Tasdizen
Div. of Engineering
Brown University
Providence, RI 02906
USA
tt@lems.brown.edu
voice: 401-863-2760
fax: 401-863-1157

Jean - Philippe Tarel
INRETS (LIVIC)
2, av. du Gen. Malleret-Joinville
BP 34-94114 ARCUEIL-CEDEX
France
Jean-Philippe.Tarel@inrets.fr
voice: ++33-1-47-40-7297
fax: ++33-1-45-47-5606

David B. Cooper
Div. of Engineering
Brown University
Providence, RI 02906
USA
cooper@lems.brown.edu
voice: 401-863-2601
fax: 401-863-1157

Abstract

An algebraic curve is defined as the zero set of a polynomial in two variables. Algebraic curves are practical for modeling shapes much more complicated than conics or superquadrics. The main drawback in representing shapes by algebraic curves has been the lack of repeatability in fitting algebraic curves to data. Usually, arguments against using algebraic curves involve references to mathematicians Wilkinson (see [1] chapter 7) and Runge (see [3] chapter 4). The first goal of this article is to understand the stability issue of algebraic curve fitting. Then a fitting method based on ridge regression and restricting the representation to well behaved subsets of polynomials is proposed, and its properties are investigated. The fitting algorithm is of sufficient stability for very fast position-invariant shape recognition, position estimation, and shape tracking, based on invariants and new representations. Among appropriate applications are shape-based indexing into image databases.

Acknowledgment This work was partially supported by NSF Grant #IIS-9802392.

1 Introduction

Algebraic $2D$ curves (and $3D$ surfaces) are extremely powerful for shape recognition and single-computation pose estimation because of their fast fitting, invariants, and interpretable coefficients, [10, 11, 13, 17, 18, 19, 21]. Significant advantages over Fourier Desciptors are their applicability to non-star shapes, to open curves, to curves that contain gaps, and to unordered curve data, Sec. 2. Under circumstances where these issues are not relevant, polynomials based on Fourier analysis may be very effective, and an interesting formulation relating Fourier series and polynomials is given in [14]. A weakness for use of algebraic curves and surfaces has been lack of stability of parameters. This paper, studies the problem and provides a solution.

The classical least-squares fitting of algebraic curves, Sec. 3, especially the more interesting cases of higher degree polynomials, suffers three major problems: local inconsistency with the continuity of the dataset; local over-sensitivity of the polynomial zero set around the data to small data perturbations; instability of the coefficients due to excessive degrees of freedom in the polynomial. Substituting an approximate Euclidean distance for algebraic distance [21] is much more stable than the classical least squares algorithm, in many cases gives useful fits, but in other cases the improvement is not sufficient to solve these major problems. Similarly, the use of the exact Euclidean distance provides better results than the algebraic distance [16]; nevertheless the fitting is sometimes not stable enough and the minimization is solved iteratively, a time consuming process. Another attempt to improve the stability of the fit was the development of fitting algorithms which ensure that the obtained zero set is bounded [8, 22, 13], but the last one is for $2nd$ degree curves only, and increased stability for all and fitting speed for the former two are still desired. Non-linear parametrizations of polynomials that are guaranteed to satisfy certain topological properties – boundedness and having a zero set that is contained within another shape such as an ellipse – that have recently appeared [9] are interesting, and their relative merits need to be studied further. The problem of an excessive number of parameters in *implicit polynomial* (IP) representations was first studied in [15] in the framework of Bayesian estimation. The linear 3L fitting algorithm [10] exhibits significantly improved curve representation accuracy and stability but there is significant value to further improvement in coefficient stability in order that algebraic curves be generally applicable for object-recognition purposes.

Following a short summary on algebraic curves in Sec. 2 and the classical least-squares fitting in Sec. 3, we investigate the stability problems of 1D polynomials in Sec. 4. In Sec. 5, the solution of the first and second problems by the 3L method [10] is analyzed from the point of view of Sec. 4. In Sec. 6, we present a new linear algorithm which produces accurate and stable curve-data representations and stable coefficients. Results of object recognition experiments based on this algorithm and a new set of invariants [18] are presented in Sec. 7.

2 Representations of Algebraic Curves

Formally, an algebraic curve is specified by a 2D IP of degree n given by the following equation:

$$f_n(x,y) = \sum_{0\le j+k\le n} a_{jk}x^jy^k = a_{00} + a_{10}x + a_{01}y + \ldots + a_{n0}x^n + a_{n-1\ 1}x^{n-1}y + \ldots + a_{0n}y^n = 0$$

The *homogeneous binary polynomial* of degree r in x and y is a form, e.g., $a_{20}x^2 + a_{11}xy + a_{02}y^2$ is the 2^{nd} degree form. The homogeneous polynomial of degree n is the so-called leading form. An algebraic curve of degree 2 is a conic, degree 3 a cubic, degree 4 a quartic, and so on. Polynomial $f_n(x,y)$ is represented by the coefficient vector $(a_{jk})_{0\le j,k;0\le j+k\le n}$ which has dimension $p = \frac{1}{2}(n+1)(n+2)$:

$$f_n(x,y) = Y^tA \tag{1}$$

where [1] $A = [\, a_{00}\; a_{10} \ldots a_{n0}\; a_{n-1\;1}\; \ldots\; a_{0n}\,]^t$ and $Y = [\, 1\;\; x\;\; \ldots\;\; x^n\;\; x^{n-1}y\;\; \ldots\;\; y^n\,]^t$. In general, the vector notation is convenient for IP fitting since fitting can be set within a linear framework as detailed in Sec. 3. A shape is represented by the *zero set* of $f_n(x)$, i.e., the set of points $\{x, y\}$ satisfying the IP equation $f_n(x, y) = 0$ which is the intersection of the surface defined by an explicit polynomial $z = f_n(x, y)$ with the plane $z = 0$, see Fig. 1.

The IP framework for shape representation and recognition is generally compared with *Fourier Shape Descriptors.* Here we briefly summarize some of the differences between these approaches. The two main types of Fourier Descriptors are: (i) that which represents a shape as a radius as a function of angle, and (ii) that which uses a complex valued function to represent the coordinates of the points along the curve as a function of arc-length. There are certain disadvantages to both approaches. (i) is limited to the set of *star-shapes* which can be represented by a single-valued radius as a function of angle; however, interesting shapes generally do not fall into this category. (ii) requires the input data sets to be an ordered set of points. (i) can not be directly used for open shapes, a preprocessing step to artificially close the curve is required. (ii) can be used for open curves, but serious difficulties with arc-length normalization arise. Arc-length parametrization is the main drawback of (ii) because arc-length can increase significantly if noise is added on to the curve. Both have problems with varying data point density and gaps in the data. IPs do not suffer from any of the problems listed above: they are directly applicable to non-star shapes, open curves, unordered data sets and are robust to noisy data sets and inhomogoneously spaced data points. The main advantage of Fourier Descriptors over IPs have been their better stability because they are an *explicit* representation. This paper focuses on the stability issue with IPs.

[1] Superscript ^{t} denotes vector and matrix transpose.

3 Classical Least-Squares Fitting

The classical and simplest way to fit an algebraic curve to data is to minimize the algebraic distance over the set of given data points (x_j, y_j), $1 \leq j \leq m$, that is

$$e_{algebraic} = \sum_{1\leq j\leq m} (f_n(x_j, y_j))^2 = A^t \underbrace{\left(\sum_{1\leq j\leq m} Y_j Y_j^t \right)}_{MM^t = S} A \tag{2}$$

by using vector representation of f_n as in (1). Define the matrix of monomials as the $p \times m$ matrix $M = [Y_1\ Y_2 \ldots Y_m]$ (or more generally, the *design matrix*), and $S = MM^t = \sum_{1\leq j\leq m} Y_j Y_j^t$ is the *scatter matrix* of the monomials. To avoid the trivial zero solution in the minimization of (2), a constraint such as $\|A\|^2 = 1$ is imposed which modifies the problem to

$$\min_A \left(A^t \left(\sum_{1\leq j\leq m} Y_j Y_j^t \right) A + \lambda (A^t A - 1) \right) \tag{3}$$

with the introduction of Lagrange multiplier λ. The solution to (3) is given by the unit eigenvector A associated with λ_{min}, the smallest eigenvalue of $SA = \lambda A$ [21]. Consequently, the classical least-squares fitting algorithm consists of computing the monomial scatter matrix S from a set of data points, and then finding the unit eigenvector of S associated with its smallest eigenvalue. Although this algorithm is affine invariant [4, 21], most of the time it is not of any practical use due to the following problems: The fitted zero set does not respect the continuity of the original data set as illustrated in Fig. 2(a)-(d) and Fig. 6(a) and (d). This problem undermines the use of classical fitting for obtaining good representations of the data. Moreover, results are highly sensitive to small errors in the data. Even seemingly negligible perturbations in the data can lead to zero sets that have no resemblance to the zerosets prior to the perturbations in the data, Fig. 2(a)-(d). Even with low order degrees, depending on the structure of the given data set, S may not provide a stable unique eigenvector under small perturbations. For example, several eigenvalues can have similar values to the smallest one, and thus the solution will span a subspace in the coefficient space when small perturbations are added to the data set. Consequently, classical fitting is also practically useless for recognition purposes based on the coefficients of the fitted polynomials.

4 Pathological Polynomials

Although we are interested in 2D polynomials, i.e., functions of x and y, it is instructive to first study stability in the 1D case. It is well known that some 1D polynomials, in particular polynomials of high degree, are ill-conditioned. Consider the pathological example due to Wilkinson [1]: $(x+1)(x+2)\ldots(x+20) = x^{20} + 210x^{19} + \ldots + 20!$. This polynomial has very large coefficients and its roots are -1, -2, -3,..., -20. An accurate calculation of the perturbed roots as given in [1] to five decimals after a tiny change of 2^{-23} is applied to the coefficient of x^{19}, are shown in Table 1. Though this example demonstrates how ill conditioned some polynomials are, it does not mean that all polynomials are so, and as a consequence that all algorithms using high degree polynomials have to be rejected as a priori unstable. In fact, we will demonstrate that it is possible to work in a subspace of non-pathological polynomials. First, let's try to understand the pathology of this polynomial. A plot of this polynomial would show varying oscillation amplitude between its roots. This type of ill-conditioned behavior of polynomials is well-known in the context of interpolation theory. Indeed, the Wilkinson polynomial is an example of Lagrange interpolation at 20 points, and it is known that Lagrange interpolation suffers from oscillation problems between data points. This is the so-called Runge problem [3]. One known solution is to change the way the interpolation is carried out. Hermite interpolation, where the first derivative of the polynomial is controlled in addition to the value of the polynomial at each given point, can be proven to converge properly for all continuous functions when the number of sampling points and thus the degree of the polynomial increases.

We are referring to interpolation theory and Hermite polynomials because they provide us with very useful insight in trying to improve the classical least-squares fitting algorithm. In essence, the problem with polynomials is that the functional relationship between its coefficients and its roots is highly non-linear. Let $p_n(x)$ be a 1D polynomial defined as: $p_n(x) = \sum_{0 \leq j \leq n} a_j x^j = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$. Roots x_k of this polynomial are defined by $p_n(x_k) = 0$. This last equation can be seen as an implicit equation for root x_k where this root is a function of coefficients a_j. To determine the sensitivity of this root to small changes of the coefficients, we differentiate $p_n(x_k) = 0$ with respect to a_j. We obtain $x_k^j + \frac{dp_n}{dx}(x_k)\frac{\partial x_k}{\partial a_j} = 0$ which is equivalent to

$$\frac{\partial x_k}{\partial a_j} = -\frac{x_k^j}{\frac{dp_n}{dx}(x_k)} \quad . \tag{4}$$

(4) has important consequences. It is desired that small or large changes in the coefficients produce small or large changes, respectively, in the roots, and vice versa. Thus we should require that $\frac{\partial x_k}{\partial a_j} = 1$. Due to the numerator x_k^j, we see that x_k should be close to values 1.0 or -1.0; otherwise, the effect of a small coefficient perturbation has a larger effect on roots with large absolute values. This explains why roots with large absolute values are less stable than others for the Wilkinson polynomial, Table 1. Due to the denominator of (4), we deduce that the sensitivity to a small coefficient perturbation is also directly dependent on the value of the first derivative of the polynomial at the root location. The Wilkinson's polynomial has derivatives $19!0!$, $-18!1!$, $17!2!$, $\ldots$, $-0!19!$ at -1, -2, -3, $\ldots$, -20 respectively. These huge variations in the first derivative $\frac{dp_n}{dx}$ contributes to the instability of the roots with respect to coefficient perturbations. Using (4), we predict, with a first order Taylor expansion, the perturbations of the roots of the Wilkinson polynomial when 2^{-23} is added to $a_{19} = 210$: These values are in good accordance with the differences between the original roots, $-20, \ldots, -1$ and the

real perturbed roots, Table 1. Tables 1 and 2 provide an experimental validation of (4).

5 Gradient-one Fitting

The insight developed in Sec. 4 into how polynomials can be ill-conditioned, enables us to determine a subset of well-conditioned polynomials. What is a well-conditioned polynomial? For the problem at hand, it is a polynomial for which the relationship between its roots and its coefficients is such that small changes in one induces small changes in the other and larger changes induce larger changes. In Sec.4, it was argued that a 1D polynomial should have root values and first derivative values at the root locations, all close to 1.0 or -1.0. We can extend this result to 2D polynomials: a set of polynomials satisfying these constraints exactly in 2D are the powers of the unit circle: $\frac{1}{2n}((x^2+y^2)^n-1)$. Members of the set of polynomials "close" to these polynomials in the coefficient space are well-conditioned. The topology of this set remains to be studied in our future work.

The first requirement for stable fitting is to apply a data set standardization to force the data points to be close to the unit circle, and thus indirectly to force the zero set of the polynomial to be as close as possible to the unit circle. The data set standardization consists of centering the data-set center of mass at the origin of the coordinate system and then scaling it by dividing the coordinates of each point by the average of the square roots of the eigenvalues of the 2×2 matrix of second order moments (normalized by the number of points in the data set). This is a Euclidean invariant measure of the object size and can be thought of as the average radius of the data points from the object center. Thus by data set standardization we are setting this measure of object size to 1.

The second requirement is to control the value of the first derivatives along the zero set, i.e, the gradient of the 2D polynomial:

$$\nabla f_n(x_j,y_j) = \begin{bmatrix} \frac{\partial f_n}{\partial x} \\ \\ \frac{\partial f_n}{\partial y} \end{bmatrix} (x_j,y_j) \tag{5}$$

The gradient vector along the zero set of the polynomial is always perpendicular to the curve defined by the zero set. Thus, if we can compute the local tangent to the curve at each point of the data set, we propose to constrain the gradient to be perpendicular to the local tangent and with unit norm. This will force the zero set of the polynomial to respect the local continuity of the data set. The calculation of the tangent to the data set at a point does not pose a serious problem. If the data set is ordered as a curve, we calculate local tangents to the data using the lines going through consecutive data points. If the data is not ordered, a fast distance transform [23, 17] can be used to generate level sets as in 3L [10] or to indirectly calculate tangent directions. When working with real images, level sets may also be generated as described in [12]. Or if the input to the fitting algorithm comes from an edge detector, edge orientations can be used as the tangent directions. The normal direction is the direction perpendicular to the tangent direction and pointing towards the outside of the object. In the case of open curves where no notion of inside/outside is available, both cases can be considered resulting in two fitted IPs which have coefficient vectors related by multiplication with -1. We do not apply any smoothing in computing the tangents even in the presence of noise; indeed, it is the fitting process which takes care of smoothing the fluctuations in the tangent direction along the curve given that there are enough points on the dataset (at least a few times the dimensionality of the IP coefficient vector). The proposed fitting technique is set as a least-squares problem with the following additional constraints: Local tangential and normal directional derivatives of the IP must be as close as possible to 0 and 1, respectively. These constraints add two terms to (2) to yield

$$e_{grad} = \sum_{j=1}^{m} \left(f_n(x_j, y_j)^2 + \mu \left((N_j^t \nabla f_n - 1)^2 + (T_j^t \nabla f_n)^2 \right) \right) \tag{6}$$

where T_j and N_j are the local tangent and normal at (x_j, y_j) and μ is the relative weight on the gradient with respect to the f^2 term. By using the vector notation (1) in (5), we deduce the vector form of the gradient:

$$\nabla f_n = \underbrace{\begin{bmatrix} \frac{\partial Y}{\partial x} \\ \\ \frac{\partial Y}{\partial y} \end{bmatrix}^t}_{dimension:2\times p} \underbrace{A}_{p\times 1} = \nabla Y^t A$$

where $p = \frac{1}{2}(n+1)(n+2)$ is the number of coefficients of a binary polynomial of degree n. And then after substitution in (6), we expand e_{grad} as:

$$e_{grad} = A^t \underbrace{\sum Y_j Y_j^t}_{S} A + \mu A^t \underbrace{\sum \nabla Y_j N_j N_j^t \nabla Y_j^t}_{S_N} A + \mu A^t \underbrace{\sum \nabla Y_j T_j T_j^t \nabla Y_j^t}_{S_T} A - 2\mu A^t \underbrace{\sum \nabla Y_j N_j}_{G_N} + \mu m$$

In this equation, S is the scatter matrix of the monomials as introduced before, S_N and S_T are the scatter matrices of the directional derivatives of monomials in directions perpendicular and tangent to the data set, respectively, and G_N is the sum of the gradients of the monomials in the normal direction. This minimization is a linear least squares problem and the solution is then formally derived as:

$$A = \mu(\underbrace{S + \mu(S_N + S_T)}_{\mathcal{S}})^{-1} G_N \tag{7}$$

Let $\mathcal{S} = S + \mu(S_N + S_T)$, a $p \times p$ matrix. A and G_N are vectors with p components.

We named this algorithm gradient-one fitting. Like Hermite interpolation [3], *gradient-one fitting is Euclidean invariant*, see [4] and Sec. 6.3, respectively, *but not affine invariant.* Gradient-one fitting is also *scale invariant* since the data standardization step sets some Euclidean invariant measure of the size of the shape to 1 before fitting. We use the scattering radius of the data points as the shape size measure; this measure is Euclidean invariant. Data set standardization introduces a numerical advantage by improving the condition number of the scatter matrix $\mathcal{S} = S + \mu(S_N + S_T)$ of the problem (7). The condition number gives an idea of the numerical stability of linear algorithms such as the computation of the inverse of a matrix [5]. Data standardization improves the stability of the fits; however, if the standardization step has to be omitted, in order to have scale invariance it is necessary to modify (7) to $A = \mu(S + s^2\mu(S_N + S_T))^{-1}G_N$, where s is the shape size measure.

The necessity to introduce information about the first derivatives was first pointed out in [10] and handled in a linear way with the so-called 3-levels (3L) fitting algorithm. The idea of the 3L fitting is to constrain the polynomial to fit not only the data set but also two level sets of the distance transform of this data set, thus preventing the presence of singularities of $f(x, y)$ in the vicinity of the data to be fitted. Therefore, indirectly, 3L fitting puts constraints on the gradient of the fitted IP. In fact, it can be proved that the gradient-one algorithm is similar to the 3L fitting algorithm expanded to the first order with respect to the inter-level distance parameter.

In comparison to the classical least-squares fits (see Fig. 6(a) and (d)), results obtained on the same data sets are much better as shown in Fig. 6(b) and (e). Especially, obtained fits are locally consistent with the continuity of the data set. To gain further insight into how local consistency is achieved by controlling the gradient across the data set, we examine Fig. 3. Fig. 3(d) shows that the gradient direction along the zero set obtained by gradient-one fitting consistently points into the shape whereas in Fig.3(b) it can be seen that this direction switches between inwards and outwards. The zero set from solution of (3) is broken into pieces as can be seen in Fig. 3(a) whereas in Fig. 3(c) the zeroset is a smooth representation of the data curve. Also notice that in the vicinity of the data, the surface in Fig. 3(b) is flatter than is the surface in Fig. 3(d) which means that with small perturbations of the data, classical fitting is prone to much larger changes in the zero set. In addition to better stability of the zero set and better shape representation power, gradient-one fitting also provides better interpolation properties which allow IPs to be robust to a certain amount of missing data along the curve. The stability of the zero set achieved by the gradient-one fitting algorithm is an important improvement over classical fitting techniques. It can be seen in Fig. 2(f) that the zero sets of the resulting fits are stable under local data perturbations. Even though, the perturbations in Fig. 2 (e) are much larger than those in Fig. 2(a)-(d), the changes in the fitted IPs are much smaller in Fig. 2(f).

Parameter μ has important effects on the properties of the fits. It controls the relative weight of the gradient constraint with respect to the algebraic distance constraint. The effect of the gradient constraint on the zero set of the fit is a smoothing of the high curvature areas. Fig. 4 is an example of smoothing of the zero set when μ is increasing. In all our experiments, μ is fixed to $\frac{1}{7}$ which gives satisfying results as shown in Fig. 6(b) and (e). This value is a good trade-off between the accuracy of the representation and the stability of the fitted parameters. *However, better stability of the estimated polynomial coefficients can be achieved with equal weights on the gradient and data fit constraints as shown in Fig. 4 because the resulting fits will be "closer" to the set of well behaved polynomials, the powers of the unit circle.* In a more general framework, μ can be made a user-specified function along the length of the curve providing more control for interactive curve representation purposes. It should also be pointed out that information about the higher order derivatives such as curvature can be incorporated into gradient-one fitting to provide additional constraints.

6 Ridge Regression Fitting

6.1 Unstable Subspaces

Although, local stability of the zero set around the data is excellent with gradient-one fitting, there is still significant room for improvement in the stability of the coefficients of the polynomial and the global behaviour of the polynomial. Coefficient vectors in certain subspaces of the coefficient space may produce very similar zero sets around the data set. As an example, assume that the data is a set of aligned points along $x - y = 0$, and that we are trying to fit a full conic. If we do the fit many times subject to small perturbations of the data, we can observe that the resulting coefficient vectors span a 3 dimensional subspace containing the solutions $x(x-y)$ and $y(x-y)$ as well as $x-y$. This is a consequence of the fact that each of these three solutions and all of their linear combinations fit the original data set equally well. The global instability of polynomials is also evident in the extra pieces of the zero set that lie away from the data, see Fig. 6 (d) and (e). Indeed, these pieces are extremely sensitive to small perturbations in the data even though the zero set around the data is stable.

We now examine global instability problems. $\mathcal{S}$, defined in (7), is symmetric positive since it is a sum of scatter matrices, and thus can be written as $\mathcal{S} = U^t \Lambda U$ where U is a rotation in the coefficient space. The elements of Λ and the columns of U are the eigenvalues and eigenvectors of $\mathcal{S}$, respectively. If there is exact collinearity in the data, $\mathcal{S}$ will be singular and one or more eigenvalues will be 0. A much more common problem is near collinearity where some eigenvalues are very small compared to others and $\mathcal{S}$ is nearly singular with a very large condition number. Least Squares Estimation produces the coefficient vector A that globally minimizes the error function in (6). Eigenvectors of $\mathcal{S}$ associated with the very small eigenvalues do not contribute to the polynomial significantly around the dataset; thus such vectors multiplied with large scalars get added into the solution in pursuit of slightly better solutions. This results in very large variances for coefficients in the subspaces spanned by these eigenvectors. In Fig. 5 the graph of a goodness of fit function in two variables is shown. Notice that the function drops off steeply with the stable variable V, but changes only very slowly with unstable W. Thus, the solution of LSE which seeks the highest point on the graph, marked LS in the Fig. 5, moves along the unstable *ridge* (heavy line in Fig. 5) with the addition of small amounts of noise to the data. Consequently, the variance of the variable W due to noise is much larger than that of V. What we desire is that scalars multiplying such eigenvectors be pushed to zero rather than up to unstabily-cancelling infinities. This requires modifying LSE as we explain next.

6.2 Ridge Regression (RR)

As stated in Section 6.1, we would like variables that do not contribute significantly to the fit to be forced to attain values as close to zero as possible while other variables are effectively unchanged. Since the solution has to move along the ridge, the stabilization of the least square is known as *Ridge Regression* (RR) [6, 24]. The method of RR modifies $\mathcal{S}$ so that it is closer to what it would be for data in which there is no collinearity, that is, data in which all the explanatory variables are uncorrelated with one another. The modified coefficient vector, A_{rr} is obtained by

$$A_{rr} = \mu(\mathcal{S} + kD)^{-1}G_N \tag{8}$$

where D is a positive definite and symmetric matrix and k is the RR parameter. Although D could in principle be chosen as any positive definite matrix, in this paper we restrict ourselves to the simple case where D is a diagonal matrix. The addition of a diagonal matrix D to the scatter matrix $\mathcal{S}$ has the effect of adding a bias which produces coefficient vectors with smaller norms (smaller $\| A_{rr} \|$). In this sense, RR is analogous to weight decay regularization used in training Neural Networks. Elements of D are functions of the sum of squared values of the monomials (in other words, D is a function of the main diagonal of $\mathcal{S}$). A specific choice for the elements of D that meets the rotational invariance requirements and which has a desired limiting behavior is proposed and explained in further detail in Section 6.3. Notice that as k is increased, $\mathcal{S} + kD$ approaches D, and A approaches the limit $A_{limit} = \frac{\mu}{k}D^{-1}G_N$. We examine the limiting behavior of G_N in Section 6.4.

When there is collinearity, (8) biases the solution closer to G_N. For the example given in the beginning of this section, $G_N = [\,0 \quad n \quad -n \quad 2\bar{x_j} \quad \bar{y_j} - \bar{x_j} \quad -2\bar{y_j}\,]^t$. Thus, if the data set is centered at the origin, the solution obtained by RR is biased toward $[\,0 \quad 1 \quad -1 \quad 0 \quad 0 \quad 0]^t$, i.e. the equation of the line $x - y = 0$ we are searching for. It can easily be shown [24] that

$$A_{rr} = U \Delta U^t A \tag{9}$$

Δ is a diagonal matrix of shrinkage factors and U is as defined in Sec. 6.1. In other words, RR modifies the Least Squares Estimate by first rotating it to obtain uncorrelated components, shrinking each component by some amount and finally restoring the original coordinate system by another rotation. The crucial point is the amount of shrinkage applied to each component. If D in (8) were chosen to be the identity matrix, then it is shown in [24] that

$$\Delta = diag(\delta_i) \ , \ \delta_i = \frac{\lambda_i}{\lambda_i + k} \tag{10}$$

where k is the RR parameter and λ_i are the eigenvalues of $\mathcal{S}$, i.e., the diagonal components of Λ. The shrinkage factor δ_i multiplies the *i'th* eigenvalue of $\mathcal{S}^{-1}$ which is λ_i^{-1}, thus the *i'th* eigenvector is shrunk by a factor of δ_i in the solution. Since the eigenvectors related to the very small eigenvalues of $\mathcal{S}$ are unstable, we would like to shrink them while leaving other eigenvectors largely unaffected. With (8), this is accomplished as shown by (10). Consider a simple case similar to the one depicted in Fig. 5 where there are two variables one of which is significantly less stable than the other. This would result in an ill-conditioned matrix $\mathcal{S}$ with eigenvalues,e.g., $\lambda_1 = 1$ and $\lambda_2 = 10^{-4}$, and $\mathcal{S}^{-1}$ having eigenvalues 1 and 10^4 which are the reciprocals of λ_1 and λ_2, respectively. If we select $k = 10^{-3}$ we obtain the shrinkage factors $\delta_1 = 0.999$ and $\delta_2 = 0.0909$. Thus, the eigenvalues of $(\mathcal{S} + kI)^{-1}$ will be $1 \times 0.999 = 0.999$ and $10^4 \times 0.0909 = 909$. Notice that the stable eigenvector corresponding to the larger eigenvalue of $\mathcal{S}$ (equivalently the smaller of $\mathcal{S}^{-1}$) remains relatively unchanged whereas the condition number is improved from $\frac{10^4}{1} = 10^4$ to $\frac{909}{0.999} \approx 909$, an approximately 11 fold improvement. We address the question of choosing the value of k in Section 6.5.

Fig. 6(c) and (f) shows fits of degrees 6 and 8 obtained by RR. Comparing these results with the results from standard gradient-one fitting shown in Fig. 6(b) and (e), we observe two important properties of RR: (i) the extra pieces of the zero set in the fit to the pliers shape is gone and both fits are bounded, and (ii) the smoothing introduced around the data set is negligible. These properties follow from the fact that stable dimensions are left largely unaffected by RR while unstable ones are shrunk to insignificant values.

The effect of increasing the parameter k from 0 to higher values is shown in Fig. 7. Notice that the unbounded pieces that are close to the data in fitting with no RR, $k = 0$, start to move away with increasing k. Actually, these pieces totally disappear and the polynomial zero set becomes bounded. Recall that in Section 6.1 it was pointed out that unboundedness and extra pieces of the zero set were symptoms of the instability in fitting. Thus RR achieves the goal of getting rid of these effects, a qualitative improvement in fitting. In Section 7 we present results of experiments that show the quantitative improvement in stability obtained by RR which we believe is strongly linked to the qualitative improvements summarized above. We also prove in Section 6.4 that a fit to data for a closed shape is guaranteed to converge to a bounded IP curve as k goes to infinity.

6.3 Rotational Invariance of RR

The question of the invariance of the fitting algorithm to Euclidean transformations of the data is important to insure repeatability of the results. In this section, we show that the Gradient-one fitting is rotation and translation invariant and that the matrix D must be of a special form to keep the rotational invariance property in RR.

When a Euclidean transformation is applied to the data set, vector Y of monomials is transformed as $Y' = V(\theta, t_x, t_y)Y$, where the $p \times p$ matrix V is a function of only θ, the applied rotation angle, and (t_x, t_y), the applied translation. The zero set of the polynomial is defined by $A^t Y = 0$. After substitution $Y = V^{-1}Y'$, the transformed coefficients are $A' = (V^t)^{-1}A$. A' resulting from fitting to Euclidean transformed data is $A' = (S' + \mu(S'_N + S'_T))^{-1}G'_N$, from (7). We now show this is exactly $(V^t)^{-1}A$. Substituting for A from (7)

$$(V^t)^{-1}A = (V^t)^{-1}\left(S + \mu\left(S_N + S_T\right)\right)^{-1}V^{-1}VG_N = \left(VSV^t + \mu\left(VS_NV^t + VS_TV^t\right)\right)^{-1}VG_N$$

But from Section 3, S is YY^t, and thus transforms as $S' = VSV^t$. Similarly, the matrices containing the information on the normals and tangents transform as: $S'_T = VS_TV^t$, $S'_N = VS_NV^t$ using the fact that normals are Euclidean covariant. G_N transforms like Y, thus $G'_N = VG_N$. This leads to $A' = (S' + \mu(S'_N + S'_T))^{-1}G'_N$ as was to be shown. Consequently, Gradient-one fitting is Euclidean Invariant. Notice that the Euclidean properties of $V(\theta, t_x, t_y)$ is used only for the computation of the normal components. This leads to a possible extension to affine invariant fitting if a method to robustly compute affine invariant normals is developed.

If we apply the same substitutions to (8), we obtain $[V^{-1}(S' + \mu(S'_N + S'_T))(V^t)^{-1} + kD]V^tA'_{rr} = V^{-1}G'_N$. RR fitting will be Euclidean invariant if $A' = (S' + \mu(S'_N + S'_T) + kD')^{-1}G'_N$ which is exactly (8) in the transformed reference system. From the equations in the preceding paragraph, we see that this requirement is satisfied if $VDV^t = D$. This means that the invariance of the algorithm to Euclidean transformations dictates the structure of the matrix D. It is known [21] that, if the Euclidean transformation is reduced to a pure rotation, V can be decomposed as $V = B^{-\frac{1}{2}}RB^{\frac{1}{2}}$ where B is the diagonal matrix of binomial coefficients:

$$B_{vv} = \frac{(i+j)}{!i!j!}\ ,\ v = j + \frac{(i+j+1)(i+j)}{2} \tag{11}$$

and R is a block diagonal rotation matrix. Rotation block R_k is associated with the k^{th} form, for each k. See [21] for details. Upon substitution of V in $VDV^t = D$, we have $B^{-1/2}RB^{1/2}DB^{1/2}R^tB^{-1/2} = D$

which simplifies to $RBDR^t = BD$ under the asumption that D is diagonal. It is sufficient for satisfying the previous equation that D is block by block, the inverse of B. Therefore, a D sufficient for rotation invariance is:

$$D_{vv} = \alpha_{i+j} \frac{i!j!}{(i+j)!} \ , \ v = j + \frac{(i+j+1)(i+j)}{2} \tag{12}$$

where α_{i+j} is a free parameter for the $i+j$*'th* block. In problems where invariance is not of concern, Principal Component Methods [7] which do not provide any freedom in the choice of D, can be used alternatively. Since invariance is a major concern for us, we choose to work in the more general framework of RR.

There are $n+1$ parameters, corresponding to the $n+1$ blocks and forms. We are free to set these parameters in a Euclidean invariant way. The simplest approach would be to set all to 1. Using the binomial coefficients once more, we set each of these parameters to the invariantly weighted sum of the diagonal elements of $\mathcal{S}$ associated with the $i+j$'th form. In other words,

$$\alpha_{i+j} = \sum_{r,l \geq 0; r+l=i+j} \frac{(r+l)!}{r!l!} \sum_{q=1}^{m} x_q^{2r} y_q^{2l}$$

the weighted total scattering of the terms in *i+j'th* degree form. This choice of α_{i+j} is Euclidean invariant. The motivation for this choice comes from the fact that RR is equivalent to adding independent random noise on the matrix of monomials. When we compute $S = MM^t$, the expected change on the off diagonal terms are 0 because of the independence of the noise added to each monomial. However, the variances, of the noises added to the monomials, add onto the main diagonal of S exactly as in RR. So our choice of α_{i+j} is equivalent to adding independent noise to each monomial with variance proportional to a Euclidean invariant function of the scattering of all the monomials in its form. This is very closely related to *weight decay* regularization used to overcome problems of *overfitting* in iterative optimization schemes [2]. We have found that this choice brings significant improvements in power of shape representation over simply setting $\alpha_{i+j} = 1$ for all i, j.

6.4 Boundedness Properties and Limiting Behavior of RR

The limit of the solution of (8) as k goes to infinity is $A_{limit} = D^{-1}G_N$, up to a scale factor. It turns out that the polynomial specified by A_{limit} has important properties. Indeed when the data shape is closed and the degree of the fitted polynomial is even, the IP curve converges to the curve given by A_{limit} which is always bounded, as shown in Fig. 8. The proof that follows is based on the divergence theorem for closed 2D curves. To begin, using (12), components a_{ij} of vector A_{limit} can be approximated as an integral along a contour, C, when the data shape is closed and the sampling of the curve is not too coarse:

$$a_{ij} = \frac{(i+j)!}{\alpha_{i+j} i! j!} \oint_C N^t \nabla(x^i y^j)$$

Therefore, by applying the divergence theorem and using the vector identity $\nabla \cdot \nabla g = \nabla^2 g$ it becomes:

$$a_{ij} = \frac{(i+j)!}{\alpha_{i+j} i! j!} \int\int \nabla^2(x^i y^j) dxdy = \frac{(i+j)!}{\alpha_{i+j} i! j!} \int\int i(i-1)x^{i-2}y^j + j(j-1)x^i y^{j-2} dxdy$$

where $\nabla^2 g$ is the Laplacian of function g and the double integral applies in the data shape's interior.

Using (1), and introducing the monomial vector $Y' = (x'^i y'^j)$, the zero set of A_{limit} is

$$A^t_{limit} Y' = \sum_{0 \le i+j \le n} a_{ij} x'^i y'^j = 0$$

To prove that the zero set of this polynomial is always bounded, it is enough to show that the leading form of this polynomial is always strictly positive. By using the two previous equations we find:

$$\sum_{i>1, j>1, i+j=n} a_{ij} x'^i y'^j = \frac{1}{\alpha_n} [\quad \int\int x'^2 \sum_{i>1, j>1, i+j=n} \tfrac{(i+j)!}{(i-2)! j!} (xx')^{i-2} (yy')^j dxdy \; + \quad \int\int y'^2 \sum_{i>1, j>1, i+j=n} \tfrac{(i+j)!}{i!(j-2)!} (xx')^i (yy')^{j-2} dxdy \quad]$$

Then, we derive that:

$$\sum_{i+j=n} a_{ij} x'^{i} y'^{j} = \frac{n(n-1)}{\alpha_n}(x'^2 + y'^2) \int \int (xx' + yy')^{n-2} dxdy$$

is always positive for even degrees. As an important consequence of this proof, it is always possible to find some $k > 0$ such that the fitted polynomial (to a closed shape) has bounded level sets, as desired.

6.5 Choosing the RR Parameter

The *bias* of an estimator is the distance between the true value of the parameter being estimated, A_{true}, and the expected value of the estimator, $\overline{A_{rr}}$. The *variance* of an estimator is its expected square deviation from its expected value, $\overline{\| A_{rr} - \overline{A_{rr}} \|^2}$. k controls the *bias-variance* tradeoff. Usually, the variance is significantly reduced by deliberately introducing a small amount of bias so that the net effect is a reduction in *total mean squared error* which is defined as $bias^2 + variance$. Introducing bias is equivalent to restricting the range of functions for which a model can account. Typically this is achieved by removing degrees of freedom. Contrary to other approaches such as *Principal Component Methods* [15, 7], RR does not explicitly remove degrees of freedom but instead smoothly reduces the variability of parameters. This makes the model less sensitive to small perturbations. Selection of the parameter k in practice can be done in one of two ways depending on what the resulting fit will be used for:

Choosing k for Shape Modeling. Here the main goal of fitting is to obtain a good representation of the shape without too much smoothing, with bounded zero sets and without extraneous pieces in the zero set. In Fig. 8, it can be seen that increasing k results in first smoothing high curvature parts of the shape and then convergence to a bounded shape that does not visually represent the data. So the aim here is to choose the smallest possible value of k that gets rid of unstable artifacts like unboundedness, see Fig. 9 for examples where k was chosen in this manner. This can be done iteratively since fitting for modeling can usually be done off-line. Parameter k can be increased from 0 to larger values until significant amounts of error start to be introduced into the fit. Polynomial Interpolated Measure (PIM) [11] can be used to track this error as a difference in the polynomial at $k = 0$ and at the value of k under consideration.

Choosing k for Recognition. Here the main goal is to minimize the total mean squared error of estimator A_{rr}. Such an optimal value of k is empirically shown to exist and is found in Sec. 7. Choosing the optimal value of k analytically remains to be done in our future work. Optimal values of k could differ for different data sets. In [20], it is shown that k can be computed from a data independent threshold τ, on the condition number of $\mathcal{S} + kD$. The optimal value of τ will be data independent.

7 Experiments

7.1 Perturbation Models

Before we present experimental results, it is important to clarify how the perturbed data sets in these experiments were generated. Most researchers in the field of computer vision use random white noise (the noise added to each point in the data is independent of others) in their experiments on shape recognition, and thus most algorithms are optimized to handle this type of noise. White noise when used with very small standard deviations is good for simulating quantization errors; however, it is not a good model for generating deformed copies of a shape as might be sketched by a human or as might appear after segmentation from an image of an object taken under slightly different viewing conditions. We would like to be able to model these variations of shape since our motivation is to use IP fitting for indexing into image databases by query by sketch and query by example. Figs. 10(a) and (b) show the silhouette of a fish and white noise with standard deviations 0.05 and 0.1, respectively.

It is clear that these shapes cannot represent the shape variations we desire. The solution we propose is simply to use colored noise instead of white noise. First generate a white noise sequence equal in length to the number of data points. Then convolve this sequence with an averaging window of length 0.15 times the number of data points. This sequence is added in the direction perpendicular to the data at each point. Figs. 10(c) and (d) were obtained with this method. Comparing these with Figs. 10(a) and (b), it appears that colored noise models represent meaningful shape distortions whereas white noise can only represent quantization errors. The connection between distortions in shapes sketched by humans and appropriate colored noise models will be investigated in future work. Also, the arbitrary choice of setting the length of the averaging window to be 0.15 times the length of the data sequence can be changed to obtain different effects in the distortion produced. Another type of perturbation used in our experiments is missing data where a random point on the given shape is picked and a number of consecutive points are removed. Removing intervals introduces much stronger perturbations then removing an equal number of randomly spaced points.

7.2 Object Recognition Experiments

Various object recognition experiments were performed to verify that RR improves object recognition performance. A set of 27 objects, Fig. 11, including real world objects and artificial free-form shapes ranging from simple to complex, was used for all of the experiments outlined in this section. It is important to note that some objects have very similar shapes such as the fighter aircrafts, eels, and fishes. This makes object recognition for this set of objects a non-trivial task.

Recognition performance was tested under various perturbation models which are combinations of colored noise, missing data and rotation as explained in Sec. 7.1. Given a perturbation model, 1000 samples (perturbed shapes) are generated from each base shape. Each sample is fit with an IP using the methods outlined in the previous sections, thus producing a sample in coefficient-vector space for each perturbed shape. Then, a recently developed complete set of invariants [18] is computed for each coefficient-vector sample. One of the most important advantages for recognition of this specific set of invariants is that each invariant is either a linear or quadratic function of the coefficients or an angle determined by a pair of components of the coefficient-vector. This leads us to believe that they should out-perform highly non-linear algebraic invariants in robustness. Finally, a mean and full covariance matrix in the invariant space is learned for each object. Test sets (100 samples of each object) are generated in the same manner independently of the training set.

Average recognition rates for the 27 objects are plotted against the logarithm of the RR parameter k in Fig. 12. Recognition rates obtained without using RR are shown with the horizontal lines. In Fig. 12(a) $4th$ degree polynomials were used with a perturbation model of 10% colored noise and random rotations combined. Optimal choice of the RR parameter provides approximately 3% increase over the already high rate of 96.5%. Note that there is an optimal value of k, this is expected since k controls the *bias-variance* tradeoff in invariant space and some value of k has to minimize $bias^2 + variance$. The following experiments verify this fact with the further important implication that for this set of objects, best recognition performance is obtained using approximately $k = 10^{-3}$ regardless of the degree of the polynomial or the perturbation model being used. One question to be investigated is if this optimal value of k will generalize to larger sets of objects.

The experiments presented in Fig. 12(b) use a stronger perturbation model combining 10% colored noise, 10% missing data and random rotations. Both $4th$ and $6th$ degree polynomials were tested. For degree 4, optimal choice of k provides 7% improvement in recognition achieving approximately 97%. For degree 6, a much more substantial 16% improvement is obtained raising the best recognition performance to approximately 99%. These top rates are impressive when one looks at some typical perturbed samples generated in this experiment, Fig 13. Note that random rotations are omitted in Fig 13 for easy comparison with the original shape. Using $6th$ degree IPs provides only a 2% advantage in recognition over using using $4th$ degree; moreover for some non-optimal values of k and with no RR it actually does worse. There are two important deductions here: 1. Since $6th$ degree IPs have more coefficients (degrees of freedom) they are more prone to problems of unstable subspaces then $4th$ degree IPs, especially for simpler shapes that might not require a $6th$ degree polynomial. Since this is exactly the problem RR sets out to solve, the observation made above is totally expected. 2. It might seem tempting to restrict object recognition to the use of $4th$ degree IPs; however, as will be made clear in the next example there are much more substantial gains to be made with the use of higher degrees in some cases. We now use even a stronger model of perturbation, by keeping the 10% colored noise and rotation and doubling the amount of missing data to 20%. Robustness to missing data crucially depends on a good representation. Fig. 12(c) confirms this statement; $4th$ degree IPs yield a top recognition rate of approxiamtely 88%, $6th$ degree IPs are able to improve this rate to approximately 94%. Having established that using high degree IPs are necessary in certain problems, it is also very important to once more realize the crucial role played by RR in the success of high degree IPs; using the optimal value of k provided a gain of over 35% compared to no RR, with $6th$ degree IPs in this example.

8 Conclusions

In the continuing quest for achieving maximum stability in the representation of curve data by algebraic curves (i.e., the zero sets of polynomials in x and y) and in the stability of the polynomial coefficients, this paper makes two important contributions. The first is an understanding of the role of data normalization and polynomial gradient-constraint in improving representation and coefficient stability. This also sheds light on why the 3L fitting algorithm [10] is so much more stable than previous fitting algorithms. The second contribution is the use of rotation-invariant RR, in the fitting, for improving the stability of both the representation and the coefficients even further. RR drives those portions of the polynomial zero-set, that are not appropriate to the curve data, far from the data. It also shrinks to near-zero those polynomial coefficients not important for representing the curve data. The remaining coefficients are stable and result in increased stability when used for pose-invariant object recognition or object pose estimation.

References

[1] F. Acton. *Numerical Methods That Work.* Harper and Row, 1970.

[2] C. M. Bishop. *Neural Networks for Patter Recognition.* Clarendon Press, 1995.

[3] P. Davis. *Interpolation and Approximation.* Blaisdell Publishing Company, 1963.

[4] K. Deepak and J. Mundy. *Geometric Invariance in Machine Vision*, chapter Fitting Affine Invariant Conics to Curves. MIT Press, 1992.

[5] G. Golub and C. V. Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore and London, 1996. Third edition.

[6] A. E. Hoerl and R. W. Kennard. Ridge regression: Biased estimation of nonorthogonal problems. *Technometrics*, 12:55–67, Feb. 1970.

[7] I. T. Jolliffe. *Principal Component Analysis.* Springer series in statistics. Springer-Verlag, 1986.

[8] D. Keren, D. B. Cooper, and J. Subrahmonia. Describing complicated objects by implicit polynomials. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(1):38–53, Jan. 1994.

[9] D. Keren and C. Gotsman. Fitting curves and surfaces with constrained implicit polynomials. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(1):31–41, Jan. 1999.

[10] Z. Lei, M. M. Blane, and D. B. Cooper. 3L fitting of higher degree implicit polynomials. In *Proceedings of Third IEEE Workshop on Applications of Computer Vision*, pages 148–153, Sarasota, Florida, USA, Dec. 1996.

[11] Z. Lei, T. Tasdizen, and D. Cooper. Pims and invariant parts for shape recognition. In *Proceedings of Sixth International Conference on Computer Vision (ICCV'98)*, pages 827–832, Mumbai, India, 1998.

[12] R. Malladi, J. A. Sethian, and B. C. Vemuri. Shape modeling with front propagation: A level set approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):158–175, 1995.

[13] M. Pilu, A. Fitzgibbon, and R.Fisher. Ellipse-specific direct least-square fitting. In *Proceedings of IEEE International Conference on Image Processing*, Lausanne, Switzerland, September 1996.

[14] O. Smedby and G. Borgefors. Shape description and segmentation in 2 and 3 dimensions by polynomial expansion. In C. Arcelli, L. P. Cordella, and G. S. di Baja, editors, *Advances in Visual Form Analysis*, pages 559–568. World Scientific, 1997.

[15] J. Subrahmonia, D. B. Cooper, and D. Keren. Practical reliable Bayesian recognition of 2D and 3D objects using implicit polynomials and algebraic invariants. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(5):505–519, May 1996.

[16] S. Sullivan, L. Sandford, and J. Ponce. Using geometric distance fits for 3-D object modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(12):1183–1196, Dec. 1994.

[17] J.-P. Tarel, H. Civi, and D. B. Cooper. Pose estimation of free-form 3D objects without point matching using algebraic surface models. In *Proceedings of IEEE Workshop on Model-Based 3D Image Analysis*, pages 13–21, Mumbai, India, 1998.

[18] J.-P. Tarel and D. Cooper. A new complex basis for implicit polynomial curves and its simple exploitation for pose estimation and invariant recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 111–117, Santa Barbara, California, USA, June 1998. To appear in IEEE Transactions on Pattern Analysis and Machine Intelligence.

[19] J.-P. Tarel, W. A. Wolovich, and D. B. Cooper. Covariant conics decomposition of quartics for 2D object recognition and affine alignment. In *Proceedings of the International Conference on Image Processing*, Chicago, Illinois, USA, Oct. 1998.

[20] T. Tasdizen, J.-P. Tarel, and D. Cooper. Improving the stability of algebraic curves for applications. Technical Report 176, LEMS, Division of Engineering, Brown University, 1998.

[21] G. Taubin. Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11):1115–1138, Nov. 1991.

[22] G. Taubin, F. Cukierman, S. Sullivan, J. Ponce, and D. Kriegman. Parameterized families of polynomials for bounded algebraic curve and surface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(3):287–303, March 1994.

[23] H. Tek and B. B. Kimia. Curve evolution, wave propagation, and mathematical morphology. In *Fourth International Symposium on Mathematical Morphology (ISMM'98)*, June 1998.

[24] H. D. Vinod and A. Ullah. *Recent Advances In Regression Methods.* Statistics: Textbooks and Monographs. Marcel Dekker, Inc., 1981.

-1.00000	-4.00000	-6.99970	-20.84691	$-13.99236 \pm 2.518831i$
-2.00000	-5.00000	-8.00727	$-10.09527 \pm 0.643501i$	$-16.73074 \pm 2.812621i$
-3.00000	-6.00001	-8.91725	$-11.79363 \pm 1.652331i$	$-19.50244 \pm 1.940331i$

Table 1: *Roots of the perturbed Wilkinson Polynomial*

-9.7998e-25	2.6102e-10	-0.00030308	0.90528	-72.188	286.9980	-36.8373
9.7620e-18	-7.2448e-08	0.0071163	-5.5366	158.9022	-227.0224	5.1379
-1.9477e-13	6.9438e-06	-0.10006	23.6634	-252.6122	118.6832	

Table 2: *Estimates for root perturbations.*

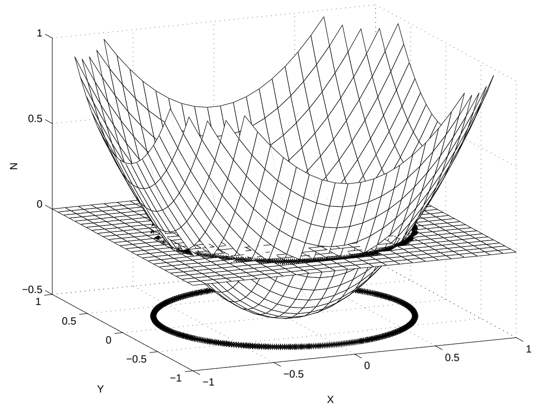


Figure 1: *A circle is the zero set of a second degree polynomial.*

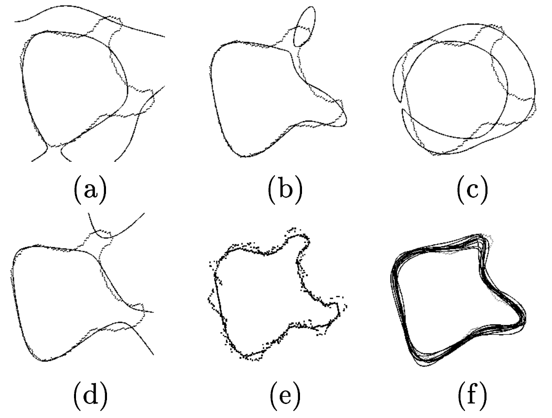
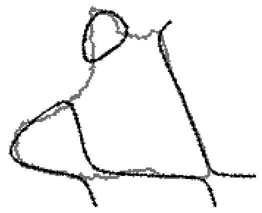
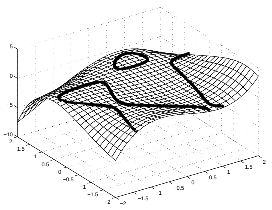
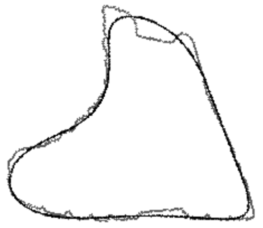


Figure 2: *(a)-(d) Classical least-squares algorithm gives unstable* 4^{th} *degree IP fits under even the smallest perturbations to the data. (e) A much more significant perturbation, (f) 10 superimposed* 4^{th} *degree polynomial fits with the gradient-one algorithm to perturbed data sets like the one in (e).*


(a)


(b)


(c)

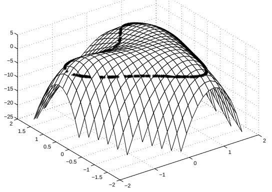
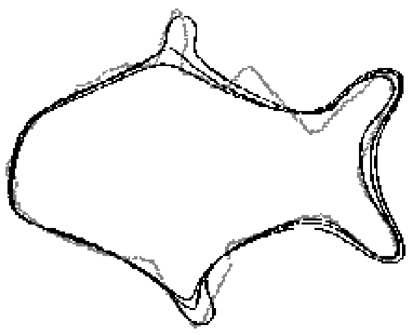
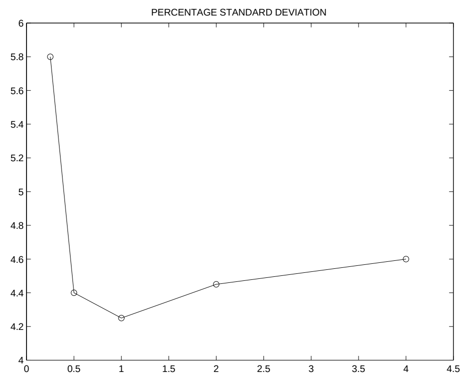

(d)

Figure 3: *Comparison of polynomial zero sets and polynomial graphs obtained by classical fitting (a)-(b) to gradient-one fitting (c)-(d).*



(a)



(b)

Figure 4: *(a)* 6th *degree IP fits with the gradient-one algorithm for three different values* μ. *(b) The average percentage standard deviation of the coefficients with respect to their average norm under colored noise (see Sec. 7.1) for increasing values of* μ.

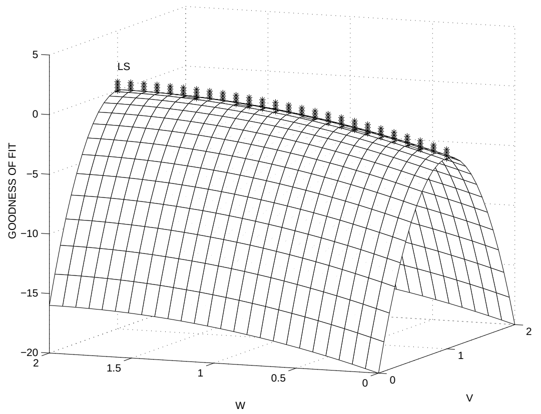


Figure 5: *Graph of an error function of two variables; here* V *is the stable variable while* W *is relatively unstable. The unstable ridge is marked by a heavier line.*

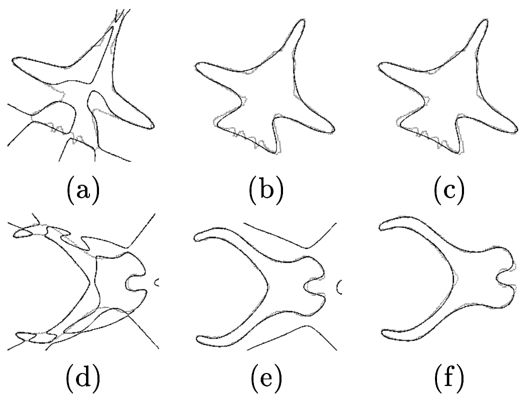


Figure 6: *(a) and (d): Classical Fitting Algorithm. (b) and (e): Gradient-one Fitting Algorithm. (c) and (f): RR Fitting Algorithm. Degree 6 and 8 are used for the airplane and pliers shapes, respectively. Notice that there are no extra components in (c) and (f).*

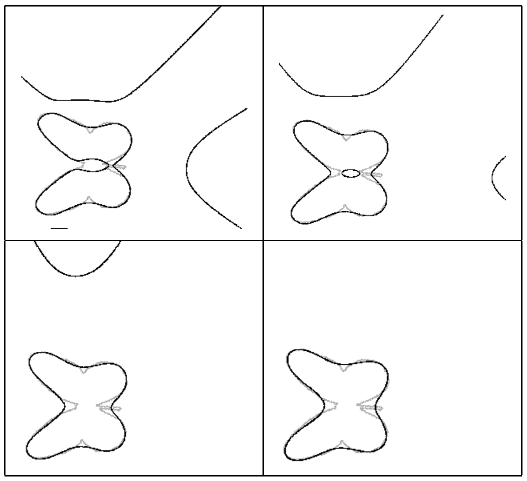


Figure 7: 6th *degree IP fits with the gradient-one algorithm and RR for increasing values of param. k.*

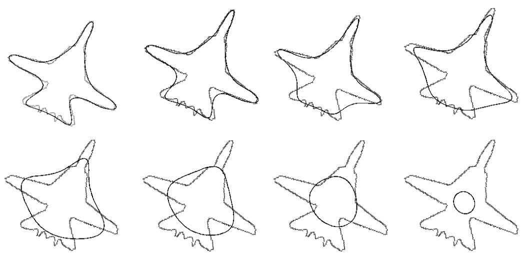


Figure 8: *Left,* 6th *degree polynomial fits with the gradient-one algorithm and RR for increasing values of parameter k ($k = 0$, $k = 0.0001$, 0.001, 0.01, 0.05, 0.5, 2.0 and 32, respectively). We can observe that the fitted zero set is becoming smoother and converging to a point.*

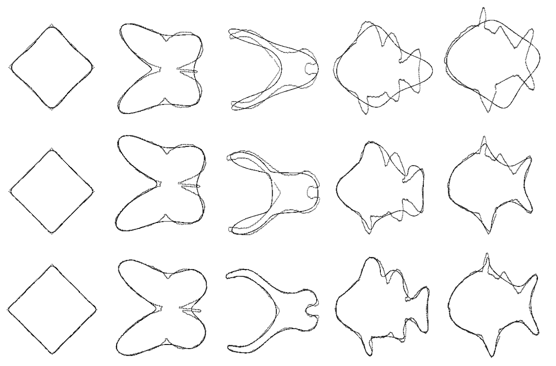


Figure 9: *Fits for* 4^{th}, 6^{th}, *and* 8^{th} *degrees with shapes of different complexities. No extra components are close to data sets. The RR parameter was chosen manually for each shape in this example.*

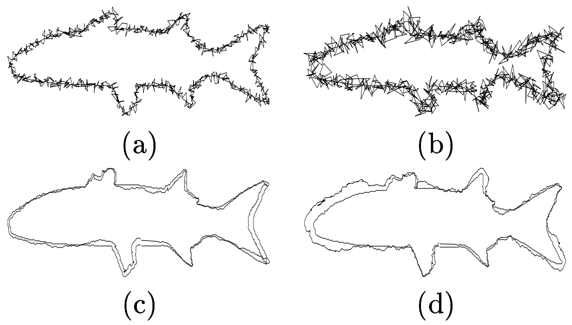


Figure 10: *Comparison of noisy data simulation using white noise (a)-(b) with standard deviations* 0.05 *and* 0.1*, respectively, and colored noise (c)-(d) with standard deviations* 0.05 *and* 0.1*, respectively.*

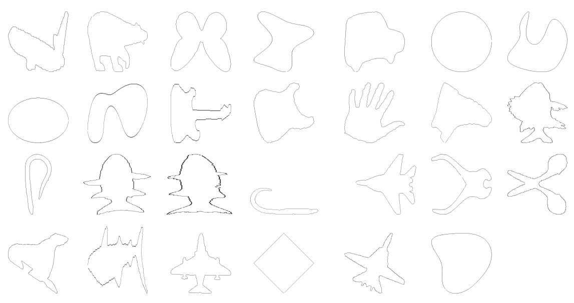
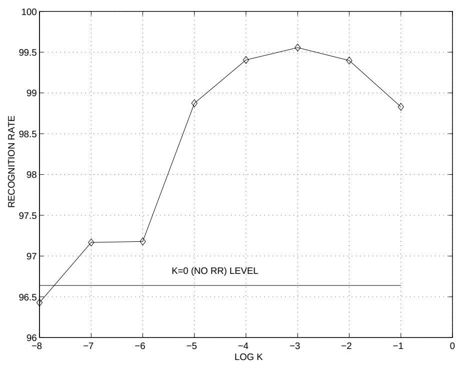
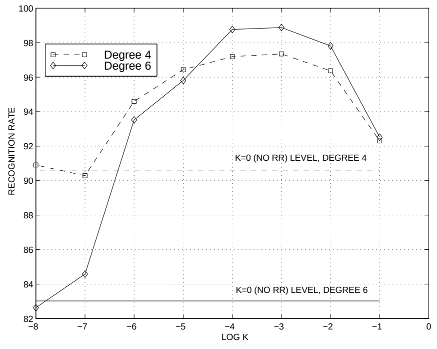


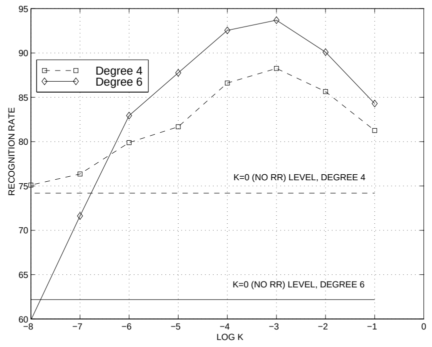
Figure 11: *Objects used in the experiments.*



(a)



(b)



(c)

Figure 12: *1000 perturbations of each object are used as the training set. Another 100 independent perturbations of each object are used as the test set. Perturbation models are (a) 10% colored noise + rotation, (b) 10% colored noise + 10% missing data + rotation and (c) 10% colored noise + 20% missing data + rotation.*

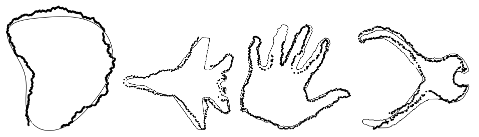


Figure 13: *A few shapes perturbed with 10% colored noise and 10% missing data.*