Stable Fitting of 2D Curves and 3D Surfaces by Implicit Polynomials

Amir Helzer, Meir Barzohar, and David Malah, *Fellow*, *IEEE*

Abstract—This work deals with fitting 2D and 3D implicit polynomials (IPs) to 2D curves and 3D surfaces, respectively. The zero-set of the polynomial is determined by the IP coefficients and describes the data. The polynomial fitting algorithms proposed in this paper aim at reducing the sensitivity of the polynomial to coefficient errors. Errors in coefficient values may be the result of numerical calculations, when solving the fitting problem or due to coefficient quantization. It is demonstrated that the effect of reducing this sensitivity also improves the fitting tightness and stability of the proposed two algorithms in fitting noisy data, as compared to existing algorithms like the well-known 3L and gradient-one algorithms. The development of the proposed algorithms is based on an analysis of the sensitivity of the zero-set to small coefficient changes and on minimizing a bound on the maximal error for one algorithm and minimizing the error variance for the second. Simulation results show that the proposed algorithms provide a significant reduction in fitting errors, particularly when fitting noisy data of complex shapes with high order polynomials, as compared to the performance obtained by the abovementioned existing algorithms.

Index Terms—Implicit polynomials, zero-set sensitivity, curve and surface fitting, stable fitting.

1 Introduction

Implicit polynomials (IP) have long been introduced for fitting 2D curves and 3D surfaces [1], [2], [3] specified by discrete data. The ability to efficiently describe complicated boundaries using the coefficients of implicit polynomials is attractive for applications in the fields of object recognition and pose estimation [4], [5], [6], [7], [11], [12], [20], coding [8], [19], boundary estimation from intensity/color images [9], and computer graphics [10]. The existence of geometric invariants [6], [7], [11] has made implicit polynomials especially appealing for the first application.

While classical *least-squares* (LS) has a simple formulation and is less complex than nonlinear methods [2], [3] for IP fitting, both greatly suffer from numerical instability, especially for high order polynomials [13], [17]. The numerical stability problem is basically the result of the extremely high sensitivity of the zero-set of the IP (that is supposed to fit the data), so that even tiny errors in the coefficients values, as would be the case in any numerical solution, may result in large fitting errors, as discussed in [17, Section IV]. This effect could also be detrimental in coding applications where the coefficients must be quantized.

To alleviate the sensitivity problem, while keeping the simplicity of the LS approach, several algorithms have recently been developed: The 3L fitting algorithm [13] introduces additional constraints, which are generated from the original data via expansion and shrinking. While providing an improvement, it still suffers from excessive sensitivity for high order polynomials needed when fitting complex shapes. The *gradient-one* fitting algorithm [17] provides further improvement by replacing the added data sets, or level sets (obtained in 3L via expansion and shrinking), by explicit differentiation and by constraining the gradient vector along the zero-set to have a fixed (unity) norm. In addition, anisotropic scaling is applied to the data, to normalize its size to unity, under a specific measure (instead of the isotropic scaling done in the 3L algorithm). The *ridge regression* (RR) fitting algorithm, also developed in [17], applies regularization to the gradient-one algorithm to alleviate the problem of spurious zero sets that typically appear in IP fitting with high order polynomials (an effect termed *global instability* in [17]).

The issue of spurious zero sets is also addressed in [8], where constraints are added to the fitting problem that provide a thin "guard strip" on each side of the object boundary in which spurious zero sets may not appear. This helps in reconstructing the object shape from the IP coefficients (of special importance in coding applications).

Other recent works on fitting curves and surfaces by IPs are presented in [14], [15], [16], [18].

In this work, we improve the performance of the gradient-one algorithm (that addresses *local stability* in fitting each data point) by constraining the norm of the gradient vector along the zero-set to the sum of absolute values of the components of the monomial vector (defined in the next section) at each data set point. That is, unlike the 3L and gradient-one algorithms, the norm of the gradient is not fixed, but is data dependent—aiming to obtain a uniform maximal deviation along the zero-set. This particular constraint is fulfilled by what we denote as the *Min-Max* fitting algorithm. It is based on using the sensitivity of the zero-set to small coefficient errors to bound the maximal fitting error. A different constraint is fulfilled by what we denote as the *Min-Var* fitting algorithm. The same zero-set sensitivity function is used to evaluate the variance of the fitting errors due to small random coefficient changes, and the algorithm constrains the norm of the gradient to the value of the root of the sum of squares of the monomial vector components, aiming to obtain a uniform variance of the error at each point of the data set.

Simulation results, which are included in the paper, show significant improvement in reducing fitting errors, not only when the coefficients are quantized, but also when fitting

• *The authors are with the Department of Electrical Engineering, Technion-Israel Institute of Technology, Haifa 32000, Israel.*
E-mail: amir_helzer@hotmail.com, meirb@visionsense.com, malah@ee.technion.ac.il.

Manuscript received 16 Dec. 2002; revised 30 Oct. 2003; accepted 8 Mar. 2004.
Recommended for acceptance by R. Basri.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 117970.

both clean and noisy data of complex shapes with high order polynomials, as compared to the gradient-one algorithm.

It should be noted that we do not address in this paper the spurious zero-set phenomenon, and the treatments in [8], that uses a "guard strip," or the ridge-regression method of [17], can be applied, in principle, to the proposed algorithms.

The layout of this paper is as follows: Section 2 provides background material on polynomial fitting and outlines the 3L and gradient-one fitting-algorithms, from which the two proposed improved algorithms evolved. In Section 3, the zero-set sensitivity function to small coefficient errors is defined and is used to bound the maximal fitting error and to evaluate its variance. These results are then used to analyze the fitting error characteristics for the 3L and gradient-one algorithms. Section 4 describes the development of the proposed fitting algorithms, including the extension of the results to weighted errors and 3D fitting. Section 5 presents simulation results obtained in comparing the proposed algorithms with the gradient-one algorithm. Section 6 summarizes and concludes the paper.

2 Background

In this section, we provide a brief overview of implicit polynomials fitting and of the 3L and gradient-one fitting algorithms.

2.1 Fitting 2D Implicit Polynomials to Curves

The objective of polynomial fitting is to describe data points (object boundary for 2D objects or surfaces for 3D objects) by the zero-set of a polynomial. That is, the value of the polynomial should be zero at the location of the data points.

The value of the polynomial at a point (x,y) can be described as the product of two vectors—a parameter vector (containing the polynomial coefficients), and a vector of monomials. For a dth order polynomial, the monomial vector is defined as:

$$\begin{aligned}&\bar{p}(x,y) = [p_1(x,y), \ldots, p_r(x,y)] = \\ &\left[x^0y^0, x^1y^0, x^0y^1, \ldots, x^dy^0, x^{d-1}y^1, \ldots, x^1y^{d-1}, x^0y^d\right],\end{aligned} \tag{1}$$

where $r = (d+1)(d+2)/2$ and the parameter vector is $\bar{a} = [a_1, a_2, \ldots, a_r]$.

The value of the polynomial described by $\bar{a}$ at location (x,y) is:

$$P_{\bar{a}}(x,y) = \bar{a}\,\bar{p}^T(x,y) \tag{2}$$

The fitting problem is therefore to find a parameter vector $\bar{a}$ that leads to a polynomial that best fits the data under a criterion to be specified. The data set is assumed to contain N points with coordinates $(x_n, y_n), n = 1, \ldots, N$. We denote the zero-set of the polynomial defined by the coefficient vector $\bar{a}$ as:

$$Z_{\bar{a}} = \{(x,y) : P_{\bar{a}}(x,y) = 0\}. \tag{3}$$

2.2 Overview of the 3L Algorithm

The 3L algorithm, developed in [13], is presented as a *linear* algorithm for fitting an implicit polynomial to a data set. The term *linear* is used by the authors to describe a problem of the form $\bar{b} = \bar{a}M$, where $\bar{b}$ is a known vector, M is a known matrix, and the vector $\bar{a}$ needs to be computed. This algorithm produces a result within one pass and no iterative computations are required. This stands in contrast to previous fitting algorithms [1], [2], which not only suffer from numerical instability, but also require an iterative solution, with unproven convergence properties.

The 3L algorithm is based on the construction of two additional data sets (level sets) that are determined from the original data set. The two additional data sets are constructed so that one set is internal and the other is external, relative to the original data set, with a distance d from it. The goal of this algorithm is to find a polynomial that has a value of zero at points belonging to the original data set and values of ε and $-\varepsilon$ at the internal and external points, respectively. To achieve this goal, the 3L algorithm uses a least-squares solution that minimizes the sum of squared errors between the required and actual polynomial values at those three data sets.

Considering the original data set points and the two added sets (internal and external) as a single set of three, N points one obtains $3N$equations in r variables (coefficients), $r < 3N$.

The goal is to minimize the total squared-error E:

$$\begin{aligned}E = &\underbrace{\sum_{n=1}^{N}\left(\bar{a}\bar{p}^T(x_n,y_n)\right)^2}_{\text{Error w.r.t original points}} + \underbrace{\sum_{n=N+1}^{2N}\left(\bar{a}\bar{p}^T(x_n,y_n)+\varepsilon\right)^2}_{\text{Error w.r.t external points}} + \\ &\underbrace{\sum_{n=2N+1}^{3N}\left(\bar{a}\bar{p}^T(x_n,y_n)-\varepsilon\right)^2}_{\text{Error w.r.t internal points}}.\end{aligned} \tag{4}$$

The constant ε is a small positive constant. The error E may be written as:

$$E = \bar{e}\,\bar{e}^T, \tag{5}$$

where the row vector $\bar{e}$ is given by,

$$\bar{e} = (\bar{a}M - \bar{b}). \tag{6}$$

The $3N$-dimensional vector $\bar{b}$ and the $r \times 3N$ Matrix M are defined by:

$$\begin{aligned}\bar{b} &= \begin{bmatrix}\bar{0} & -\bar{\varepsilon} & \bar{\varepsilon}\end{bmatrix} \\ M &= \begin{bmatrix}M_0 & M_{EX} & M_{IN}\end{bmatrix}\end{aligned} \tag{7}$$

with,

$$\begin{aligned}M_0 &= \left[\bar{p}^T(x_1,y_1) \quad \ldots \quad \bar{p}^T(x_N,y_N)\right] \\ M_{EX} &= \left[\bar{p}^T(x_{N+1},y_{N+1}) \quad \ldots \quad \bar{p}^T(x_{2N},y_{2N})\right] \\ M_{IN} &= \left[\bar{p}^T(x_{2N+1},y_{2N+1}) \quad \ldots \quad \bar{p}^T(x_{3N},y_{3N})\right].\end{aligned} \tag{8}$$

The vectors $\bar{\varepsilon}$ and $\bar{0}$, making up the vector $\bar{b}$, are constant row vectors of length N with elements ε and 0, respectively.

Assuming that $r < 3N$ (overdetermined problem), a feasible least squares (LS) solution for the problem of obtaining $\bar{a}$ that minimizes E in (5) is:

$$\bar{a}_{LS} = \bar{b}M^T\left(MM^T\right)^{-1}. \tag{9}$$

The parameter vector $\bar{a}_{LS}$ is the best parameter-vector (in the LS-error sense) of a polynomial whose zero-set approximates the location of the original data set, and whose values on both sides of the original data set (at distance "d") are approximately $\pm\varepsilon$ (positive on the inside and negative on the outside—in the above construction).

2.3 Overview of the Gradient-One Algorithm

This algorithm is presented in [17] and besides the different way the data set is standardized (normalized by anisotropic scaling [17]), it is basically a modification of the 3L algorithm by which the operation of shrinking and expansion of the original data set is replaced by explicit differentiation. Such a modification was also considered in [8], [18]. That is, the norm of the polynomial gradient vector along the zero-set is approximated in the 3L algorithm by the slope ε/d. Based on considerations relating to the sensitivity of the zero-set to small changes in the coefficient values, this norm is constrained in [17] to unity. Yet, another difference is that the authors in [17] propose to minimize a modified version of the squared error function, by adding a weight factor μ to the gradient dependent terms (equivalent to weighting by μ the two right terms in (4), in the case of 3L).

Based on the above discussion, and since practical values of μ are close to 1 (see, for example, Fig. 4 in [17]), we denote the gradient-one algorithm for the particular value of $\mu = 1$ as the *modified 3L* algorithm and we present below this algorithm by casting the fitting error in the form of (6). This will simplify and facilitate the exposition of our proposed algorithms in the sequel.

We have mentioned earlier that the gradient-one algorithm applies first anisotropic scaling to the data set in order to standardize (or normalize) it. This is basically a preprocessing operation that we do not consider here to be part of either the modified 3L algorithm or any other algorithm we discuss below. In other words, we assume that the data set that we are dealing with has been standardized in some way and the comparison of the performance of the different algorithms assumes the same data normalization.

2.3.1 The Modified 3L Algorithm

Since the zero set is supposed to fit the data set and the gradient vector is perpendicular to the zero set, we need to bring the polynomial gradient at the location of each data point to the direction of the line locally perpendicular to the data set near each data set point (see Fig. 1). According to the gradient-one algorithm its norm is constrained to unity.

The components of the polynomial gradient are given by:

$$\begin{aligned} \frac{d}{dx} P_{\bar{a}}(x,y)\big|_{(x=x_n, y=y_n)} &= \bar{a}\ \bar{p}_X^T(x_n, y_n) \\ \frac{d}{dy} P_{\bar{a}}(x,y)\big|_{(x=x_n, y=y_n)} &= \bar{a}\ \bar{p}_Y^T(x_n, y_n), \end{aligned} \tag{10}$$

where the vectors $\bar{p}_X(x_n, y_n), \bar{p}_Y(x_n, y_n)$ denote the derivatives with respect to x and y, respectively, of the monomial vector, defined in (1), at data point (x_n, y_n).

In Fig. 1, the angle relative to the x axis of the perpendicular vector to the data set at (x_n, y_n) is denoted by α_n. It is also seen that, in order to calculate α_n, we fit a regression line to the points in the neighborhood of the data point (x_n, y_n) under consideration.

Note that using a line that fits several points about each data point, helps in combating noise or small perturbations in the data. It also alleviates effects that may be caused by singularities in the data, like abrupt slope changes or relatively large gaps in the data.

The vectors $\bar{v} = [v_1, \ldots, v_N]$, $\bar{w} = [w_1, \ldots, w_N]$ (where N is the number of points in the data set) contain the elements of the perpendicular vectors for each of the data set points. Each pair of elements (v_n, w_n) in $\bar{v}, \bar{w}$, is a unit vector pointing in the direction that is locally perpendicular to the data set at each point n (i.e., to the straight line approximation).

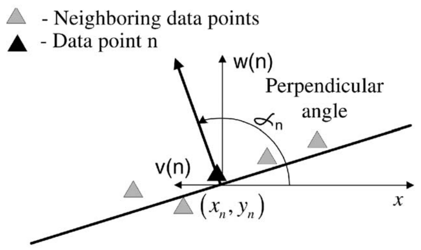


Fig. 1. Fitting a regression line to 5 points neighboring the point (x_n, y_n) and the normal vector to the line at that point.

Having estimated α_n from the data set, we can calculate the components of the vectors $\bar{v}, \bar{w}$, at each data point from the relations:

$$\left.\begin{aligned} \tfrac{w_n}{v_n} &= tg(\alpha_n) \\ \sqrt{(v_n^2 + w_n^2)} &= 1 \end{aligned}\right\} \Rightarrow \begin{aligned} w_n &= \sin(\alpha_n) \\ v_n &= \cos(\alpha_n). \end{aligned} \tag{11}$$

The values of $\bar{b}$ and M in (9) determine the value of the polynomial and its gradient. In order for the gradient of the polynomial to be perpendicular to the data set and have a unity norm, and to keep the requirement that the value of the polynomial at the location of the data points be equal to zero, $\bar{b}$ and M become:

$$\begin{aligned} \bar{b} &= [\,\bar{0} \quad \bar{v} \quad \bar{w}\,] \\ M &= [\,M_0 \quad M_X \quad M_Y\,], \end{aligned} \tag{12}$$

where,

$$\begin{aligned} M_0 &= [\,\bar{p}^T(x_1, y_1) \quad \ldots \quad \bar{p}^T(x_N, y_N)\,] \\ M_X &= [\,\bar{p}_X^T(x_1, y_1) \quad \ldots \quad \bar{p}_X^T(x_N, y_N)\,] \\ M_Y &= [\,\bar{p}_Y^T(x_1, y_1) \quad \ldots \quad \bar{p}_Y^T(x_N, y_N)\,]. \end{aligned} \tag{13}$$

The LS solution in (9) can now be used with the above expressions for $\bar{b}$ and M.

3 Zero-Set Sensitivity Function

As demonstrated and discussed in [17, Section IV] (although mainly for a 1D polynomial), the stability problem in fitting a high order polynomial to data emanates from the high sensitivity of the zero-set to small changes in the values of the polynomial coefficients.

We examine now, therefore, how small changes in the coefficient values affect the location of a point on the zero-set. Since the zero-set is continuous, we cannot measure the distance between two points on it before and after a parameter change. We define therefore the change in a zero-set point ($d\bar{z}$) as the distance between an original zero-set point, $\bar{z} = (z_x, z_y)$, and the closest point on the new zero-set, obtained after the change in the parameters.

Let's define a sensitivity matrix function at a zero-set point by the following $2 \times r$ matrix:

$$S_{\bar{a}}^{\bar{z}}(x,y) = \frac{d\bar{z}(x,y)}{d\bar{a}}. \tag{14}$$

This function expresses the relation between small changes (errors) in the coefficients and small changes in the location of zero-set points. The change in the location of a zero-set

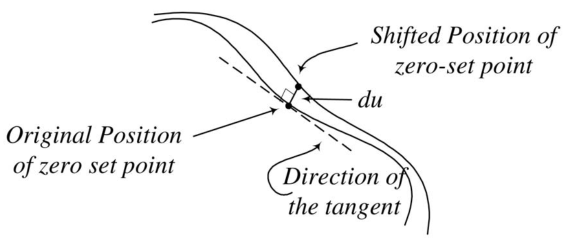

Shifted Position of zero-set point
du
Original Position of zero set point
Direction of the tangent

Fig. 2. Location of a zero-set point before and after a small change in the coefficients.

point $\bar{\varepsilon}_Z(x,y) = [\bar{\varepsilon}_X \; \bar{\varepsilon}_Y]$, resulting from a small change in the parameters $\bar{\delta}_a = [\delta_{a_1} \ldots \delta_{a_r}]$, is the product of the error components with the above sensitivity matrix:[1]

$$\bar{\varepsilon}_Z(x,y) \cong \bar{S}^{\bar{z}}_{\bar{a}}(x,y)\bar{\delta}^T_a. \tag{15}$$

3.1 Zero-Set Sensitivity Function in the Normal Direction

Small changes in the position of zero-set points along a tangent direction move zero-set points back into the zero-set. Therefore, for the purpose of evaluating zero-set changes, it is sufficient to examine changes in the direction that is perpendicular (normal) to the zero-set. We denote by $u(x,y)$ the component of $d\bar{z}(x,y)$ that is locally perpendicular to the zero-set (see Fig. 2):

$$u(x,y) = \langle \bar{z}(x,y) \cdot \nabla P_{\bar{a}}(x,y) \rangle, \tag{16}$$

where, $\langle a \cdot b \rangle$ denotes the inner-product between a and b, $\nabla f(x,y)$ denotes the gradient of f at (x,y), and $P_{\bar{a}}(x,y)$ is the value of the polynomial (determined by $\bar{a}$) at location (x,y). Therefore, the sensitivity function of interest here is given by the vector $\bar{S}^u_{\bar{a}}(x,y) = \frac{du(x,y)}{d\bar{a}}$ and denotes the relation between changes in the zero-set, in a locally perpendicular direction to the zero-set, and changes in the coefficient vector.

A key point is to observe that this function can be written as a product of two independent parts:

$$\bar{S}^u_{\bar{a}}(x,y) = \frac{du(x,y)}{dP_{\bar{a}}(x,y)} \frac{dP_{\bar{a}}(x,y)}{d\bar{a}}. \tag{17}$$

The right-hand part of the product in (17) quantifies the change in the value of the polynomial at (x,y) due to a small change in the parameter vector. This part is a vector (has an element for each of the elements in $\bar{a}$). The left-hand part quantifies the deviation of the zero-set point, in the direction perpendicular to the zero-set, due to a small change in the value of the polynomial at (x,y). This part is a scalar.

We next evaluate each part of the sensitivity function in (17) separately, beginning with the left-hand part. The deviation in the location of a zero point in a direction locally perpendicular to the zero-set, due to a change in the parameter vector, is described in Fig. 2.

For small changes in the parameter vector, the ratio between the position error in the perpendicular direction, du and the change in the value of the function at point (x,y) is the inverse of the gradient magnitude of $P_{\bar{a}}(x,y)$:

$$\frac{du}{dP_{\bar{a}}}(x,y) = \frac{1}{\|\nabla P_{\bar{a}}(x,y)\|} = \frac{1}{\sqrt{\left(\frac{\partial P_{\bar{a}}(x,y)}{\partial x}\right)^2 + \left(\frac{\partial P_{\bar{a}}(x,y)}{\partial y}\right)^2}}. \tag{18}$$

The right-hand part of the sensitivity function in (17), $\frac{dP_{\bar{a}}(x,y)}{d\bar{a}}$, can be directly calculated from (2) (i.e., using $P_{\bar{a}}(x,y) = \bar{a}\,\bar{p}^T(x,y)$, where $\bar{p}(x,y)$ is the monomial vector). The result is:

$$\frac{dP_{\bar{a}}(x,y)}{d\bar{a}} = \bar{p}(x,y). \tag{19}$$

Using (18) and (19), the sensitivity function in (17) can now be written as:

$$\bar{S}^u_{\bar{a}}(x,y) = \frac{\bar{p}(x,y)}{\|\nabla P_{\bar{a}}(x,y)\|}. \tag{20}$$

3.2 Zero-Set Fitting Error Bound and Variance

Having obtained (20), we use it to obtain a bound on the maximal fitting error and the value of the error variance, at each data point, assuming small changes in the coefficients.

The error in the direction perpendicular to the zero-set, $\varepsilon_u(x,y)$ is:

$$\varepsilon_u(x,y) = \bar{S}^u_{\bar{a}}(x,y) \cdot \bar{\delta}^T_a = \frac{\bar{p}(x,y)}{\|\nabla P_{\bar{a}}(x,y)\|}\bar{\delta}^T_a = \frac{\sum_{k=1}^{r} p_k(x,y)\delta_{a_k}}{\|\nabla P_{\bar{a}}(x,y)\|}. \tag{21}$$

For a given point (x,y) on the zero set, the maximal error is bounded by:

$$\begin{aligned} |\varepsilon_u(x,y)| &= \frac{\left|\sum_{k=1}^{r} p_k(x,y)\delta_{a_k}\right|}{\|\nabla P_{\bar{a}}(x,y)\|} \le \frac{\sum_{k=1}^{r} |p_k(x,y)| \cdot |\delta_{a_k}|}{\|\nabla P_{\bar{a}}(x,y)\|} \\ &\le \delta_{\max} \frac{\sum_{k=1}^{r} |p_k(x,y)|}{\|\nabla P_{\bar{a}}(x,y)\|}, \end{aligned} \tag{22}$$

where, $\delta_{\max} = \max\{|\delta_{a_k}|\}$.

When components of the parameter error vector are independent random variables with zero mean, the variance of $\varepsilon_u(x,y)$ can be calculated by:

$$\begin{aligned} \mathrm{var}(\varepsilon_u(x,y)) &= \frac{\mathrm{var}\left(\sum_{k=1}^{r} p_k(x,y)\delta_{a_k}\right)}{\|\nabla P_{\bar{a}}(x,y)\|^2} \\ &= \frac{\sum_{k=1}^{r} \left((p_k(x,y))^2 \mathrm{var}(\delta_{a_k})\right)}{\|\nabla P_{\bar{a}}(x,y)\|^2}. \end{aligned} \tag{23}$$

Thus, when all the error components have the same variance ($\mathrm{var}(\delta_{a_k}) = \sigma^2_\delta$, like when all the coefficients are quantized with the same word length—in bits), we obtain:

$$\mathrm{var}(\varepsilon_u(x,y)) = \sigma^2_\delta \frac{\sum_{k=1}^{r} p^2_k(x,y)}{\|\nabla P_{\bar{a}}(x,y)\|^2}. \tag{24}$$

Since these properties were derived using first order approximation of the polynomial value, they are only valid when the coefficient errors are small.

1. This is a first order Taylor expansion approximation. The accuracy of the approximation depends on the magnitude of the error components.

3.3 Sensitivity Analysis of the 3L and Gradient-One Algorithms

In Section 3.1, we analyzed the sensitivity of the zero-set to small changes in the coefficient values. This analysis holds for points on the zero-set of the implicit polynomial.

When the fitting of an IP to the given data is good, the value of the polynomial at the data points is close to zero. Thus, instead of examining the sensitivity at points on the zero-set, the sensitivity may be examined at the data points. This substitution allows the evaluation of the maximal error resulting from small coefficient changes without having to find the zero-set of the fitting polynomial. Of course, this substitution should be made only when the fitting is sufficiently tight. Therefore, assuming that the 3L algorithm produces tight fitting, the error for each data set point n is bounded by (22).

The expansion and shrinking operations used by the 3L algorithm (when done very tightly about the original data set) is equivalent to differentiation of the polynomial. According to the 3L algorithm, constant values of the polynomial ($\pm\varepsilon$) are required at a fixed distance d from the data set. This implies a requirement (or constraint) for constant derivative values in the direction perpendicular to the data set, leading to a constant gradient value near the data set points: $\|\nabla P_{\bar{a}}(x_n, y_n)\| = \frac{\varepsilon}{d}, n = 1, 2, \ldots, N$ (assuming that the ratio is a good estimate of the gradient). For the gradient-one (or modified 3L) algorithm, the gradient norm is determined by explicit differentiation and is constrained to unity.

Thus, the maximal error at each data point is bounded for the above two algorithms by:

$$\begin{aligned} |\varepsilon_u(x_n, y_n)| &\leq \frac{\delta_{\max}}{\|\nabla P_{\bar{a}}(x_n, y_n)\|} \sum_{k=1}^{r} |p_k(x_n, y_n)| \\ &= \delta_{\max} c \sum_{k=1}^{r} |p_k(x_n, y_n)|, \end{aligned} \tag{25}$$

where the constant c is equal to d/ε for the 3L algorithm and $c = 1$ for the gradient-one (and, hence, also for the modified 3L) algorithm.

It is clear from (25) that both the 3L and gradient-one algorithms yields different error bound values at different data points, depending on the values of the monomial vector $\bar{p}(x_n, y_n)$ components at each data point. In the next section, we derive two algorithms that aim to produce either a constant error bound value (*Min-Max* algorithm) or a constant variance (*Min-Var* algorithm), at all the data points, and consequently achieve improved performance.

4 Improved Fitting Algorithms

In this section, we use the results of the last section to construct improved fitting algorithms. Our goal is to obtain a polynomial with a better zero-set fitting stability than the 3L and gradient-one algorithms. That is, lower sensitivity to both coefficient quantization and numerical computation errors. Numerical stability is a known problem when high order polynomials are used for fitting, as typically needed for fitting complex shapes. As could be expected, the lower sensitivity to coefficient changes improves also the fitting to noisy data, as we demonstrate in Section 5.3.

In order to obtain an optimal solution to the fitting problem, one needs first to define an optimization criterion. We are interested here in obtaining a coefficient vector $\bar{a}$ that produces a polynomial with two properties: 1) best fit to the data (i.e., $P_{\bar{a}}(x_n, y_n) = 0$), 2) minimal deviation of the zero-set due to small changes in the coefficients.

Using the error bound in (22) with a maximum coefficient error of $\delta_{\max}$, we look for a polynomial that minimizes $\delta_{\max} \sum_{k=1}^{r} |p_k(x_n, y_n)| / \|\nabla P_{\bar{a}}(x_n, y_n)\|$ at each data set point.

The first requirement above also implies that the tangent direction of the polynomial along the zero-set equals to the tangent direction of the data for each of the data set points. This is because the gradient of the polynomial is perpendicular to the zero set, leading to the requirement:

$$\frac{\partial P_{\bar{a}}(x, y)}{\partial y} \Bigg/ \frac{\partial P_{\bar{a}}(x, y)}{\partial x} \Big|_{(x_n, y_n)} = tg(\alpha_n), \tag{26}$$

where α_n is the angle of the local perpendicular to the data set about point n located at (x_n, y_n)—see Fig. 1.

Since no data point has priority over any other point (if no error weighting is used), we can limit the maximal fitting error, due to changes in the coefficients, to a constant value by requiring that the value of $\delta_{\max} \sum_{k=1}^{r} |p_k(x_n, y_n)| / \|\nabla P_{\bar{a}}(x_n, y_n)\|$ would be the same (a constant) for all the given data points, i.e., for $n = 1, \ldots, N$. Since the value of this constant does not affect the optimization, we require the following:

$$\|\nabla P_{\bar{a}}(x_n, y_n)\| = \sum_{k=1}^{r} |p_k(x_n, y_n)| \quad for\ n = 1, \ldots, N. \tag{27}$$

To further clarify this point, note that scaling the value of the gradient norm by a factor $K > 1$, may appear as causing a reduction of the value of the bound in (22). However, since this corresponds to scaling all the polynomial coefficients by K, this factor will cancel out by the necessary increase of the value of $\delta_{\max}$ by the same factor, so as to keep the relative error in the coefficients the same.

The LS solution for the requirements presented above can be calculated within the framework of the solution described in Section 2.3.1. Modifying (13) according to the requirement in (27) yields:

$$\begin{aligned} M_0 &= \left[\bar{p}^T(x_1, y_1) \quad \ldots \quad \bar{p}^T(x_N, y_N) \right] \\ M_X &= \left[\bar{p}_X^T(x_1, y_1) \Bigg/ \sum_{k=1}^{r} |p_k(x_1, y_1)| \quad \ldots \quad \bar{p}_X^T(x_N, y_N) \Bigg/ \sum_{k=1}^{r} |p_k(x_N, y_N)| \right] \\ M_Y &= \left[\bar{p}_Y^T(x_1, y_1) \Bigg/ \sum_{k=1}^{r} |p_k(x_1, y_1)| \quad \ldots \quad \bar{p}_Y^T(x_N, y_N) \Bigg/ \sum_{k=1}^{r} |p_k(x_N, y_N)| \right] \end{aligned} \tag{28}$$

which upon substitution into (12) yields a solution that we denote as the *Min-Max* solution.

Using the same formulation, we can consider a minimum variance criterion that would minimize the variance of the

TABLE 1
Summary of 2D *Min-Max* and *Min-Var* Fitting Algorithms

Data points (input)	$(x_1, y_1), \ldots, (x_N, y_N)$	
Monomial vector	$\bar{p}(x,y) = [p_1(x,y), \ldots, p_r(x,y)] =$ $[x^0y^0, x^1y^0, x^0y^1, \ldots, x^dy^0, x^{d-1}y^1, \ldots, x^1y^{d-1}, x^0y^d]$	
Parameter vector (output)	$\bar{a} = [a_1, a_2, \ldots, a_r]$	
Size of output	$r = (d+1)(d+2)/2$	
Least squares solution	Non-weighted solution	Weighted solution
	$\bar{a}_{LS} = \bar{b}M^T\left(MM^T\right)^{-1}$	$\underline{a}_{WLS} = \bar{b}W^2M^T\left(MW^2M^T\right)^{-1}$ (for diagonal W)
Structure of M and $\bar{b}$	$\bar{b} = [\bar{0} \quad \bar{v} \quad \bar{w}]$ $M = [M_0 \quad M_X \quad M_Y]$	
Contents of $\bar{b}$	$v_n = \cos(\alpha_n)$ $w_n = \sin(\alpha_n)$	
Contents of M for *Min-Max* algorithm	$M_0 = [\bar{p}^T(x_1,y_1) \;\ldots\; \bar{p}^T(x_N,y_N)]$ $M_X = \left[\bar{p}_X{}^T(x_1,y_1) \Big/ \sum_{k=1}^{r}\lvert p_k(x_1,y_1)\rvert \;\ldots\; \bar{p}_X{}^T(x_N,y_N) \Big/ \sum_{k=1}^{r}\lvert p_k(x_N,y_N)\rvert\right]$ $M_Y = \left[\bar{p}_Y{}^T(x_1,y_1) \Big/ \sum_{k=1}^{r}\lvert p_k(x_1,y_1)\rvert \;\ldots\; \bar{p}_Y{}^T(x_N,y_N) \Big/ \sum_{k=1}^{r}\lvert p_k(x_N,y_N)\rvert\right]$	
Contents of M for *Min-Var* algorithm	$M_0 = [\bar{p}^T(x_1,y_1) \;\ldots\; \bar{p}^T(x_N,y_N)]$ $M_X = \left[\bar{p}_X{}^T(x_1,y_1) \Big/ \sqrt{\sum_{k=1}^{r} p_k{}^2(x_1,y_1)} \;\ldots\; \bar{p}_X{}^T(x_N,y_N) \Big/ \sqrt{\sum_{k=1}^{r} p_k{}^2(x_N,y_N)}\right]$ $M_Y = \left[\bar{p}_Y{}^T(x_1,y_1) \Big/ \sqrt{\sum_{k=1}^{r} p_k{}^2(x_1,y_1)} \;\ldots\; \bar{p}_Y{}^T(x_N,y_N) \Big/ \sqrt{\sum_{k=1}^{r} p_k{}^2(x_N,y_N)}\right]$	

error. Modifying (13) as shown in (29) and substituting in (12) yields a solution that we denote as the *Min-Var* solution.

$$M_0 = \left[\bar{p}^T(x_1,y_1) \quad \ldots \quad \bar{p}^T(x_N,y_N)\right]$$
$$M_X = \left[\bar{p}_X^T(x_1,y_1) \Big/ \sqrt{\sum_{k=1}^{r} p_k^2(x_1,y_1)} \quad \ldots \quad \bar{p}_X^T(x_N,y_N) \Big/ \sqrt{\sum_{k=1}^{r} p_k^2(x_N,y_N)}\right]$$
$$M_Y = \left[\bar{p}_Y^T(x_1,y_1) \Big/ \sqrt{\sum_{k=1}^{r} p_k^2(x_1,y_1)} \quad \ldots \quad \bar{p}_Y^T(x_N,y_N) \Big/ \sqrt{\sum_{k=1}^{r} p_k^2(x_N,y_N)}\right]. \tag{29}$$

The above formulation was done with the assumption that all the data points have the same priority—in terms of goodness of fit. Prioritizing the data points (i.e., giving different weights to the errors at different points) yields the following weighted cost function:

$$E_W = \bar{e}_W\bar{e}_W^T = \left(\bar{a}M - \bar{b}\right)WW^T\left(\bar{a}M - \bar{b}\right)^T, \tag{30}$$

where W is a $3N \times 3N$ diagonal weighting matrix whose diagonal elements are the relative weights. The weighted-LS solution is:

$$\underline{a}_{WLS} = \bar{b}WW^TM^T\left(MWW^TM^T\right)^{-1} = \bar{b}W^2M^T\left(MW^2M^T\right)^{-1}. \tag{31}$$

Note that the above improved fitting algorithms for 2D curves, can be simply extended to 3D surfaces by applying the following modifications: 1) All coordinates are given in 3D. 2) Perpendicular vectors are now calculated as normals to tangent surfaces (instead of normals to lines). The pertinent equations are presented in Table 2.

4.1 Summary of Fitting Algorithms

Tables 1 and 2 summarize the imporved fitting algorithms, presented in this section, for fitting 2D and 3D curves.

5 Simulation Results

In this section, we present simulation results of comparisons made between the *modified 3L* fitting-algorithm (i.e., the *gradient-one algorithm* with $\mu = 1$, see Section 2.3) and the proposed *Min-Max* and *Min-Var* algorithms, derived in Section 4. Simple scaling in each axis is used to normalize the 2D data used in all experiments to the range [-1,1].

5.1 Sensitivity to Coefficient Changes

A convenient way to demonstrate the characteristics of the sensitivity function of the zero-set to coefficient changes, for the different algorithms, is via quantization of the polynomial coefficients. Quantization of the coefficients with a given number of bits gives only a single instance of the coefficient error vector and does not allow a statistical analysis of the error properties. We use, therefore, uniformly distributed random noise that is added to the polynomial coefficients to simulate the effects of quantization or small random coefficient changes. The results of this test would indicate which algorithm yields better fitting results under small coefficient changes.

The shape shown in Fig. 3, having 252 data points, was used to test and compare the *modified 3L*, *Min-Var*, and *Min-Max* algorithms in this experiment. Random noise, having a uniform distribution in the range corresponding to the least significant bit (LSB) in the binary representation of the normalized coefficients to the range of [-1,1], was generated.

TABLE 2
Summary of 3D *Min-Max* and *Min-Var* Fitting Algorithms

Data points (input)	$(x_1, y_1, z_1), \ldots, (x_N, y_N, z_N)$	
Monomial vector	$\overline{p}(x,y,z) = [p_1(x,y,z), \ldots, p_r(x,y,z)] =$ $[x^{l_1} y^{m_1} z^{n_1}, \ldots, x^{l_r} y^{m_r} z^{n_r} \ : l_i + m_i + n_i \le d]$	
Parameter vector (output)	$\overline{a} = [a_1, a_2, \ldots, a_r]$	
Size of output	$r = 2d^3 + 12d^2 + 22d + 12$	
Least squares solution	Non-weighted solution	Weighted solution
	$\overline{a}_{LS} = \overline{b} M^T (MM^T)^{-1}$	$\underline{a}_{WLS} = \overline{b} W^2 M^T (MW^2 M^T)^{-1}$ (for diagonal W)
Structure of M and $\overline{b}$	$\overline{b} = [\overline{0} \quad \overline{v} \quad \overline{w} \quad \overline{q}]$ $M = [M_0 \quad M_X \quad M_Y \quad M_Z]$	
Contents of M for *Min-Max* algorithm	$M_0 = [\overline{p}^T(x_1,y_1,z_1) \ \ldots \ \overline{p}^T(x_N,y_N,z_N)]$ $M_X = \left[\overline{p}_X{}^T(x_1,y_1,z_1) / \sum_{k=1}^{r} \lvert p_k(x_1,y_1,z_1) \rvert \ \ldots \ \overline{p}_X{}^T(x_N,y_N,z_N) / \sum_{k=1}^{r} \lvert p_k(x_N,y_N,z_N) \rvert \right]$ $M_Y = \left[\overline{p}_Y{}^T(x_1,y_1,z_1) / \sum_{k=1}^{r} \lvert p_k(x_1,y_1,z_1) \rvert \ \ldots \ \overline{p}_Y{}^T(x_N,y_N,z_N) / \sum_{k=1}^{r} \lvert p_k(x_N,y_N,z_N) \rvert \right]$ $M_Z = \left[\overline{p}_Z{}^T(x_1,y_1,z_1) / \sum_{k=1}^{r} \lvert p_k(x_1,y_1,z_1) \rvert \ \ldots \ \overline{p}_Z{}^T(x_N,y_N,z_N) / \sum_{k=1}^{r} \lvert p_k(x_N,y_N,z_N) \rvert \right]$	
Contents of M for *Min-Var* algorithm	$M_0 = [\overline{p}^T(x_1,y_1,z_1) \ \ldots \ \overline{p}^T(x_N,y_N,z_N)]$ $M_X = \left[\overline{p}_X{}^T(x_1,y_1,z_1) / \sqrt{\sum_{k=1}^{r} p_k{}^2(x_1,y_1,z_1)} \ \ldots \ \overline{p}_X{}^T(x_N,y_N,z_N) / \sqrt{\sum_{k=1}^{r} p_k{}^2(x_N,y_N,z_N)} \right]$ $M_Y = \left[\overline{p}_Y{}^T(x_1,y_1,z_1) / \sqrt{\sum_{k=1}^{r} p_k{}^2(x_1,y_1,z_1)} \ \ldots \ \overline{p}_Y{}^T(x_N,y_N,z_N) / \sqrt{\sum_{k=1}^{r} p_k{}^2(x_N,y_N,z_N)} \right]$ $M_Z = \left[\overline{p}_Z{}^T(x_1,y_1,z_1) / \sqrt{\sum_{k=1}^{r} p_k{}^2(x_1,y_1,z_1)} \ \ldots \ \overline{p}_Z{}^T(x_N,y_N,z_N) / \sqrt{\sum_{k=1}^{r} p_k{}^2(x_N,y_N,z_N)} \right]$	
Contents of $\overline{b}$	$[v_n \quad w_n \quad q_n]^T = \overline{u}_n$	
Normal vector to data surface at point n	$\overline{u}_n$	

The same noise vector is added to the coefficients obtained for each of the examined fitting algorithms. The statistics of the results obtained from using 100 independent noise vectors for each of three different noise levels—corresponding to quantization by 9 bits are shown in Table 3.

For each algorithm two error measures are considered:

$$E_{RMS} \triangleq \sqrt{\frac{1}{N}\sum_{n=1}^{N} e_n^2}; \quad E_{MAX} \triangleq \max\{e_1, \ldots, e_N\}, \tag{32}$$

where the error e_n for each data point is the distance between the nearest point on the polynomial zero-set and the data point (x_n, y_n) and where N is the number of points in the data set.

In the table, the mean and variance of E_{MAX} and the value of E_{RMS} over all data points and 100 error vectors are given for the three examined algorithms.

By examining this table, it is evident that both the Min-Max and Min-Var algorithms result in significantly lower fitting errors, in terms of both error measures, than the modified 3L algorithm. This could be expected from the sensitivity considerations discussed in the previous two sections. Similar conclusions were obtained for a wide range of noise levels and different polynomial orders. The two proposed algorithms demonstrate similar performance, although the Min-Max algorithm appears to result in somewhat lower error-measure values, on average, in the above experiment.

5.1.1 Graphical Demonstration

In this section, we graphically demonstrate some of the results obtained in the above experiment.

In the first (left) column of Fig. 3, the values of the sensitivity function, for each point of the data set, are shown for each of the eighth order polynomials that were obtained by the three examined fitting algorithms. Since the error values are computed in the direction perpendicular to the desired zero-set, the value of the error bound in (22) at each data point is plotted in Fig. 3 as a vector that is perpendicular to the zero-set, with a length proportional to the value of bound.

It is clear from the results demonstrated in Fig. 3 that the *modified 3L* algorithm is the most sensitive of the three examined algorithms and that the proposed *Min-Max* and *Min-Var* algorithms are significantly better.

It is noted that the fitting by the proposed algorithms is much better even without coefficient noise (left column). As a matter of fact, the increased sensitivity in using an eighth order polynomial by the modified 3L algorithm results in a

TABLE 3
Comparison of Error Statistics for the *Modified 3L*, *Min-Var*, and *Min-Max* Fitting Algorithms

Algorithm →	*Modified 3L*		*Min-Var*		*Min-Max*	
Error Measure ↓	Mean	Variance	Mean	Variance	Mean	Variance
E_{MAX}	0.3579	0.0272	0.0931	0.0115	**0.0649**	**0.0021**
E_{RMS}	0.1009		0.0363		**0.0216**	

The results shown are for an eighth order polynomial and a noise level corresponding to 9-bit coefficient quantization. The statistics is based on 100 independent error vectors.

gap in the zero-set (upper left side of the original shape) and the shape is not fully fitted.

The middle column in Fig. 3, shows the fitting results obtained for a single instance of random noise added to the coefficients, equivalent to quantization with a 9-bit quantizer. The fitting with the modified 3L algorithm completely breaks down, while the proposed algorithms perform well.

In the third column, we show an overlay of 20 fittings by each of the algorithms when 20 independent random noise vectors are added to the coefficients (in Table 3 the statistics is given for 100 independent noise vectors). Again, the advantage of the proposed algorithms over the modified 3L algorithm is clearly seen.

As mentioned earlier, the spurious zeros sets seen in Fig. 3 are a common phenomenon, and usually do not affect the utilization of the IP coefficients in some tasks, like object recognition. If shape reconstruction is required, it is important that these sets do not intersect with the main zero-set that's fitting the data. We do not address here this issue. We refer to possible future work on this subject in the summary.

5.2 Fitting Noisy Data

Four of the shapes that were used for comparing of the effects of noise in the data on the fitting results are shown in Table 4.

Since the results obtained with the *Min-Max* and *Min-Var* algorithms are quite similar, only the results of the comparison between the *modified 3L* and the *Min-Max* fitting algorithms are shown below. The IP order used to fit each shape in the figures below was selected as the lowest order that provides a reasonably good fit for **both** the original and the noisy data. The noisy data was generated by adding

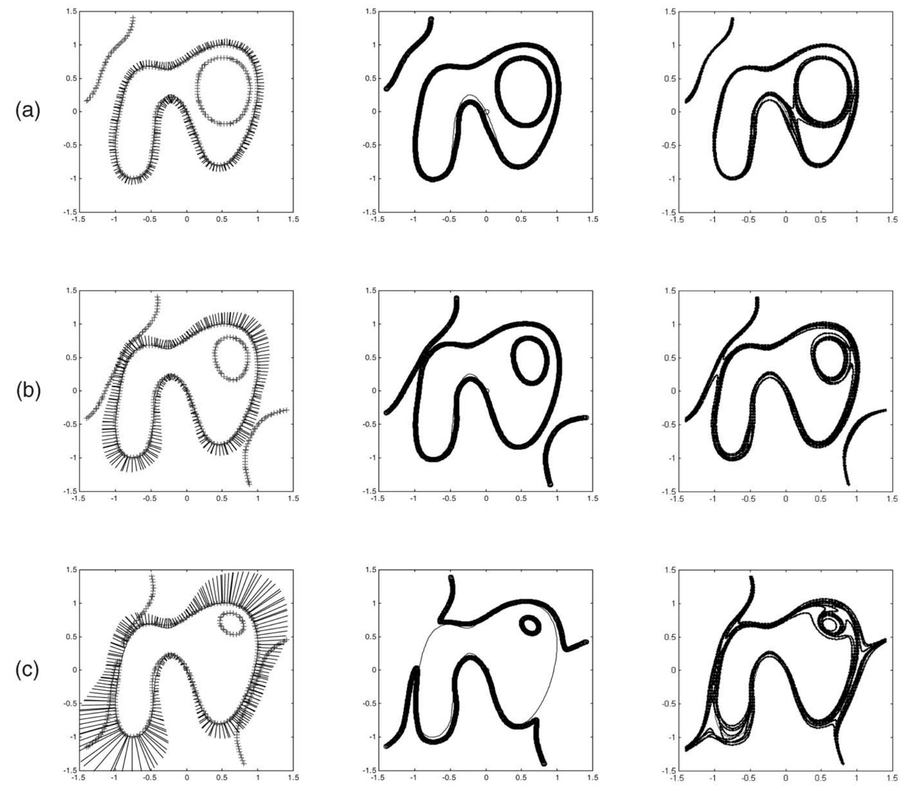
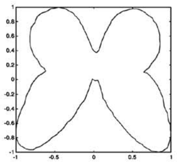
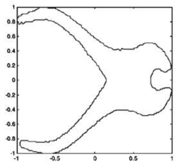
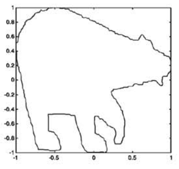
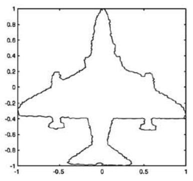


Fig. 3. Sensitivity and fitting errors for the *modified 3L*, *Min-Var*, and *Min-Max algorithms*, using eighth order polynomials. Rows: (a) Upper: *Min-Max* algorithm, (b) Middle: *Min-Var* algorithm, and (c) Lower: *Modified 3L* algorithm. Left column: Coefficients are not quantized. Center column: Single instance of added noise (ninth LSB). Right column: Overlay of 20 fittings using uniform random noise (ninth LSB) to the coefficients.

TABLE 4
Shapes Used in Fitting Algorithms Comparison

Shape name	Image	Shape name	Image
Butterfly		Pliers	
Bear		Airplane	

(Data of these shapes was obtained from the Laboratory for Engineering Man/Machine Systems (LEMS), Brown University.)

colored noise to the original data. The noise is produced by generating a random sequence, with independent uniformly distributed elements in the range $[-\frac{1}{4}:\frac{1}{4}]$, and convolving this sequence with an averaging filter of length 9.

Besides a visual examination of the results shown in Figs. 4 and 5, the comparison is also based on the values of the maximal and RMS errors, E_{MAX} and E_{RMS}, respectively, as defined in (32), for each fitted shape. As in Section 5.1, the error at each data point is defined as the distance of the data point to the closest zero-set point. Note again that shapes are normalized to the range [-1,1] by simple scaling in each axis, before the fitting algorithms are applied.

The values of these error measures, obtained for the original and two noisy versions of each shape, and for each of

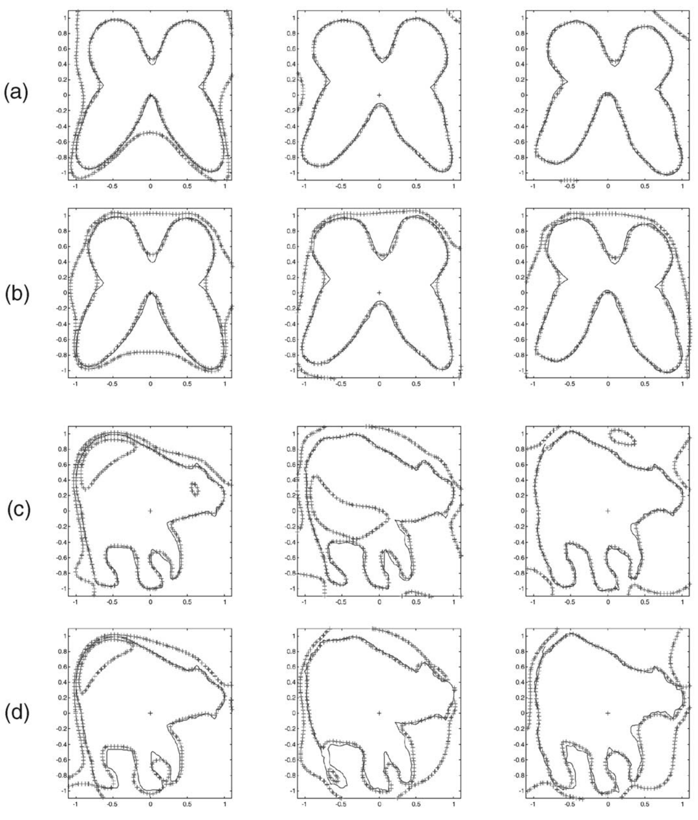


Fig. 4. Fitting results for the *Min-Max (M-M)* and *Modified-3L (M-3L)* algorithms: Rows: (a) "Butterfly" fitted with *Min-Max*, (b) "Butterfly" fitted with *M-3L*, (c) "Bear" fitted with *Min-Max*, and (d) "Bear" fitted with *M-3L*. Columns: Left-Original data, Center and Right-Two different instances of noisy data (colored noise).

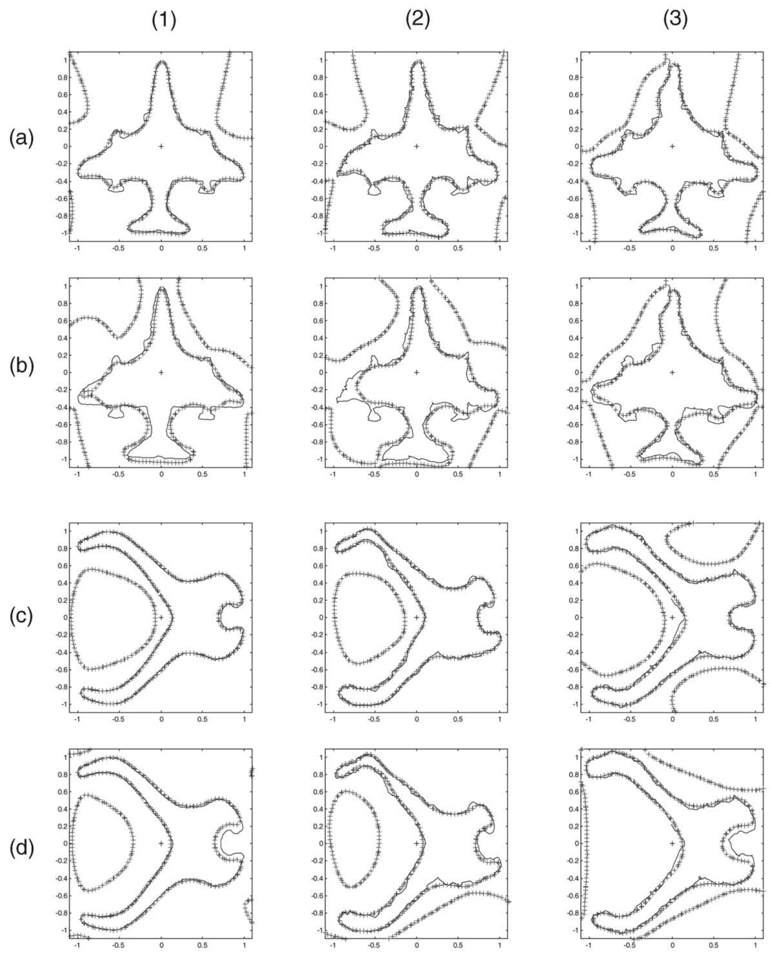


Fig. 5. Fitting results for the *Min-Max (M-M)* and *Modified-3L (M-3L)* algorithms: Rows: (a) "Airplane" fitted with *Min-Max*, (b) "Airplane" fitted with *M-3L*, (c) "Pliers' fitted with *Min-Max*, and (d) "Pliers" fitted with *M-3L*. Columns: (1) original data, (2) and (3), noisy data (colored noise).

the examined two fitting algorithms, are shown in Tables 5 and 6.

It is clear from the fitting results shown in the figures and the numerical results shown in the tables, that for all the four shapes, the *Min-Max* performs consistently better than the *modified 3L* algorithm, both visually and in terms of the error measures used. We reached the same conclusion after applying the *Min-Var* algorithm to these shapes, as well as when we applied both proposed algorithms on additional shapes (not shown here because of lack of space).

It is important to note that the advantage of the proposed algorithms over the *modified 3L* algorithm is more significant when the shape is more complex, like the "Bear" and "Airplane" shapes, i.e., when high order IPs are needed to fit the data. In particular, for these two shapes, a reduction of about 50 percent in error measure values is obtained for some of the noisy versions shown.

In this work, we address only the issue of contour fitting and do not perform any object recognition task. Yet, it should be stressed that stable fitting is an important factor in any recognition task. It doesn't appear that recognition can be done at all if the fitting breaks down, as it happens in Fig. 4, row (c), center column, where the modified 3L algorithm is failing to fully fit a noisy version of "bear."

Furthermore, even if the fitting doesn't break down, an aspect related to recognition performance is the change in the polynomial coefficient vector due to noise in the data (a form of "global stability"). To quantify this aspect, we have measured the normalized squared-norm of the difference between the coefficient vectors obtained for the clean data and noisy data (obtained by adding a small amount of colored noise), for the examined three algorithms. We have found in our simulations that, on average (over 20 noisy data sets for each shape, and several shapes), the min-max algorithm performed best, with the highest reduction in the measured normalized squared-error in the coefficient vector being obtained for the more complex shapes. For example, a reduction by a factor of 1.4 was obtained for "bear" and a factor 2 for "pliers."

The combination of improved local fitting by the proposed algorithms (for both clean and noisy data)—especially for complex shapes, and the typically smaller change (error) in the coefficient vector due to data noise, points to a better potential of these algorithms in the various tasks IPs are used for, than provided by the modified 3L algorithm.

6 Summary and Conclusions

In this paper, we introduce an approach for stable fitting by implicit polynomial of 2D curves and 3D surfaces that is based on reducing the sensitivity of the fitting polynomial zero-set to small coefficient changes.

TABLE 5
Error Statistics Obtained from Fitting Results Shown in Fig. 4 for *Min-Max (M-M)* and *Modified-3L (M-3L)* Fitting Algorithms

Shape	Measurement		1 (orig)	2 (noisy)	3 (noisy)
Butterfly	E_{MAX}	*M-M*	**0.0740**	**0.0526**	**0.0867**
		M-3L	0.1055	0.0857	0.1197
	E_{RMS}	*M-M*	**0.0191**	**0.0180**	**0.0205**
		M-3L	0.0277	0.0237	0.0279
Bear	E_{MAX}	*M-M*	**0.0616**	**0.0952**	**0.073**
		M-3L	0.1249	0.1640	0.1061
	E_{RMS}	*M-M*	**0.0209**	**0.0240**	**0.0225**
		M-3L	0.0318	0.0475	0.0335

TABLE 6
Error Statistics Obtained from Fitting Results Shown in Fig. 5 for *Min-Max (M-M)* and *Modified-3L (M-3L)* Fitting Algorithms

Shape	Measurement		1 (orig)	2 (noisy)	3 (noisy)
Airplane	E_{MAX}	*M-M*	**0.0827**	**0.1244**	**0.1062**
		M-3L	0.1471	0.2109	0.1304
	E_{RMS}	*M-M*	**0.0261**	**0.0362**	**0.0297**
		M-3L	0.0492	0.0689	0.0406
Pliers	E_{MAX}	*M-M*	**0.0353**	**0.0572**	**0.0745**
		M-3L	0.1053	0.0920	0.1044
	E_{RMS}	*M-M*	**0.0153**	**0.0207**	**0.0234**
		M-3L	0.0285	0.0284	0.0345

Based on a sensitivity analysis for 2D polynomials, we develop two fitting algorithms of 2D curves, denoted *Min-Max* and *Min-Var*. The two algorithms exhibit similar performance. The *Min-Max* algorithm attempts to minimize a bound on the maximal error due to small coefficient changes. The *Min-Var* algorithm, minimizes the error variance due to coefficient errors. We also show how the proposed algorithms should be modified to support the fitting of 3D data.

The main difference between the proposed algorithms and the known 3L and gradient-one algorithms is that instead of constraining the value of the norm of the gradient vector of the IP along the zero set to a fixed value, each of the proposed algorithms constrains this norm to different appropriate values that vary along the zero set in a data dependent manner, such that the fitting error measure used by each algorithm is uniform along the data set.

In simulations, we compare the proposed algorithms with the modified 3L algorithm (a particular instance of the gradient-one algorithm), in fitting several different object shapes. We demonstrate that the proposed algorithms provide not only better fitting for both the original data and for noisy versions obtained by adding colored noise, but also reduce, on average, the coefficient vector error due to noise in the data, particularly for complex shapes, which typically require high order IPs for proper fitting. The improved fitting and the reduced sensitivity to noise in the data could prove useful in all the applications that use implicit polynomial fitting, such as object recognition, contour coding, computer graphics, CAD and others.

Future work should consider the issue of spurious zero-sets observed in the simulations, by combining the regularization technique (ridge-regression) of [17], or the "guard-strip" of [8], with the algorithms proposed in this work.

ACKNOWLEDGMENTS

The authors thankfully acknowledge the very helpful suggestions of the anonymous reviewers. They are grateful to Professor David B. Cooper and to Mr. Andrew Willis of the Laboratory for Engineering Man/Machine Systems (LEMS), Brown University, for providing them with the data files of the shapes used in the simulations, and for allowing them to experiment with their gradient-one algorithm via their Web site. They also thank Mrs. Zoya Landa for her help in running some of the experiments presented in this paper.

REFERENCES

[1] T.W. Sederberg and D.C. Anderson, "Implicit Representation of Parametric Curves and Surfaces," *Computer Vision, Graphics, and Image Processing,* vol. 28, no. 1, pp. 72-84, 1984.

[2] G. Taubin, "Estimation of Planar Curves, Surfaces and Nonplanar Space Curves Defined by Implicit Equations, with Applications to Edge and Range Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 11, pp. 1115-1138, Nov. 1991.

[3] G. Taubin, F. Cukierman, S. Sullivan, J. Ponce, and D.J. Kriegman, "Parameterized Families of Polynomials for Bounded Algebraic Curve and Surface Fitting," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 16, pp. 287-303, 1995.

[4] D. Forsyth, J.L. Mundy, A. Zisserman, C. Coelho, A. Heller, and C. Rothwell, "Invariant Descriptors for 3D Object Recognition and Pose," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 13, no. 10, pp. 971-992, Oct. 1991.

[5] D.A. Forsyth, "Recognizing Algebraic Surfaces from Their Outlines," *Proc. Int'l Conf. Computer Vision,* pp. 476-480, May 1993.

[6] M. Barzohar, D. Keren, and D. Cooper, "Recognizing Groups of Curves Based on New Affine Mutual Geometric Invariants, with Applications to Recognizing Intersecting Roads in Aerial Images," *Proc. IAPR Int'l Conf. Pattern Recognition,* vol. 1, pp. 205-209, Oct. 1994.

[7] J.-P. Trael and D.B. Cooper, "The Complex Representation of Algebraic Curves and Its Simple Exploitation for Pose Estimation and Invariant Recognition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 7, pp. 663-674, July 2000.

[8] A. Helzer, M. Bar-Zohar, and D. Malah, "Using Implicit Polynomials for Image Compression," *Proc. 21st IEEE Convention of the Electrical and Electronic Eng. in Israel,* pp. 384-388, Apr. 2000.
[9] T. Tasdizen and D.B. Cooper, "Boundary Estimation from Intensity/Color Images with Algebraic Curve Models," *Int'l Conf. Pattern Recognition,* pp. 1225-1228, 2000.
[10] C. Bajaj, I. Ihm, and J. Warren, "Higher-Order Interpolation and Least-Squares Approximation Using Implicit Algebraic Surfaces," *ACM Trans. Graphics,* vol. 12, no. 4, pp. 327-347, 1993.
[11] J. Subrahmonia, D. Cooper, and D. Keren, "Practical Reliable Bayesian Recognition of 2D and 3D Objects Using Implicit Polynomials and Algebraic Invariants," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, pp. 505-519, 1996.
[12] J.-P. Tarel, W.A. Wolovich, and D.B. Cooper, "Covariant Conics Decomposition of Quartics for 2-D Object Recognition and Affine Alignment," *Proc. Int'l Conf. Image Processing,* Oct. 1998.
[13] M.M. Blane, Z. Lei, H. Civil, and D.B. Cooper, "The 3L Algorithm for Fitting Implicit Polynomials Curves and Surface to Data," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 3, Mar. 2000.
[14] Z. Lei and D.B. Cooper, "New, Faster, More Controlled Fitting of Implicit Polynomial 2D Curves and 3D Surfaces to Data," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* June 1996.
[15] Z. Lei and D.B. Cooper, "Linear Programming Fitting of Implicit Polynomials," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 2, pp. 212-217, Feb. 1998.
[16] D. Keren and C. Gotsman, "Fitting Curves and Surfaces with Constrained Implicit Polynomials," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 21, no. 1, pp. 31-41, Jan. 1999.
[17] T. Tasdizen, J.-P. Tarel, and D.B. Cooper, "Improving the Stability of Algebraic Curves for Applications," *IEEE Trans. Image Processing,* vol. 9, no. 3, pp. 405-416, Mar. 2000.
[18] A. Helzer, M. Bar-Zohar, and D. Malah, "Robust Fitting of Implicit Polynomials with Quantized Coefficients to 2D Data," *Proc. 15th Int'l Conf. Pattern Recognition,* pp. 290-293, Sept. 2000.
[19] A. Helzer, "Using Implicit Polynomials for Contour Coding," MSc Thesis, Technion-Israel Inst. of Technology, Haifa, Israel, Dec. 2000.
[20] C. Oden, A. Ercil, V.T. Yildiz, H. Kirmizita, and B. Buke, "Hand Recognition Using Implicit Polynomials and Geometric Features," *Proc. Third Int'l Conf. Audio-and-Video-Based Biometric Person Authentication, AVBPA-2001,* June 2001.

Amir Helzer received the BSc and MSc degrees in 1995 and 2001, respectively, from the Technion Israel Institute of Technology, Haifa, Israel in electrical engineering. Since then, he has held positions in the area of chip-design and electrical engineering in Rafael and Virata. His research interests include image processing using implicit polynomials and adaptive computing machines.

Meir Barzohar received the BS and MS degrees in electrical engineering from the Technion Israel Institute of Technology in 1973 and 1978, respectively, and the MS degree in applied mathematics and PhD degree in electrical engineering from Brown University in 1993, working on new geometric, algebraic, and probabilistic models for finding and recognizing roads and their features in aerial images. Dr. Barzohar has been employed by Rafael Israel since 1974 as a researcher engineer, in the areas of communication, digital signal processing, image processing, and image compression. In 1982, he was on a sabbatical at RCA Laboratories, Princeton New Jersey working on digital video signal processing for FM transmission and television scrambler for which he received a patent. From 1989 to 1994, he was granted a sabbatical and leave of absence from Rafael to join Brown University to pursue the PhD degree and later a postdoctoral position. From 1994 to 2001, he was a senior research scientist in the computer vision group in Rafael, working on probabilistic models for detection and tracking objects in image sequences. Since 2001, he has been at Visionsense medical company, Petah Tikva Israel; as a group leader in the image processing group and working on 3D geometric camera models, image processing for development of stereoscopic endoscope. Dr. Barzohar is also an adviser for master degree students in the Electrical Engineering Department of the Technion. His research interests are computer vision, 2D and 3D object recognition, Bayesian estimation and decision theoretic framework for image segmentation, hyperspectral image analysis and representation approaches, image compression, image processing.

David Malah received the BSc and MSc degrees in 1964 and 1967, respectively, from the Technion Israel Institute of Technology, Haifa, Israel, and the PhD degree in 1971 from the University of Minnesota, Minneapolis, all in electrical engineering. From 1971-1972, he was an assistant professor in the Electrical Engineering Department of the University of New Brunswick, Fredericton, NB, Canada. In 1972, he joined the Electrical Engineering Department of the Technion, where he is a full professor, holding the Elron/Elbit Chair in Electrical Engineering. During the period 1979 to 2001, he spent about six years, cumulatively, of sabbaticals and summer leaves at AT&T Bell Laboratories, Murray Hill New Jersey, and AT&T Labs, Florham Park New Jersey, performing research in the areas of speech and image communication. Since 1975, he has been the academic head of the Signal and Image Processing Laboratory (SIPL), at the Technion, Electrical Engineering Department, which is active in image and speech communication research and education. His main research interests are in image, video, speech and audio coding; speech enhancement; image processing; digital watermarking applications, and in digital signal processing techniques. He has been a fellow of the IEEE since 1987.