CCIT Report #406 December 2002

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

Amir Helzer, Meir Barzohar and David Malah
Department of Electrical Engineering
Technion- Israel Institute of Technology, Haifa 32000, Israel
ahelzer@virata.com, meirb@envision.co.il, malah@ee.technion.ac.il

Index Terms: Implicit polynomials, zero set sensitivity, curve and surface fitting, robust fitting, coefficient quantization.

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 presented in this paper aim at producing polynomials that are robust to coefficient errors. Special emphasis is given here to errors due to coefficient quantization. The development of the algorithms begins with an analysis of zero-set errors caused by coefficient errors. The result of this analysis provides means for evaluating the performance of existing fitting algorithms and for the development of new algorithms that yield more stable polynomials. We also show that although the proposed algorithms are designed to be robust to coefficient errors, they also produce tighter fits than the other algorithms examined, even when the coefficients are practically unquantized.

1 Introduction

Implicit polynomials (IP) have long been introduced for fitting 2D curves and 3D surfaces [1,2,3]. 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], coding [8], 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.

In all the applications involving implicit polynomials, the basic and most desired properties are tight fitting and stability. Besides a tight fit, the curve described by an IP must be stable and not change much when the polynomial coefficients change only slightly, like when the coefficients are represented by a limited number of bits. These requirements are the basis for the polynomial-fitting algorithms described in this paper.

In recent years, the subject of implicit polynomial fitting has seen much progress - beginning with the classical *least-squares* (LS) fitting algorithm [1] and followed by the 3L fitting algorithm [12,13] and others [14-18]. The classical least-squares fitting algorithm suffers from a very high sensitivity to coefficient errors. This high sensitivity is attributed to insufficient constraining that results in too many degrees of freedom. To alleviate the sensitivity problem, the 3L fitting-algorithm introduces additional constraints, which are generated from the original data via expansion and shrinking. This solution provides much improved performance, in terms of stability and tightness, as compared to the earlier LS algorithm. However, it still suffers from numerical problems as a result of expanding and shrinking discrete data. This algorithm was further improved in [17] by basing the fitting on ridge regression.

In our study, we achieve improved fitting stability by considering the sensitivity of the polynomial coefficients to quantization. Our work actually originated from a coding application [8], where our concern was that high sensitivity of the coefficients results in a higher bit rate.

Therefore, reducing the sensitivity as much as possible was crucial. To accomplish this goal, we investigated the sensitivity of zero-set points to coefficient quantization. We developed a sensitivity function, which evaluates the zero-set errors due to small changes in coefficient values. Having developed this sensitivity function, it became possible to analyze the error properties of existing algorithms (classical least-squares and 3L*)* and to develop our own fitting algorithms, which minimize the sensitivity to coefficient quantization. The sensitivity analysis indicates that the sensitivity is controlled by two attributes: The gradient of the polynomial at the zero-set point, and the value of its monomial[1]. We noticed that earlier work dealt only with the gradient, without relating to the value of the monomial. We show that the minimization of the value of the sensitivity function at the data points improves the fitting stability to coefficient quantization. The proposed algorithms are denoted here as *Min-Max* and *Min-Var* algorithms. Furthermore, we conclude (and demonstrate by simulations) that these algorithms yield tighter and more robust fitting, even when the coefficients are practically not quantized. Improved fitting results are especially noticeable when high-order polynomials are used (6th and above). Thus, the algorithms developed to optimize polynomial fitting for coding turn out to be very useful for other applications as well.

The layout of this paper is as follows:

Section 2 provides background material on polynomial fitting and outlines the *3L* fitting-algorithm, from which our algorithms evolved. Section 3 provides an analytical tool for calculating bounds on the variation of zero-sets of implicit polynomials due to small errors in the polynomial's coefficients values. Section 4 describes the development of the proposed fitting algorithms. Section 5 presents simulation results in comparing the proposed algorithms with the 3L algorithm. Section 6 summarizes and concludes the paper.

[1] See (1) in section 2.1.

2 Background

In this section we provide a brief overview of implicit polynomials fitting.

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's coefficients), and a vector of monomials.
For a dth order polynomial, the monomial vector is defined as:

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

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

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

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

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), $n = 1,\dots,N$.

We denote the zero-set of the polynomial defined by the coefficient vector $\bar{a}$ as:

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

2.1.1 Overview of the 3L algorithm

The 3L algorithm developed in [12,13] 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 calculated. 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 require an iterative solution, with unproven convergence properties.

The 3L algorithm is based on the construction of two additional data 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 (see Fig. 1). 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.

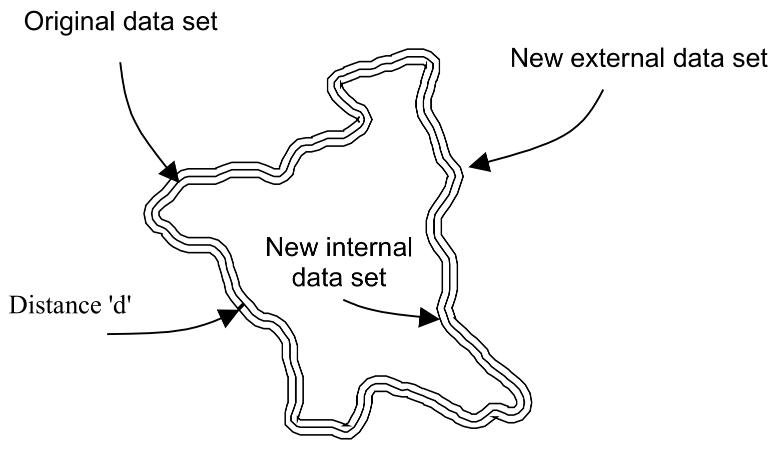

Fig. 1: Original data set (center line) and new internal and external data sets.

Considering the original data set points and the two added sets (internal and external) as a single set of $3N$ points. The first N points are the original data points (points $1,...,N$), the second group of N points ($N+1,...,2N$) are the external points, and the third group of N points ($2N+1,...,3N$) are the internal points.

Thus, for N original data set points and r polynomial coefficients we obtain $3N$ equations with r variables. For the fitting problem to be over-determined, the relation r < 3 N should be satisfied.

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

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

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

$$E=\bar{e}\,\bar{e}^{T} \qquad (5)$$

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

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

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

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

with,

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

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.

The *least squares* (LS) solution for the problem of obtaining $\bar{a}$ that minimizes E in (5) is:

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

This solution is feasible when the minimization problem is over-determined.

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 outside and negative on the inside - in the above construction).

3 Sensitivity to coefficient errors

In this section we consider the effect of coefficient quantization on the location of the zero-set points via a sensitivity function. We begin with a formulation of bounds on the error at the location of the zero-set points due to coefficients quantization. We than compute these error bounds for the 3L fitting algorithm presented in section 2.1.1. As a result, we find that the 3L algorithm can be improved by minimizing not only the distance between the zero-set and the data points but also the sensitivity of the zero-set to small changes (or errors) in the coefficients. We denote the improved algorithms as – *Min-Max* and *Min-Var*, according to the optimized property.

3.1 Zero-set sensitivity to parameter changes

When the values of the coefficients in the parameter vector change, the entire zero-set changes. In this subsection we examine how changes in the parameter 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 function at a zero-set point $\bar{z} = (z_x, z_y)$ by the following $2 \times r$ matrix:

$$S_{\bar{a}}^{\bar{z}}(x, y) = \frac{d\bar{z}(x, y)}{d\bar{a}} \tag{10}$$

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 point $\bar{\varepsilon}_Z(x,y)=[\bar{\varepsilon}_X \quad \bar{\varepsilon}_Y]$, 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 function[2]:

$$\bar{\varepsilon}_Z(x,y)\cong \bar{S}_{\bar{a}}^{\bar{z}}(x,y)\bar{\delta}_a^{\,T} \tag{11}$$

3.1.1 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):

$$u(x,y)=\langle \bar{z}(x,y)\cdot \nabla P_{\bar{a}}(x,y)\rangle \tag{12}$$

where $\langle a\cdot b\rangle$ denotes the inner-product between a and b, and $\nabla f(x,y)$ 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 $S_{\bar{a}}^{u}(x,y)=\frac{du(x,y)}{d\bar{a}}$, 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.

This function can be written as a product of two independent parts:

$$\bar{S}_{\bar{a}}^{u}(x,y)=\frac{du}{dP_{\bar{a}}(x,y)}\frac{dP_{\bar{a}}(x,y)}{d\bar{a}} \tag{13}$$

The right-hand part of the product in (13) 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

[2] This is a 1st order Taylor expansion approximation. The accuracy of the approximation depends on the magnitude of the error components.

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 (13) 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.

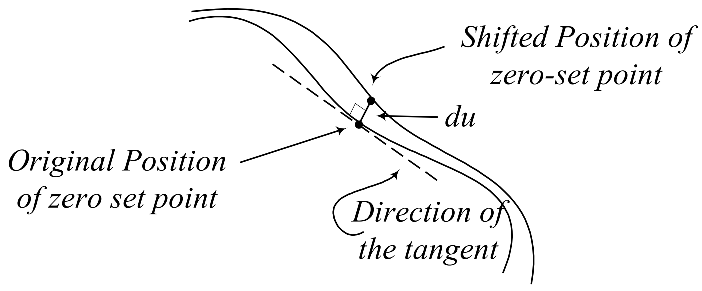
Shifted Position of zero-set point
du
Original Position of zero set point
Direction of the tangent

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

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 $P_{\bar{a}}(x, y)$:

$$\frac{du}{dP_{\bar{a}}}(x,y) = \frac{1}{\|\nabla P_{\bar{a}}(x,y)\|} = \frac{1}{\sqrt{\left(\frac{\partial P_{\bar{a}}(x,y)}{\partial x}\right)^2 + \left(\frac{\partial P_{\bar{a}}(x,y)}{\partial y}\right)^2}} \tag{14}$$

The right-hand part of the sensitivity function in (13), $\frac{dP_{\bar{a}}(x,y)}{d\bar{a}}$, 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:

$$\frac{dP_{\bar{a}}(x,y)}{d\bar{a}} = \bar{p}(x,y). \tag{15}$$

Using (14) and (15), the sensitivity function in (13) can now be written as:

$$\bar{S}_{\bar{a}}^{u}(x,y)=\frac{\bar{p}(x,y)}{\left\|\nabla P_{\bar{a}}(x,y)\right\|} \tag{16}$$

3.1.2 Zero-set fitting error bounds and variance

Having obtained (16), we use it to obtain bounds on the fitting errors due to small changes in the coefficients. The error in the direction perpendicular to the zero-set, $\varepsilon_u(x,y)$, is:

$$\varepsilon_u(x,y)=\bar{S}_{\bar{a}}^{u}(x,y)\cdot\bar{\delta}_a^{\,T}=\frac{\bar{p}(x,y)}{\left\|\nabla P_{\bar{a}}(x,y)\right\|}\bar{\delta}_a^{\,T}=\frac{\sum_{k=1}^{r}p_k(x,y)\delta_{a_k}}{\left\|\nabla P_{\bar{a}}(x,y)\right\|} \tag{17}$$

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

$$\left|\varepsilon_u(x,y)\right|=\frac{\left|\sum_{k=1}^{r}p_k(x,y)\delta_{a_k}\right|}{\left\|\nabla P_{\bar{a}}(x,y)\right\|}\le\frac{\sum_{k=1}^{r}\left|p_k(x,y)\right|\left|\delta_{a_k}\right|}{\left\|\nabla P_{\bar{a}}(x,y)\right\|}\le\delta_{\max}\frac{\sum_{k=1}^{r}\left|p_k(x,y)\right|}{\left\|\nabla P_{\bar{a}}(x,y)\right\|} \tag{18}$$

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

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:

$$\operatorname{var}\left(\varepsilon_u(x,y)\right)=\frac{\operatorname{var}\left(\sum_{k=1}^{r}p_k(x,y)\delta_{a_k}\right)}{\left\|\nabla P_{\bar{a}}(x,y)\right\|^2}=\frac{\sum_{k=1}^{r}\left(\left(p_k(x,y)\right)^2\operatorname{var}\left(\delta_{a_k}\right)\right)}{\left\|\nabla P_{\bar{a}}(x,y)\right\|^2}. \tag{19}$$

Thus, when all the error components have the same variance ($\operatorname{var}\left(\delta_{a_k}\right)=\sigma_\delta^2$, like when all the coefficients are quantized with the same word length – in bits), we obtain:

$$\operatorname{var}\left(\varepsilon_u(x,y)\right)=\sigma_\delta^2\frac{\sum_{k=1}^{r}p_k^{\,2}(x,y)}{\left\|\nabla P_{\bar{a}}(x,y)\right\|^2} \tag{20}$$

Since these properties were derived using 1st order approximation of the polynomial value, they are only valid when the coefficient errors are small.

3.2 Analysis of the 3L algorithm

In section 3.1 we analyzed the sensitivity of the zero-set to small changes in the coefficient vector. 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 checking 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 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 (18).

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 for constant derivative values in the direction perpendicular to the data set, leading to a constant gradient value near the data set points: $\left\|\nabla P_{\bar{a}}\left(x_n, y_n\right)\right\| = \frac{\varepsilon}{d}, \ n = 1,2,\ldots,N.$

The maximal error for each data points is therefore bounded by:

$$\left|\varepsilon_u\left(x_n, y_n\right)\right| \le \frac{\delta_{\max}}{\left\|\nabla P_{\bar{a}}\left(x_n, y_n\right)\right\|} \sum_{k=1}^{r} \left|p_k\left(x_n, y_n\right)\right| = \delta_{\max} \frac{d}{\varepsilon} \sum_{k=1}^{r} \left|p_k\left(x_n, y_n\right)\right| \qquad (21)$$

It is clear from (21) that the 3L algorithm yields different error bound values at different data points, depending on the value of the monomial vector $\bar{p}(x_n, y_n)$ at each data point. In section 4 we derive algorithms that aim to produce constant error bound values at all the data points.

4 Stable Fitting Algorithm

In this section we use the results of section 3.1 to construct improved fitting algorithms. Our goal is to obtain a polynomial with a better zero-set stability than the 3L algorithm, w.r.t. changes in the coefficients.

We begin by modifying the original 3L algorithm. This modification comprises the replacement of the expansion and shrinking operations with differentiation of the polynomial. The motivation for and the description of this modification are presented in section 4.1. In section 4.2 we continue by modifying the fitting cost function by the polynomial to achieve desired stability properties. Section 4.3 describes the extension of the results to 3D polynomial fitting of surfaces.

4.1 Modification of the 3L algorithm

In the 3L algorithm, to achieve a solution that tightly fits the data, it is needed to generate the internal and external data sets close to the original data points. This implies that for tight fitting results ε and d (defined in subsection 2.1.1) should approach zero.

If we attempt to satisfy this demand and examine the results of the *least squares* solution $\left(\bar{a}_{LS} = \bar{b}M^T\left(MM^T\right)^{-1}\right)$, we would notice that the factor $M^T\left(MM^T\right)^{-1}$ heads to infinity while $\bar{b}$ goes to zero. This means that small numerical errors, resulting from inverting the matrix $\left(MM^T\right)$ or from the process of creating the internal and external data sets, will be manifested by large fitting errors.

Fitting attempts have shown that it is possible to choose the value of the parameter d at about 5% of the geometric size of the object described by the data and obtain stable results. Smaller values of d (which are desirable for obtaining a tighter fit) yield less stable fitting results.

Choosing large values for the parameter d can also cause problems of the type seen in Fig. 3:

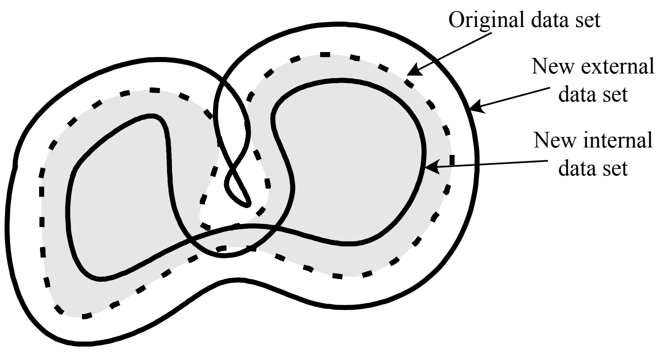

Fig. 3: Demonstration of possible expansion and shrinking results when a large value of d, relative to the object size, is used.

As seen in this example, choosing a large value of d may lead to the generation of conflicting requirements if the involved sets (original, internal and external) overlap, thus leading to poor fitting results.

We describe here a method for replacing the added sets by explicit differentiation of the polynomial. The motivation for this replacement is as follows.

The basic solution, including the expanding and shrinking of the original data points and the solution of (9), is actually an implementation of an approximated differentiation. Fig. 4 demonstrates the location of the added internal and external sets and the values of the polynomial at these sets. This figure also shows how the required values generate an implicit requirement for the value of the derivative of the polynomial in the direction perpendicular to the original data set.

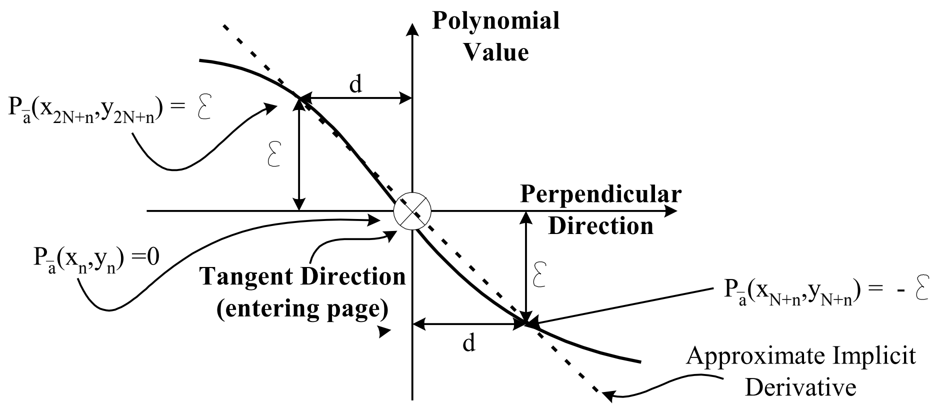
Polynomial Value
d
$P_{\bar a}(x_{2N+n},y_{2N+n}) = \varepsilon$
ε
Perpendicular Direction
$P_{\bar a}(x_n,y_n)=0$
Tangent Direction (entering page)
ε
$P_{\bar a}(x_{N+n},y_{N+n}) = -\varepsilon$
d
Approximate Implicit Derivative

Fig. 4: View of added internal and external points from the tangent direction at (x_n, y_n)

By replacing the numerical approximation of the differentiation operation (implemented via the expansion and shrinking of the data) with an explicit analytical differentiation of the polynomial, we avoid the numerical difficulties in the original 3L algorithm.

We begin by calculating the differentials of the monomial in both axes. The vectors $\overline{p}_X(x_n, y_n), \overline{p}_Y(x_n, y_n)$ denote the derivatives with respect to x and y, respectively, of the monomial vector at point (x_n, y_n):

$$\begin{aligned} \overline{p}_X(x_n,y_n) &= \frac{d}{dx}\overline{p}(x,y)\Big|_{(x=x_n,y=y_n)} \\ \overline{p}_Y(x_n,y_n) &= \frac{d}{dy}\overline{p}(x,y)\Big|_{(x=x_n,y=y_n)} \end{aligned} \tag{22}$$

Multiplying the derivatives of the monomials by the coefficient vector yields the respective derivatives of the polynomial according to:

$$\begin{aligned} \frac{d}{dx}P_{\overline{a}}(x,y)\Big|_{(x=x_n,y=y_n)} &= \overline{a}\ \ \overline{p}_X^{\ T}(x_n,y_n) \\ \frac{d}{dy}P_{\overline{a}}(x,y)\Big|_{(x=x_n,y=y_n)} &= \overline{a}\ \ \overline{p}_Y^{\ T}(x_n,y_n) \end{aligned} \tag{23}$$

We attempt to bring the polynomial differential at the location of the original data set points to the direction of the line locally perpendicular to the data set near each data set point (see Fig .5).

The angle relative to the X axis of the perpendicular vector is denoted by α_n, as shown in Fig. 5, which also shows that in order to calculate α_n, we fit a straight line (1st order polynomial) to the points about the data point (x_n, y_n) under consideration.

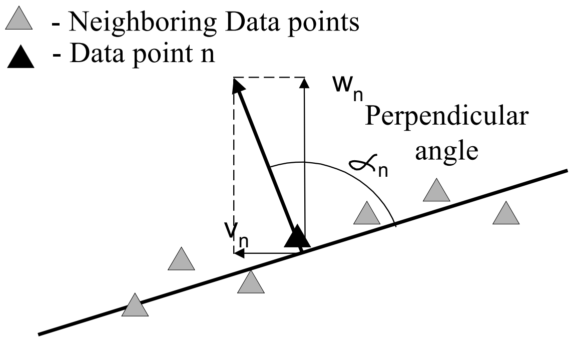

Fig .5: Fitting a 1st order polynomial to 7 points about the point(x_n, y_n)

The vectors $\overline{v} = [v_1,..., v_N]$, $\overline{w} = [w_1,..., w_N]$ (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 $\overline{v}$, $\overline{w}$, 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).

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

$$\left.\begin{array}{l} \dfrac{v_n}{w_n} = tg(\alpha_n) \\ \sqrt{\left(v_n^{\ 2} + w_n^{\ 2}\right)} = \dfrac{\varepsilon}{d} \end{array}\right\} \Rightarrow \begin{array}{l} v_n = \dfrac{\varepsilon}{d}\sin(\alpha_n) \\ w_n = \dfrac{\varepsilon}{d}\cos(\alpha_n) \end{array} \qquad (24)$$

The values of $\overline{b}$ 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 an absolute value of $\dfrac{\varepsilon}{d}$ (as seen in Fig. 4), and to keep the value of the polynomial at the location of the data points equal to zero, $\overline{b}$ and M become:

$$\begin{aligned} \bar{b} &= \begin{bmatrix} \bar{0} & \bar{v} & \bar{w} \end{bmatrix} \\ M &= \begin{bmatrix} M_0 & M_X & M_Y \end{bmatrix} \end{aligned} \tag{25}$$

where,

$$\begin{aligned} M_0 &= \left[\bar{p}^T(x_1, y_1) \ \ \ldots \ \ \bar{p}^T(x_N, y_N) \right] \\ M_X &= \left[\bar{p}_X{}^T(x_1, y_1) \ \ \ldots \ \ \bar{p}_X{}^T(x_N, y_N) \right] \\ M_Y &= \left[\bar{p}_Y{}^T(x_1, y_1) \ \ \ldots \ \ \bar{p}_Y{}^T(x_N, y_N) \right] \end{aligned} \tag{26}$$

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

In the 3L algorithm, the required value of the gradient at each data point is set to a constant value of[3] ε / d. Equation (9) can, therefore, be written as:

$$\bar{a}_{LS} = \frac{\varepsilon}{d} \bar{b}_{norm} M^T \left(M M^T \right)^{-1} \tag{27}$$

where, in the vector $\bar{b}_{norm}$ the elements of vectors $\bar{v}, \bar{w}$, in $\bar{b}$ are replaced by the normalized elements $v_{norm,n}$, $w_{norm,n}$, satisfying

$$\left. \begin{aligned} &\frac{v_{norm,n}}{w_{norm,n}} = tg(\alpha_n) \\ &\sqrt{\left(v_{norm,n}{}^2 + w_{norm,n}{}^2 \right)} = 1 \end{aligned} \right\} \Rightarrow \begin{aligned} v_{norm,n} &= \sin(\alpha_n) \\ w_{norm,n} &= \cos(\alpha_n) \end{aligned} \tag{28}$$

Changing the scale value ε / d in (27) does not affect the location of the zero-set of the polynomial and is therefore permissible. When allocating bits for each coefficient, we scale the coefficients anyway, so that the largest coefficient equals 1. Since most often scaling is involved, we simplify notation by ignoring the constant $-\varepsilon / d$ in (27).

[3] This needs to be the absolute value of the gradient. However, the sign of the gradient should remain constant at all the data points.

4.2 Improved fitting algorithms

In order to obtain an optimal solution to the fitting problem one needs first to define the criterion according to which the optimization process would be carried out.

We are interested in obtaining a coefficient vector $\overline{a}$ that produces a polynomial with two properties: a) best fit to the data ($P_{\overline{a}}(x_n, y_n) = 0$), b) minimal deviation due to changes in the coefficients.

Using the error bound in (18) with a maximum coefficients error of $\delta_{\max}$ we look for a polynomial which minimizes $\dfrac{\delta_{\max}\sum_{k=1}^{r}\left|p_k(x_n, y_n)\right|}{\left\|\nabla P_{\overline{a}}(x_n, y_n)\right\|}$ for each of the data set points.

The first requirement also implies that the tangent direction of the polynomial equals the tangent direction of the data for each of the data set points. This is due to the fact that the gradient of the polynomial is perpendicular to the zero-set. This leads to the requirement:

$$\frac{\partial P_{\overline{a}}(x_n, y_n)}{\partial y} \Bigg/ \frac{\partial P_{\overline{a}}(x_n, y_n)}{\partial x} = tg(\alpha_n) \qquad (29)$$

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 .5.

We denote the coefficient vector that best achieves these requirements as $\overline{a}_{OPT}$.

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 $\dfrac{\delta_{\max}\sum_{k=1}^{r}\left|p_k(x_n, y_n)\right|}{\left\|\nabla P_{\overline{a}}(x, y)\right\|}$ would be the same for all the given data points, i.e., for $n = 1,\dots,N$. Since the value of this constant does not affect the optimization, we require the following:

$$\left\|\nabla P_{\bar{a}}(x,y)\right\| = \sum_{k=1}^{r}\left|p_k(x_n,y_n)\right| \quad for\ n = 1,\dots,N \tag{30}$$

The LS solution for the requirements presented above can be calculated within the framework of the solution described in section 3.1. Modifying (24) according to the requirement in (30) yields:

$$\left.\begin{array}{l} \dfrac{v_n}{w_n} = tg(\alpha_n) \\ \sqrt{\left(v_n^{\,2} + w_n^{\,2}\right)} = \sum_{k=1}^{r}\left|p_k(x_n,y_n)\right| \end{array}\right\} \Rightarrow \begin{array}{l} v_n = \left(\sum_{k=1}^{r}\left|p_k(x_n,y_n)\right|\right)\sin(\alpha_n) \\ w_n = \left(\sum_{k=1}^{r}\left|p_k(x_n,y_n)\right|\right)\cos(\alpha_n) \end{array} \tag{31}$$

which upon substitution into (25) 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 error. Modifying (24) as shown in (32) and substituting in (25) yields a solution that we denote as the *Min-Var* solution.

$$\left.\begin{array}{l} \dfrac{v_n}{w_n} = tg(\alpha_n) \\ \sqrt{\left(v_n^{\,2} + w_n^{\,2}\right)} = \sqrt{\sum_{k=1}^{r} p_k^{\,2}(x_n,y_n)} \end{array}\right\} \Rightarrow \begin{array}{l} v_n = \sqrt{\sum_{k=1}^{r} p_k^{\,2}(x_n,y_n)}\sin(\alpha_n) \\ w_n = \sqrt{\sum_{k=1}^{r} p_k^{\,2}(x_n,y_n)}\cos(\alpha_n) \end{array} \tag{32}$$

4.2.1 Adding weights to data points

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 cost function:

$$E_W = \bar{e}_W \bar{e}_W^{T} \tag{33}$$

where, generalizing (6),

$$\bar{e}_W = \left(\bar{a}M - \bar{b}\right)W = \bar{a}MW - \bar{b}\,W \tag{34}$$

Here W is a $3N \times 3N$ *diagonal* weighting matrix whose diagonal elements are the relative weights. I.e., the diagonal elements $1,\dots,N$ are the weights for the zero fitting errors, and elements $N+1,\dots,2N$ and $2N+1,\dots,3N$ are the weights for the sensitivity

Replacing $\overline{a}M$ by $\overline{a}MW$ and $\overline{b}$ by $\overline{b}W$, the LS solution in (9) becomes:

$$\underline{a}_{WLS} = \overline{b}WW^TM^T\left(MWW^TM^T\right)^{-1} = \overline{b}W^2M^T\left(MW^2M^T\right)^{-1} \qquad (35)$$

The selection of the weights depends on the application and fitting goal. If some points have priority over others, than both the zero values and gradients of these points should receive greater weight values than those of other points. E.g., if point m has priority over point l than

$$\begin{aligned} &W(m,m) > W(l,l) \\ &W(m+N,m+N) > W(l+N,l+N) \\ &W(m+2N,m+2N) > W(l+2N,l+2N) \end{aligned} \qquad (36)$$

However, the fitting can also be optimized to produce global characteristics. Choosing large values for the first N elements, gives preference to a good fit with less regard to the sensitivity, while choosing large values for the second and third N elements gives preference to a stable fit on the expense of its tightness.

4.3 Extension to 3D fitting

The algorithms presented above for 2D curves can be extended to 3D space for fitting surfaces of 3D bodies.

The development of the fitting algorithms for 3D bodies follows the procedure presented above, with the following modifications:

- All coordinates are given in 3D.
- Perpendicular vectors are now calculated as normals to tangent surfaces (instead of normals to lines).

The LS solution for the 3D fitting problem has still the form in (9), i.e.,

$$\overline{a}_{LS} = \overline{b}M^T\left(MM^T\right)^{-1} \qquad (37)$$

but here,

$$\begin{aligned}\bar{b} &= [\bar{0} \quad \bar{v} \quad \bar{w} \quad \bar{q}] \\ M &= [M_0 \quad M_X \quad M_Y \quad M_Z]\end{aligned} \tag{38}$$

where,

$$\begin{aligned}M_0 &= [\bar{p}^T(x_1, y_1, z_1) \ \dots \ \bar{p}^T(x_N, y_N, z_N)] \\ M_X &= [\bar{p}_X{}^T(x_1, y_1, z_1) \ \dots \ \bar{p}_X{}^T(x_N, y_N, z_N)] \\ M_Y &= [\bar{p}_Y{}^T(x_1, y_1, z_1) \ \dots \ \bar{p}_Y{}^T(x_N, y_N, z_N)] \\ M_Z &= [\bar{p}_Z{}^T(x_1, y_1, z_1) \ \dots \ \bar{p}_Z{}^T(x_N, y_N, z_N)]\end{aligned} \tag{39}$$

$$[v_n \quad w_n \quad q_n]^T = g_n \cdot \bar{u}_n \tag{40}$$

where $\bar{u}_n$ is a unit vector locally perpendicular[4] to the surface near data point n. This vector can be computed by fitting a 1st order 3D polynomial (a plane) to the points about point n. The coefficient vector of this 1st order polynomial is in fact the desired vector $\bar{u}_n$.

g_n is calculated according to the selected fitting algorithm:

$$\begin{aligned}&\text{For } 3L: && g_n = 1 \\ &\text{For } Min\text{-}Max: && g_n = \sum_{k=1}^{r} |p_k(x_n, y_n, z_n)| \\ &\text{For } Min-Var: && g_n = \sqrt{\sum_{k=1}^{r} p_k{}^2(x_n, y_n, z_n)}\end{aligned} \tag{41}$$

The monomial differentials $\bar{p}_X(x_n, y_n, z_n)$, $\bar{p}_Y(x_n, y_n, z_n)$ are as defined for the 2D case and $\bar{p}_Z(x_n, y_n, z_n)$ is the differential of the monomial vector in the Z direction for data point n.

4.4 Summary of fitting algorithms

The following tables summarize the different fitting algorithms presented in this section, for fitting both 2D curves and and 3D surfaces.

[4] According to (2), the monomial vector is perpendicular to the zero set of the polynomial.

Table I: Summary of 2D *Min-Max* and *Min-Var* fitting algorithms

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

Table II: Summary of 3D *Min-Max* and *Min-Var* fitting algorithms

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

5 Simulation Results

In this section we present simulation results of the 3L fitting-algorithm and of the proposed *Min-Max* and *Min-Var* algorithms, derived in section 4. Further detail can be found in [19].

5.1 Sensitivity to coefficient quantization

In this subsection we examine by simulations the sensitivity of the algorithms presented here to coefficient changes. Quantization of the coefficients to the closest integer gives only a single instance of the coefficient error vector and does not allow a statistical analysis of the error properties due to quantization errors. We use, therefore, uniformly distributed random noise to simulate the effects of quantization. The results of this test would indicate which algorithm yields more robust fitting results under coefficient quantization.

An object whose boundary is shown in Fig. 6, comprising of 204 data points, was used to test the 3L*, Min-Var,* and *Min-Max* algorithms.

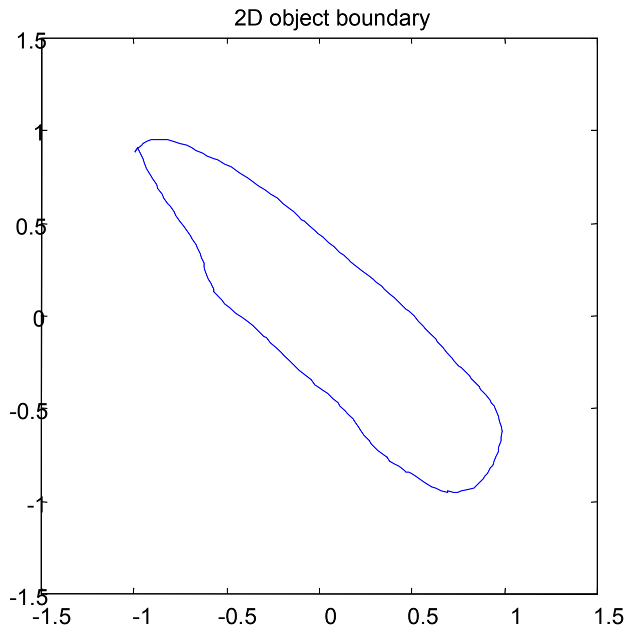

Fig. 6: Object boundary used in simulations

Tables III(a) and III(b) compare the errors obtained using the examined fitting algorithms for the boundaries of the tested object, quantized with several different number of bits per coefficient. The same random noise, having a uniform distribution in the range corresponding to the least significant bit (LSB) in the binary representation of the coefficients, was added to the coefficient vectors obtained via each fitting algorithm. 500 independent error vectors were examined in this test. For each algorithm two error measures were considered:

$$E_{RMS} \triangleq \sqrt{\sum_{n=1}^{N} e_n^2}\ ; \quad E_{MAX} \triangleq \max\{e_1, \ldots, e_N\} \tag{42}$$

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).

The mean value and variance (over the different noise vectors) of these error measures are shown in Table III(a) and Table III(b).

Table III: Comparison of error statistics for the *3L*, *Min-Var,* and *Min-Max* fitting algorithms

(a) 12th order polynomial and 16bits coefficients

Algorithm →	*3L*		*Min-Var*		*Min-Max*	
Error Measrure ↓	Mean	Variance	Mean	Variance	Mean	Variance
E_{RMS}	0.04	0.034	0.02	***0.003***	***0.019***	0.0033
E_{MAX}	0.15	0.22	0.046	***0.007***	***0.044***	0.015

(b) 4th order polynomial and 8bits coefficients

Algorithm →	*3L*		*Min-Var*		*Min-Max*	
Error Measure ↓	Mean	Variance	Mean	Variance	Mean	Variance
E_{RMS}	0.0415	0.0021	0.0352	***0.0017***	***0.0317***	0.0017
E_{MAX}	0.1132	0.0258	0.0979	***0.0232***	***0.0795***	0.0348

(c) 4th order polynomial and 6bits coefficients

Algorithm →	*3L*		*Min-Var*		*Min-Max*	
Error Measure ↓	Mean	Variance	Mean	Variance	Mean	Variance
E_{RMS}	0.042	0.001	0.035	***0.0007***	***0.032***	0.0008
E_{MAX}	0.11	0.007	0.097	***0.0063***	***0.079***	0.0088

By examining the above three tables, it is evident that using different fitting criteria results in different performance. The original 3L algorithm does not define any stability goal and therefore achieves no best score in any of the tests. The mean error is minimal when the *Min-Max* algorithm is used. This result is expected because the fitting criterion demands the lowest maximal error and therefore also leads to a minimal mean error. Accordingly, the error variance for both objects is lowest for the *Min-Var* algorithm. This result is expected because the criterion minimizes this quantity.

5.2 Plots of sensitivity function

In this subsection the behavior of the sensitivity function obtained by the different algorithms examined above is graphically demonstrated. The boundary shown in Fig. 6 was used in the simulations of this section. We use a relatively simple shape as it allows a clear illustration of the fitting errors characteristics due to coefficient quantization.

The values of the sensitivity function for each point of the data set was calculated, using the polynomials obtained by the different fitting algorithms examined. The fitting results by 8th order polynomials and the values of the sensitivity function are plotted in Fig. 7(a1, b1, c1). The sensitivity function determines the change in the location of the zero-set, in the direction perpendicular to the desired zero-set, due to changes in the coefficients. Therefore, the values of the sensitivity function are plotted in Fig. 7 as vectors that are perpendicular to the zero-set, with a length proportional to the value of the function. For plotting purposes the values of the sensitivity function were normalized (divided by an appropriate fixed number). As seen in Fig. 7, all three algorithms produced good fitting results when the coefficients are not quantized. This is because the specific object boundary is relatively simple for an 8th order polynomial. Yet, it is clear from the results demonstrated in Fig. 7 that the 3L algorithm is the most sensitive of the 3 examined algorithms and that the *Min-Max* algorithm obtains the best results.

5.3 Fitting a 3D surface

Fig. 8 shows two faces of a 3D object (left column – front view; right column – back view), with the top row displaying the original object, and the next rows - the fitting results with the 3L, *Min-Var* and *Min-Max*, algorithms, respectively. The improvement in fitting over the 3L algorithm, especially by the Min-Max algorithm is clearly seen. This is also manifested by the root-mean-squared (RMS) values of the distances between the input data points and the zero set. The values obtained are: 3.67, 3.5, and 1.91, for the above examined three algorithms, respectively.

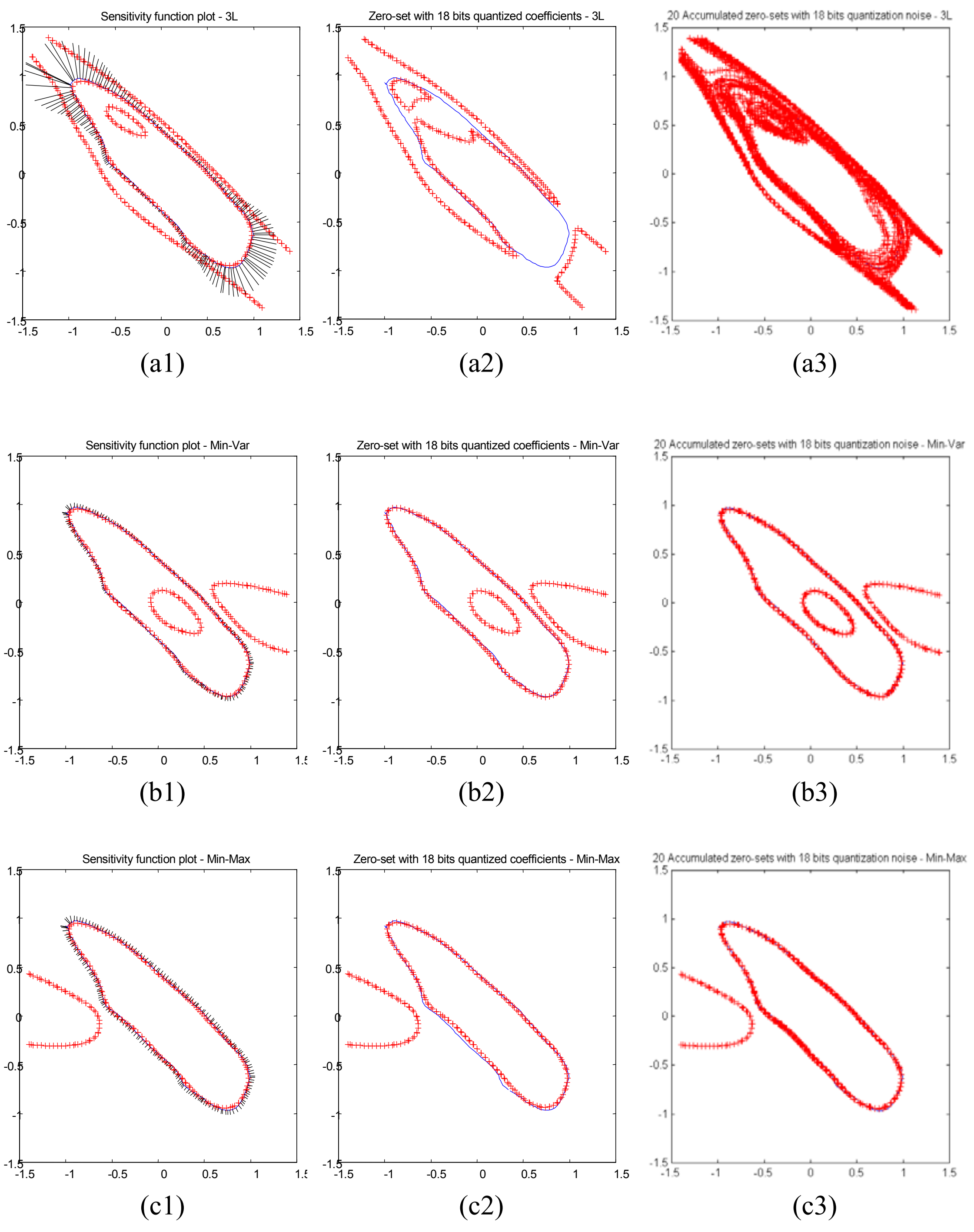

Fig. 7: Sensitivity and fitting errors of polynomial zero-sets for the *3L, Min-Var,* and *Min-Max* algorithms: (a) Upper row – *3L* algorithm, (b) Middle row – *Min-Var* algorithm, (c) Lower row – *Min-Max* algorithm.

Left column (1) – Coefficients are not quantized.

Center column (2) –18 bits coefficient quantization.

Right column (3) – 20 accumulated zero-sets with noise added to coefficients.

Noise has independent samples from a uniform distribution with a variance of 1.1e-6 (LSB in 18 bits).

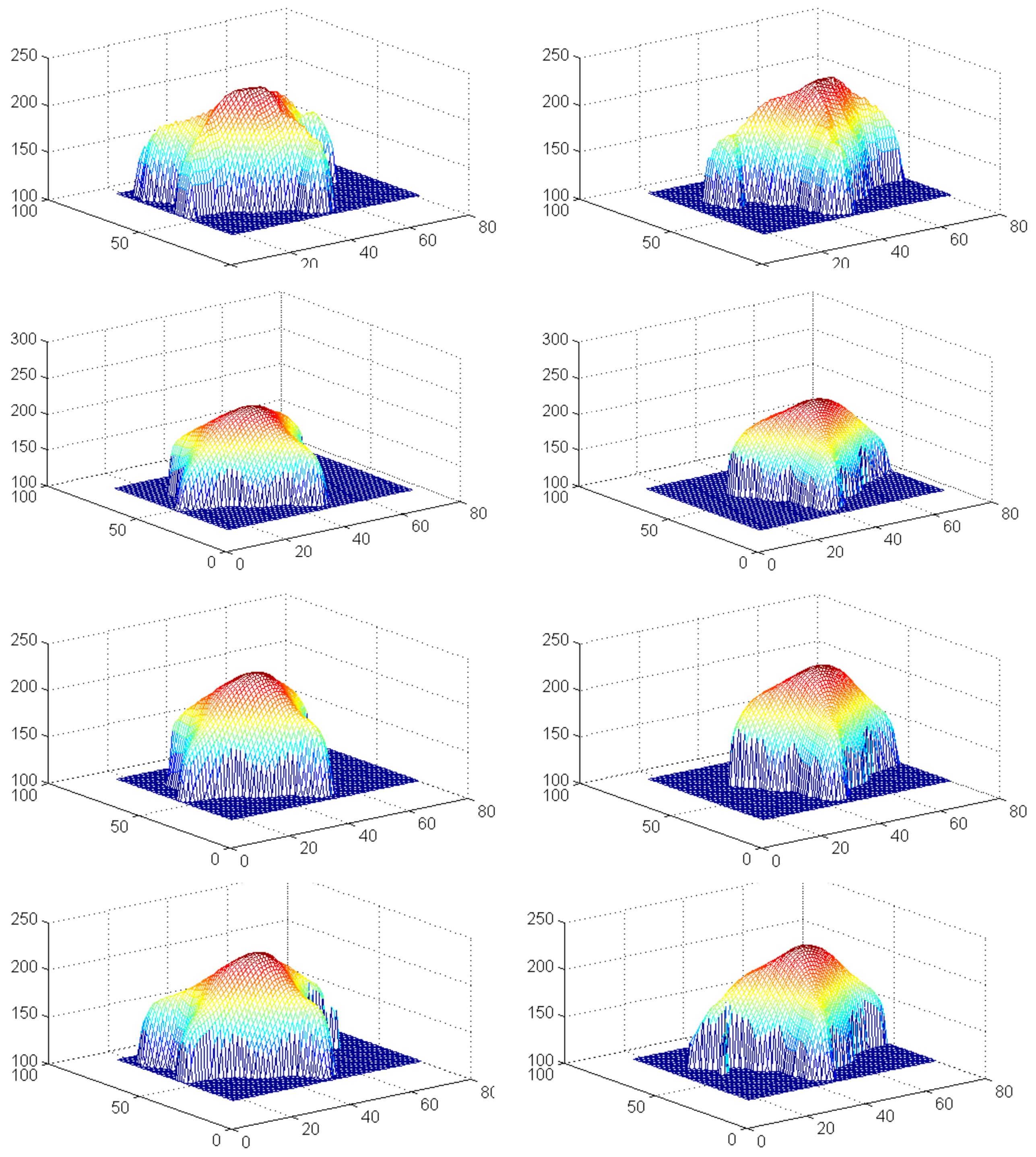

Fig. 8: 3D Fitting results: Left – Shape from front, Right – Shape from back.
Top row – Original data; Second row – fitting results with *3L* algorithm; Third row – fitting results with *Min-Var* algorithm; Bottom row – fitting results with *Min-Max* algorithm

6 Summary and conclusions

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

An application of implicit polynomials to contour coding had brought the subject of sensitivity to coefficient quantization to our attention. We began our development with a study of the sensitivity of the zero-set of implicit polynomials to coefficient quantization obtained by existing fitting algorithms. The analysis that we provide explains the differences in stability between the examined fitting algorithms and shows why the 3L algorithm out-performs earlier algorithms. We provide a numerical tool, rather than just an intuitive explanation.

A conclusion from the sensitivity analysis is that the 3L fitting algorithm could be still greatly improved, in terms of fitting tightness and stability, by adding stability considerations to the fitting process. In this paper we show the steps in the development of the proposed fitting algorithms, aimed at producing implicit polynomials that are tight about the data set and robust to coefficient quantization.

We present two algorithms, denoted *Min-Max* and *Min-Var* fitting algorithms. The two algorithms exhibit similar performance. The first, the *Min-Max* algorithm, minimizes the maximal error due to coefficient quantization. The second algorithm, *Min-Var*, minimizes the error variance due to coefficient errors. Which algorithm should be used depends on the application. For example, in contour coding applications, where the coefficients are quantized and the maximal displacement of the zero-set should be minimized, the *Min-Max* algorithm is more effective. We also show how the proposed algorithms can be modified to support fitting of 3D data.

Although the development of the proposed fitting algorithm was motivated by an image coding application, it was found to achieve better fitting than the 3L algorithm, even when the coefficients are not quantized. We have demonstrated that uniform sensitivity to coefficient errors for all the data points, leads to improved fitting. The proposed fitting algorithms could therefore have an advantage in all the applications that use implicit polynomial fitting, such as object recognition, contour coding, computer graphics, CAD and others.

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, Berlin, 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” IAPR International Conference on Pattern Recognition, Jerusalem, pp. Vol. 1, pp. 205-209,October 1994.

[7] Jean-Philippe Trael, David 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 Engineers in Israel, Tel-Aviv, Israel, pp. 384-388, April 2000.

[9] Tolga Tasdizen, David B. Cooper: Boundary Estimation from Intensity/Color Images with Algebraic Curve Models. Int’l Confernce Pattern Recognition (ICPR), 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] Z. Lei, M.M. Blane, and D.B. Cooper, “3L Fitting of Higher Degree Implicit Polynomials”, Technical Report LEMS TR-160, Brown University LEMS Lab., 1997.

[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 Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 3, March 2000.

[14] Z. Lei and D.B. Cooper, “New, Faster, More Controlled Fitting of Implicit Polynomial 2D Curves and 3D Surfaces to Data”, IEEE Conference on Computer vision and Pattern Recognition, (San Francisco), 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] Daniel Keren, Craig 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. On Image Processing, Vol. 9, No. 3, pp. 405-416, March 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. on Pattern Recognition, Barcelona, pp. 290-293, Sept. 2000.

[19] A. Helzer “Using Implicit Polynomials for contour coding”, M.Sc Thesis, Technion – Israel Institute of Technology, Haifa, Israel, Dec. 2000.
Available at: http://www-sipl.technion.ac.il/publications/thesis/helzer/helzer-thesis.pdf.