Robust Fitting of Implicit Polynomials with Quantized Coefficients to 2D Data

Amir Helzer Meir Barzohar[1] David Malah
Department of Electrical Engineering, Technion, Haifa 32000, ISRAEL
eehelzer@techunix.technion.ac.il

Abstract

This work presents a new approach to contour representation and coding. It consists of an improved fitting of high-degree (4^{th} to 18^{th}) implicit polynomials (IPs) to the contour, which is robust to coefficient quantization. The proposed approach to solve the fitting problem is a modification of the 3L *linear solution developed by Lei et al and is more robust to noise and to coefficient quantization. We use an analytic approach to limit the maximal fitting error between each data point and the zero-set generated by the quantized polynomial coefficients. We than show that consideration of the quntization error (which led to a specific sensitivity criterion) also brought about a significant improvement in fitting IPs to noisy data, as compared to the* 3L *algorithm.*

1. Introduction

Implicit polynomials (IPs) are global models with good representation power of complex object boundaries in 2D images and surfaces in 3D range data [1,3,4]. Their power of discrimination by using euclidean and affine invariants has made them attractive for object recognition [2,5]. This work deals with fitting an IP of degree up to eighteen, whose zero-set is a geometric model for an object boundary. This model can be used in the context of a general scheme for contour-based coding. IPs should be used where the rate required for contour coding is critical and the complexity involved in polynomial fitting is justified. This work takes into account the effect of quantization used to compress the resulting polynomial coefficients from real numbers to integers (for example, 8 bits for each coefficient). The zero-set fitting generated by the quantized coefficients is the reconstructed data after the compression process [7]. The proposed algorithm for fitting is robust to noise and to coefficient quantization. Our approach originates from a sensitivity criterion, which leads to a derivation of tight error bounds, and to a fitting algorithm that focuses on minimizing these bounds – yielding a min-max solution to the fitting problem. The sensitivity function, describing the relation between the value changes of the polynomial coefficients (quantization effect) and the location changes of the zero-set, allows us to calculate the fitting error caused by the quantization effect for each data point. The sensitivity function has to have the same value at each of the data points in order to minimize the maximal fitting error and to reduce large local errors, which may appear in the original *3L* fitting algorithm. We have based our development on the *3L* algorithm [4], but made necessary changes to improve it. Our conclusion is that the *3L* algorithm can be improved in terms of a tighter fit and robustness to coefficients quantization, as presented in this paper.

The layout of this paper is as follows: Section 2 poses the fitting problem. Section 3 presents a study of the sensitivity of the zero-set relative to changes in the coefficients of the polynomial, and the resulting error properties. In Section 4 we construct our fitting algorithm, based on the *3L* algorithm and the results of Section 3. Section 5 provides simulation results and Section 6 summarizes the paper.

2. Fitting Problem

The object of polynomial fitting is to describe data points (object boundary for 2D objects or surfaces for 3D objects) with the zero-set of a polynomial. Therefore, we would like the value of the polynomial to 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 d^{th} degree polynomial, the monomial vector is denoted as:

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

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:

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

[1] Meir Barzohar is also with the Computer Vision Group in Rafael, Haifa, Israel

0-7695-0750-6/00 $10.00 © 2000 IEEE

We are looking for a parameter vector $\bar{a}$ that leads to a polynomial that best fits the data under a criterion to be specified.

3. Zero set sensitivity to parameter changes

When the values of the coefficients in the parameter vector change, the entire zero-set changes. In this Section we examine how changes in the parameter vector affect the location of a point on the zero-set. Since the zero-set is continuous, we cannot measure the distance between two points (before and after a parameter change). The location of a point on the zero-set can be defined as $\bar{z} = (z_t, z_u)$ where z_t is the component locally tangent to the zero-set, and z_u is the component locally perpendicular to the zero-set (see Fig.1). We denote the angle of the perpendicular relative to the x axis as α.

Small changes in the tangent direction move zero-set points back into the zero-set, therefore for the purpose of evaluating changes in the zero-set, it is sufficient to examine changes in the component z_u only. We define the change in a zero-set point ($du(x,y)$) as the perpendicular component of the distance between the original zero-set point and the closest point on the new data set (following the change in the parameter vector).

We can define the perpendicular component of the change in the location of a zero-set point $\varepsilon_u(x,y)$, resulting from a change in the parameters $\bar{\varepsilon}_a = [\varepsilon_{a_1} \ \ldots \ \varepsilon_{a_r}]$ as the product of the error components with the associated sensitivity function:

$$\varepsilon_u(x,y) = \bar{S}_{\bar{a}}^{u}(x,y)\bar{\varepsilon}_a^{\,T} \qquad (3)$$

where the sensitivity function $\bar{S}_{\bar{a}}^{u}(x,y)$ is a vector defined by:

$$S_{\bar{a}}^{u}(x,u) = \frac{du(x,y)}{d\bar{a}} \qquad (4)$$

3.1. Sensitivity function evaluation

The sensitivity function (4) can be written as a product of two independent parts:

$$\bar{S}_{\bar{a}}^{u}(x,y) = \frac{du(x,y)}{dP_{\bar{a}}(x,y)} \frac{dP_{\bar{a}}(x,y)}{d\bar{a}} \qquad (5)$$

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

The right-hand part of the product in (5) describes the change in the value of the polynomial, at the location of the original zero-set point, 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 of the product describes 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 the location of the original zero-set point. This part is a scalar.

We shall evaluate each part of the product in (5) separately, beginning with the left-hand part.

The deviation in the location of a zero-set point in a direction locally perpendicular to the zero-set, following a change in the parameter vector, is described in Fig. 1.

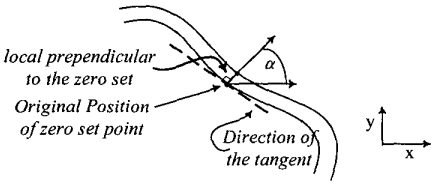

Fig. 1. – Location of a zero-set point before and after a small change in the coefficients

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

$$\frac{du}{dP_{\bar{a}}}(x,y) = \frac{1}{\|\nabla P_{\bar{a}}(x,y)\|} = \frac{1}{\sqrt{\left(\frac{\partial P_{\bar{a}}(x,y)}{\partial x}\right)^2 + \left(\frac{\partial P_{\bar{a}}(x,y)}{\partial y}\right)^2}} \qquad (6)$$

The right-hand part of the product in (5) can be directly calculated from (2), and results in:

$$\frac{dP_{\bar{a}}(x,y)}{d\bar{a}} = \frac{d\left(\bar{a}\,\bar{p}^{T}(x,y)\right)}{d\bar{a}} = \bar{p}(x,y) \qquad (7)$$

Using (5), (6) and (7) the sensitivity function in (4) can now be written as:

$$\bar{S}_{\bar{a}}^{u}(x,y) = \frac{\bar{p}(x,y)}{\|\nabla P_{\bar{a}}(x,y)\|} \qquad (8)$$

3.2. Zero-set fitting errors

Now, we can examine the resulting fitting errors due to small changes in the coefficients and derive some useful bounds on these errors.

Using (3) and (8), the error in the direction perpendicular to the zero-set, $\varepsilon_u(x,y)$, is:

$$\varepsilon_u(x,y) = \bar{S}_{\bar{a}}^{u}(x,y)\cdot\bar{\varepsilon}_a^{\,T} = \frac{\sum_{k=1}^{r} p_k(x,y)\varepsilon_{a_k}}{\|\nabla P_{\bar{a}}(x,y)\|} \qquad (9)$$

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

$$|\varepsilon_u(x,y)| \le \frac{\left|\sum_{k=1}^{r} p_k(x,y)\varepsilon_{a_k}\right|}{\|\nabla P_{\bar{a}}(x,y)\|} \le \varepsilon_{MAX} \frac{\sum_{k=1}^{r} |p_k(x,y)|}{\|\nabla P_{\bar{a}}(x,y)\|} \qquad (10)$$

where $\varepsilon_{MAX} = \max\{|\varepsilon_{a_k}|\}$.

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

$$\mathrm{var}(\varepsilon_u(x,y)) = \frac{\sum_{k=1}^{r}\left((p_k(x,y))^2\ \mathrm{var}(\varepsilon_{a_k})\right)}{\|\nabla P_{\bar{a}}(x,y)\|^2} \qquad (11)$$

and when all the error components have the same variance ($\mathrm{var}(\varepsilon_K)=\sigma_\varepsilon^2$), we obtain:

$$\sqrt{\mathrm{var}(\varepsilon_u(x,y))} = \sigma_\varepsilon \frac{\sqrt{\sum_{k=1}^{r} p_k^{\,2}(x,y)}}{\|\nabla P_{\bar{a}}(x,y)\|} \qquad (12)$$

4. A robust fitting algorithm

In Section 3 we analyzed the changes in the location of zero-set points resulting from changes in the parameters. We would like the zero-set to remain as close as possible to the original data set points. If we consider the resulting parameter vector as the sum of a locally optimal parameter vector and some added error, we can look at each data-set point as being a point on the zero-set of the locally optimal parameter vector. Therefore, we can use the results of the above analysis by substituting the zero-set points by the given data-set points to evaluate the expected fitting error.

Reference [6] includes a complete description of the formulation of the fitting algorithm. Here we present the highlights of this formulation.

In order to obtain the best parameter vector we need to define a criterion according to which we would perform the optimization process.

We are interested in two properties – a) best fit to the data, b) minimal deviation due to changes in the coefficients. From the results of Section 3 we conclude that the deviation of the zero-set of the polynomial due to changes in the coefficients is governed by the ratio between some operand on the value of the monomial vector and the gradient of the polynomial (10), (12). The bound on the maximal error, or on the variance depends on that operand. To obtain a best fit and to minimize this deviation, we construct a cost function e_n, which is minimized when the value of the polynomial and the sensitivity function at the location of data-set point (x_n, y_n) is minimal. The sum of squared errors is used as a global cost function - E. Therefore, a coefficient vector that minimizes E leads to an optimal fit, under the criterion defined by the selected bound on the sensitivity function.

$$E = \bar{e}\,\bar{e}^T \qquad (13)$$

where e_n is the error at data point $(x_n, y_n), n = 1,2,...,N$, and the vector $\bar{e} = (e_1,...,e_N)$ is calculated by:

$$\bar{e} = (\bar{a}M - \bar{b}) \qquad (14)$$

with the following definitions:

$$\bar{b} = [\bar{0}\ \ \overline{dx}\ \ \overline{dy}] \qquad M = [M_0\ \ M_X\ \ M_Y] \qquad (15)$$

where $\overline{dx}, \overline{dy}$ are vectors defining the required differentials (see below) of the polynomial at the data points and,

$$M_0 = [\bar{p}^T(x_1,y_1)\ \ ...\ \ \bar{p}^T(x_N,y_N)]$$
$$M_X = [\bar{p}_X{}^T(x_1,y_1)\ \ ...\ \ \bar{p}_X{}^T(x_N,y_N)] \qquad (16)$$
$$M_Y = [\bar{p}_Y{}^T(x_1,y_1)\ \ ...\ \ \bar{p}_Y{}^T(x_N,y_N)]$$

with,

$$\bar{p}_X(x_n,y_n) = \frac{d}{dx}\bar{p}(x_n,y_n)\ ;\ \bar{p}_Y(x_n,y_n) = \frac{d}{dy}\bar{p}(x_n,y_n) \qquad (17)$$

According to [6], the vectors $\overline{dx}$ and $\overline{dy}$ in (15) characterize the fitting algorithm. To obtain a fitting algorithm which minimize the max error (*Min-Max*), $\overline{dx} = [dx_1,..., dx_N]$ and $\overline{dy} = [dy_1,..., dy_N]$ should satisfy:

$$\frac{dy_n}{dx_n} = tg(\alpha_n)$$
$$\sqrt{(dx_n^{\,2} + dy_n^{\,2})} = \sum_{k=1}^{r} |p_k(x_n,y_n)| \qquad (18)$$

where α_n is the angle of the local perpendicular to the data-set around point (x_n, y_n) relative to the x axis, as shown in Fig 1.

Using the same formulation, we can implement *a* fitting algorithm, which minimize the variance (*Min-Var*) of the error (over the elements of the error components). This is done by requiring $\overline{dx}$ and $\overline{dy}$ to satisfy:

$$\frac{dy_n}{dx_n} = tg(\alpha_n)$$
$$\sqrt{(dx_n^{\,2} + dy_n^{\,2})} = \sqrt{\sum_{k=1}^{r} p_k^{\,2}(x_n,y_n)} \qquad (19)$$

Since no data point has priority over any another point (if no weighting is used), we can limit the maximal fitting error due to changes in the coefficients to a given value ε_{MAX} (see (10)) by requiring that:

$$\|\nabla P_{\bar{a}}(x,y)\| = \sum_{k=1}^{r} |p_k(x_n,y_n)| \ \ for\ n = 1,...,N \qquad (20)$$

The coefficient vector that minimizes E in (13) is denoted as $\bar{a}_{OPT}$. We obtain $\bar{a}_{OPT}$ by evaluating:

$$\bar{a}_{OPT} = \bar{b}M^T\left(MM^T\right)^{-1} \qquad (21)$$

where $\bar{b}, M$ are defined in (15).

5. Simulation results

In this Section we present simulation results of the original *3L* fitting algorithm in comparison with the *Min-Max* and *Min-Var* algorithms described in Section 4.

The data used here is taken from segmented medical images.

5.1. Sensitivity analysis

Here we examine the sensitivity of the algorithms presented in this paper to coefficient changes. We add uniformly distributed random noise to the coefficients to simulate the effect of quantization.

Tables 1(a), 1(b) compare the errors obtained using the examined fitting algorithms for the boundaries of two different objects, each quantized with a different number of bits per coefficient.

The same random noise was added to the coefficient vectors obtained via each fitting algorithm. For each algorithm two error criteria were examined:

$$E_{RMS} = \sqrt{\sum_{n=1}^{N} e_n^2}\ ,\ E_{MAX} = \max\{e_1,...,e_n\} \qquad (22)$$

The mean value and variance (over different noise vectors) of these tests are shown in Tables 1(a), 1(b).

(a) **TABLE 1**

Algorithm =>	*3L*		*Min-Max*		*Min-Var*	
Criteria	Mean	Var	Mean	Var	Mean	Var
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)

Algorithm =>	*3L*		*Min-Max*		*Min-Var*	
Criteria	Mean	Var	Mean	Var	Mean	Var
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

(a): Order – 12, Bits – 16 (b): Order – 4, Bits – 6

The original *3L* algorithm does not define any sensitivity goal and therefore achieves no best score in any test.

For both objects, the mean error is the minimum when using the *Min-Max* algorithm, and the error variance is minimal when using the *Min-Var* algorithm.

5.3. Unquantized fitting

We present in this subsection the comparison of fitting two objects using the *3L* algorithm and the proposed *Min-Max* algorithm. In each case the fitting was done with 14^{th} degree polynomials. The fitting results of two different boundaries are shown in Fig 2.

It is possible to see the significant improvement in the fitting results going from the left column (*3L*) to the right column (*Min-Max*).

Polynomials of high degree (here 14^{th}) exhibit high sensitivity to noise. The shown examples clearly demonstrate the effect of using a sensitivity criterion on the quality of the solution.

6. Conclusion

To summarize, we conclude that the fitting algorithm presented in this paper can generate implicit polynomials that well represent object boundaries. This representation can be useful for a variety of applications, among which, are object recognition and, with the addition of constraints for avoiding spurious zero-set points [6,7], also contour coding of animated objects.

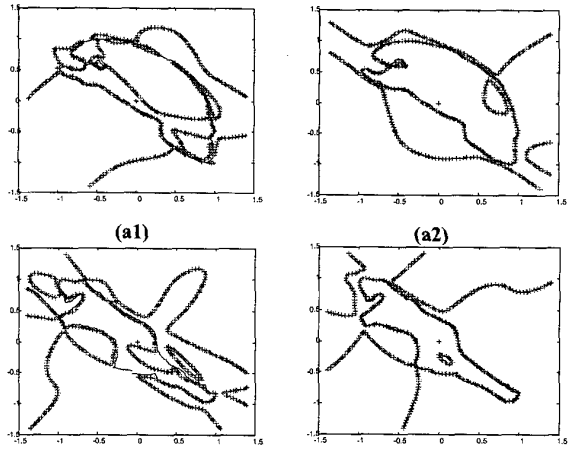

Fig. 2 – Fitting results: 14^{th} order polynomials
Left – *3L* algorithm, right – *Min-Max* algorithm
Continuous line displays original data.
Crosses display zero-set of the polynomial.

7. References

[1] D. Keren, D. Cooper, and J. Subrahmonia, "Describing Complicated Objects by Implicit Polynomials", IEEE Trans. PAMI, Vol.16, No.1, pp.38-53, Jan. 1994.

[2] J. Subrahmonia, D. Cooper, and D. Keren, "Practical Reliable Bayesian Recognition of 2D and 3D Objects Using Implicit Polynomials and Algebraic Invariants", IEEE Trans. PAMI, Vol. 18, pp.505-519, 1996.

[3] 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. PAMI, Vol. 13, No. 11, pp.1,115-1,138, Nov. 1991.

[4] 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.

[5] 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 ICPR, (Jerusalem, Israel), October 1994.

[6] A. Helzer "Representation of Object Boundaries in Images for Efficient Coding", M.Sc. Thesis, Technion, in preperation.

[7] A. Helzer, M. Barzohar, D. Malah "Using Implicit Polynomials for Image Compression", Proc. 21^{st} IEEE Convention of Electrical and Electronic Eng. in Israel, Tel-Aviv, pp 384-388, April 2000.