*IRWIN AND JOAN JACOBS*
*CENTER FOR COMMUNICATION AND INFORMATION TECHNOLOGIES*

# 2D Object Description and Recognition Based on Contour Matching by Implicit Polynomials

**Zoya Landa, David Malah and Meir Barzohar**

**CCIT Report #755**
**February 2010**


*DEPARTMENT OF ELECTRICAL ENGINEERING*
*TECHNION - ISRAEL INSTITUTE OF TECHNOLOGY, HAIFA 32000, ISRAEL*

CCIT REPORT #755 February 2010



# 2D Object Description and Recognition Based on Contour Matching by Implicit Polynomials

Zoya Landa, David Malah, and Meir Barzohar

E-mail: zrybak@yahoo.com, malah@ee.technion.ac.il, meirb@visionsense.com.

**Abstract**

This work deals with 2D object description and recognition based on coefficients of implicit polynomials (IP). We first improve the description abilities of recently published Min-Max and Min-Var algorithms by replacing algebraic distances by geometric ones in the relevant cost function. We show that a recognizer based on a full set of recently published linear, quadrature, and angular rotation invariants, derived from a polynomial of a predetermined degree, has difficulties in distinguishing among objects which are either too complicated to be modeled by a polynomial of that degree or are simple and can be successfully modeled by a polynomial of a lower degree. We propose a recognition approach that is based on deriving linear rotation invariants from several polynomials of different degrees, fitted to the object shape, as well as on their fitting errors. This approach is found to considerably improve the recognition and is denoted as Multi Order (degree) and Fitting Errors Technique (MOFET). We also propose a Shape Transform, based on the Scatter Matrix of the objects' shape, which transforms each object to its "Mother Shape". The Mother Shape is unique, up to rotation, for all the objects that are related to the original shape via an Affine transform. Thus, we are able to handle affine transformed objects as well. Finally, we compare the performance of our approach with the Curvature Scale Space (CSS) method and find that it has an advantage over CSS, at about the same complexity.

**Index Terms**

Implicit polynomials, object recognition, zero-set sensitivity, curve fitting, stable fitting, affine transform.

## I. INTRODUCTION

Implicit polynomials have been exploited for quite some time for the representation and recognition of 2D data, defined by discrete points taken along its boundary [1] – [10], [14]-[17]. Over the years, different algorithms for fitting Implicit Polynomials to the data were developed. Early iterative fitting algorithms [4], [5] often failed to derive an Implicit Polynomial defining a single closed curve that fits well the boundary of the data, i.e., they were unable to obtain a reliable representation. Furthermore, due to numerical instability and high computational cost, these algorithms are able to fit the data by polynomials of relatively low degree only (usually, up to degree 4), and, therefore, fail to represent or recognize relatively complicated objects. A modification of [5], proposed in [6], solves the problem of reliable representation for star-shaped contours. Recent versions of fitting algorithms: 3L [1] and, especially, Gradient1 [2], Min-Max and Min-Var [3], apply a linear LS (Least Squares) solution to the fitting problem (having, therefore, a lower computational cost than earlier iterative algorithms), appear to have much better performance in both representation and recognition tasks. Hence, in our work we focus on the last three algorithms, namely: Gradient1, Min-Max and Min-Var.

In our work we address both representation and recognition tasks. For improved representation we propose a modification of the Min-Max and Min-Var algorithms, resulting in better modeling of the data. Instead of minimizing the values of the polynomial at the data points, i.e., the algebraic distances between the data points and the Implicit Polynomial, we minimize the approximated geometric distances between the data points and the Implicit Polynomial (IP) zero-sets [5], which is a much more adequate measure of representation quality. By using this measure we obtain a better representation of the data.

The main drawback of IP-s for recognition applications is their low performance when used with a database consisting of both simple and complicated objects. Generally, a recognizer based on a relatively low degree polynomial would be able to effectively recognize simple objects (i.e., objects satisfactorily represented by a polynomial of that low degree), but would have low performance in recognition of more complicated objects (which are badly represented by a low degree polynomial). Polynomials of a higher degree can represent faithfully any object in the database, but it turns out that in the case of simple objects the coefficients defining the IP would be extremely sensitive to even minor perturbation of the data, impeding the recognition. Hence,

we propose a recognizer that is based both on coefficients of IP-s of several degrees and on their fitting errors. We denote this technique as Multi Order (degree) and Fitting Error Technique ( MOFET ).

The ability to recognize objects given in different orientations is very important. Geometric rotation invariants, derived for IP-s in [9], makes this technique very attractive for recognition tasks. These invariants are based on the polynomial coefficients. In order to take advantage of these invariants, we should make sure that the relation between polynomials fitted to different orientations of the same data should be characterized by rotation only. However, polynomials derived by Min-Max and Min-Var algorithms don't fulfill this condition. Hence, we propose a way to modify these algorithms so that they will meet this requirement. Then, we can base the MOFET recognizer on polynomial invariants instead of polynomial coefficients.

We address in this work not only rotation but also the more general *affine* transform. In many practical cases the relation between two different projections of the same 3D object can be approximated by an affine transform. We propose a simple and efficient technique for eliminating the effect of such transform. We first transform the data in such a way that its Scatter matrix would be equal to the identity matrix. This transform is based on $1^{st}$ and $2^{nd}$ order moments of the data, and, therefore is robust to data perturbations. As a result of this transform we get what we call the "Mother shape" of the data, which is unique up to rotation for any data related affinely to the original one. We then fit the IP to the transformed data and apply the MOFET recognizer based on the rotation invariants derived from the resulting IP coefficients.

Finally, we compare the performance of the MOFET recognizer with other IP-based techniques and with the Curvature Scale Space (CSS) technique [18], which recently became part of the MPEG-7 standard.

The organization of this paper is as follows:

Section II provides background information on Implicit Polynomials and on the recently developed Min-Max and Min-Var algorithms [3], and suggests modifications to these algorithms that improve their fitting performance.

Section III reviews recently developed rotation invariant object recognition techniques that are based on IP of predefined degree, obtained by any of the IP fitting techniques described above, and discusses their properties.

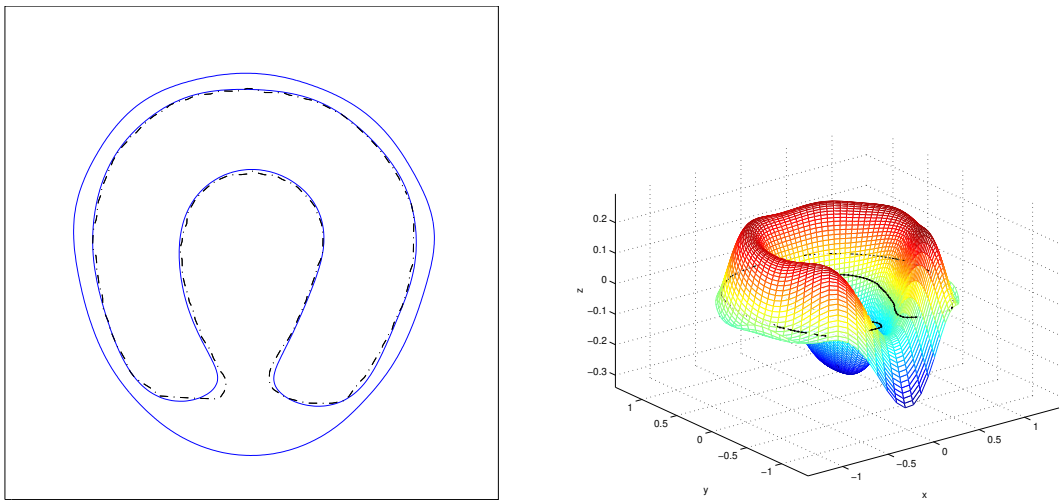
Section IV presents the proposed MOFET algorithm.

Section V introduces the Shape Transform of an object into its Mother-shape, hence allowing also recognition of data that underwent an affine transform.

Section VI compares the Mother-shape based MOFET recognizer performance with that of the CSS technique, and section VII summarizes the work and draws conclusions.

## II. Background

The ability to efficiently describe curves that represent the boundary of 2D objects is important in computer vision tasks and computer graphics. Implicit Polynomials provide a solution to this problem, using the coefficients of the polynomial to represent the data. The technique assumes that the data is a curve that is lying in the zero-plane (i.e., in 3D space, with axes x,y,z , the zero-plane is defined by z=0), and that one of the zero-sets of the polynomial (z= f(x,y) ) is supposed to fit this curve.




(a) (b)

Fig. 1. (a) - curve representing the boundary of an object (dashed line) and the zero-sets of the 4-th degree polynomial attempting to fit the object (solid). Note that there is a spurious zeros-set (outside solid curve). (b) - surface of a 4-th degree polynomial attempting to fit the object.

An example of a curve lying in the zero-plane and a polynomial describing it is shown in Fig. 1.

### A. 2D Object description by Implicit Polynomials

As will be shown in the sequel (subsection II-B), a polynomial that efficiently describes a 2D object, given by a set of points (data-set) along the object's boundary, should meet several conditions. We will start here from the obvious and most intuitive requirement: the zero-set of the fitting polynomial should be close to the data-set.

Let's define:

$\underline{x} \stackrel{\triangle}{=} [x_1 \;\; x_2 \; ... \; x_n]'$: a vector of the first coordinates of the data set, where $(\cdot)'$ denotes the transpose of $(\cdot)$;

$\underline{y} \stackrel{\triangle}{=} [y_1 \;\; y_2 \; ... \; y_n]'$: a vector of the second coordinates of the data set;

$P^r_{\underline{a}}(x,y) = a_{0,0} + a_{1,0}x + a_{0,1}y + a_{2,0}x^2 + a_{1,1}xy + a_{0,2}y^2 + \ldots + a_{r,0}x^r + a_{r-1,1}x^{r-1}y + \ldots + a_{0,r}y^r$: a polynomial of degree $r$ with coefficients-vector

$$\underline{a} \stackrel{\triangle}{=} [a_{0,0} \;\; a_{1,0} \;\; a_{0,1} \; ... \; a_{r,0} \;\; a_{r-1,1} \; ... \; a_{0,r}]'. \tag{1}$$

The above expression can then be written as:

$$P^r_{\underline{a}}(x,y) = \underline{a}'\underline{p}^r(x,y), \tag{2}$$

where,

$$\underline{p}^r(x,y) \stackrel{\triangle}{=} [1 \;\; x \;\; y \;\; x^2 \;\; xy \;\; y^2 \;\; ... \;\; x^r \;\; x^{r-1}y \;\; ... \;\; y^r]' \tag{3}$$

is a vector of monomials at point $(x,y)$.

The length of both the monomial vector and the coefficient vector is $\frac{(r+1)(r+2)}{2}$. The zero-sets of $P^r_{\underline{a}}(x,y)$ are denoted by $ZS\left(P^r_{\underline{a}}(x,y)\right) = \left\{(x,y) : P^r_{\underline{a}}(x,y) = 0\right\}$.

Note, that a polynomial may have several zero-sets, where a zero-set is defined as a continuous curve satisfying $P^r_{\underline{a}}(x,y) = 0$. Actually, a polynomial of degree $r$ may have up to $r$ zero-sets.

Thus, we would like to find a polynomial having a zero-set that best fits the given data-set.

### *B. Gradient-one, MinMax and MinVar algorithms*

In this subsection we bring a brief overview of the recent fitting algorithms that are used in the sequel. These algorithms are based on a simple linear LS solution, generally produce a single zero-set that is close to the data-set (instead of two or more zero-sets fitting the data-set), and are numerically stable even when using polynomials of relatively high degree (in our work we used up to an 8-th degree polynomials).

#### 1. **Gradient-one algorithm**

In order to produce a single zero-set resembling the data-set, the Gradient-one algorithm [2] attempts to fulfill several conditions. the first condition is to minimize the algebraic distance between the polynomial and the data-set: i.e.,

$$\min_{\underline{a}} \sum_{k=1}^{n} P_{\underline{a}}^{r}(x_k, y_k)^2 \tag{4}$$

or

$$\min_{\underline{a}} \sum_{k=1}^{n} (\underline{a}' \underline{p}^{r}(x_k, y_k))^2, \tag{5}$$

which can be put in the form

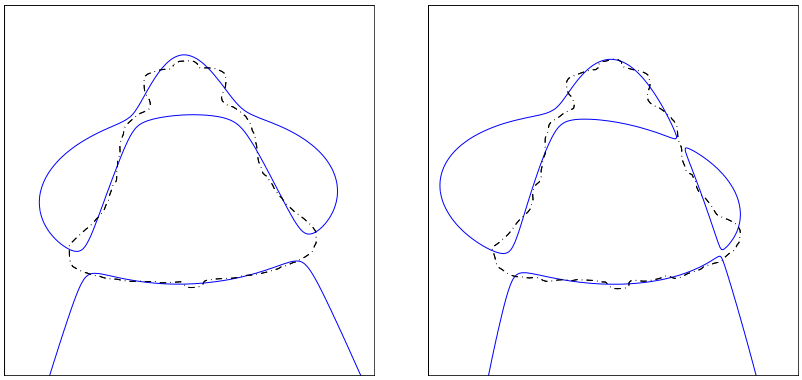
$$\min_{\underline{a}} \underline{a}' M' M \underline{a}, \tag{6}$$

where the $n \times \frac{(r+1)(r+2)}{2}$ $matrix\ of\ monomials$ M is defined by:

$$M = \begin{bmatrix} \underline{p}^{r}(x_1, y_1)' \\ \underline{p}^{r}(x_2, y_2)' \\ \cdots \\ \underline{p}^{r}(x_n, y_n)' \end{bmatrix} \tag{7}$$

The solution of (6) is simple (O(n) complexity) and is given by the eigenvector of M'M having the minimal eigenvalue. However, this solution may produce several zero-sets lying nearby the data-set, i.e., zero-set splitting ( Fig. 2a) and, in case that one or more eigenvalues are close to the minimal eigenvalue, it would be highly sensitive to even small perturbations of the data (Fig. 2b).

To avoid zero-set splitting, Gradient-one exploits the fact that the gradient vector at a point belonging to the zero-set is perpendicular to the zero-set tangent at this point. Assuming that the resulting zero-set indeed lies nearby the data-set, the Gradient-one algorithm requires that



(a) (b)

Fig. 2. Shortcomings of the LS solution (a) 4-th degree polynomial fit: 2 zero-sets attempting to fit the "bell". Actually, there is no single zero set resembling the data-set. (b) Colored noise, std = 0.01 was added to the "bell" coordinates. Although, the difference between the original figure and the noisy one is hardly noticeable. The same 4-th degree polynomial fit quite differs from the one appearing in (a).

the gradient of the polynomial at the data-set points will be perpendicular to the data-set curve-tangent.

In addition, the authors of [2] show that large fluctuations in the first derivative of the polynomial can result in a polynomial sensitive to data-set fluctuations. Hence, the required value of the gradient at data-set points is set to 1.

The last two requirements (i.e., gradient value and direction) are formulated by demanding that the polynomial derivatives in the directions that are normal and tangent to the data-set curve-tangent be equal to one and zero, respectively.

The first derivative of the polynomial in the x and y directions are, from (2):

$$\frac{\partial P^r_{\underline{a}}(x,y)}{\partial x} = \frac{\partial \underline{p}^r(x,y)'}{\partial x} \cdot \underline{a} \tag{8}$$

$$\frac{\partial P^r_{\underline{a}}(x,y)}{\partial y} = \frac{\partial \underline{p}^r(x,y)'}{\partial y} \cdot \underline{a} \tag{9}$$

Denoting by $\alpha$ the angle between the normal at the point $(x,y)$ and the x-axis, the derivative in the normal direction is given by:

$$\begin{aligned} \underline{g}'_N(x,y) \cdot \underline{a} \quad &= \cos(\alpha) \cdot \tfrac{\partial \underline{p}^r(x,y)'}{\partial x} \cdot \underline{a} \\ &+ \sin(\alpha) \cdot \tfrac{\partial \underline{p}^r(x,y)'}{\partial y} \cdot \underline{a}, \end{aligned} \tag{10}$$

and the derivative in the tangent direction is

$$\begin{aligned} \underline{g}'_T(x,y)\cdot\underline{a} = \quad & -\sin(\alpha)\cdot\frac{\partial \underline{p}^r(x,y)'}{\partial x}\cdot\underline{a} \\ & +\cos(\alpha)\cdot\frac{\partial \underline{p}^r(x,y)'}{\partial y}\cdot\underline{a} \end{aligned} \tag{11}$$

Finally the Gradient-one algorithm formulates the following minimization problem:

$$\begin{aligned} \min_{\underline{a}}\{ \quad & \textstyle\sum_{k=1}^{n} P^r_{\underline{a}}(x_k,y_k)^2 + \\ & \textstyle\sum_{k=1}^{n} (\underline{g}'_N(x_k,y_k)\cdot\underline{a}-1)^2 + \\ & \textstyle\sum_{k=1}^{n} (\underline{g}'_T(x_k,y_k)\cdot\underline{a})^2\}. \end{aligned} \tag{12}$$

2. **Min-Max and Min-Var Algorithms**

These two algorithms [3] are based on a $sensitivity\ function$ that determins the sensitivity of the zero-set to polynomial coefficients perturbations:

$$du(x,y) = \frac{\underline{p}^r(x,y)'}{||\nabla P_{\underline{a}}(x,y)||}\cdot d\underline{a}, \tag{13}$$

where $du$ is the distance passed by a point $(x,y)$, belonging to $ZS\left(P^r_{\underline{a}}(x,y)\right)$, as a result of a small change, $d\underline{a}$, in the coefficients. The algorithms developed in [3] propose to constrain the values of the gradient at the data-set points (which are assumed to be close to the derived zero-set), so that all points belonging to the obtained zero-set would be equally sensitive to small coefficient perturbations.

In the case of Min-Max, the assumption is that elements of $d\underline{a}$ are independently distributed in the range $[-\varepsilon\ \varepsilon]$. As a result, the maximum distance that the zero-set point (x,y) passes is given by [3]:

$$max_d(x,y) = \frac{w_{mm}(x,y)}{||\nabla P_{\underline{a}}(x,y)||}\cdot|\varepsilon|, \tag{14}$$

where,

$$w_{mm}(x,y) \stackrel{\triangle}{=} \sum_{i=1}^{\frac{(r+1)(r+2)}{2}} |p^r_i(x,y)| \tag{15}$$

and the subscript $i$ denotes the i-th element in the vector of monomials.

In the case of Min-Var, the assumption is that elements of $d\underline{a}$ are independently distributed while the variance of each element is $\xi^2$. As a result, the variance of the distance zero-set point (x,y) passes, $var(du(x,y))$, is given by [3]:

$$var(du(x,y)) = \frac{w^2_{mv}(x,y)}{||\nabla P_{\underline{a}}(x,y)||^2}\cdot\xi^2, \tag{16}$$

where

$$w_{mv}(x,y) \stackrel{\triangle}{=} \sqrt{\underline{p}^r(x,y)'\underline{p}^r(x,y)} \tag{17}$$

The authors of [3] require the ratio $\frac{w_{mm}(x,y)}{||\nabla P_{\underline{a}}(x,y)||}$ for Min-Max or the ratio $\frac{w_{mv}(x,y)}{||\nabla P_{\underline{a}}(x,y)||}$ for Min-Var to be constant (equal to 1) along the data-set. Note that changing this constant to some positive value $k$ (both $max_d(x,y)$ and $var(du(x,y))$ cannot be negative) would just multiply the resulting coefficient vector $\underline{a}$, which has no effect on the polynomial's zero-sets.

As in the Gradient-one case, the gradient is supposed to be perpendicular to the data-set. Thus, the Min-Max algorithm minimizes

$$\begin{aligned} \min_{\underline{a}} \quad & \{\textstyle\sum_{i=1}^n P_{\underline{a}}^r(x_i,y_i)^2 + \\ & \textstyle\sum_{i=1}^n \left(\frac{\underline{g}_T'(x_i,y_i)\cdot\underline{a}}{w_{mm}(x_i,y_i)}\right)^2 + \\ & \textstyle\sum_{i=1}^n \left(\frac{\underline{g}_N'(x_i,y_i)\cdot\underline{a}}{w_{mm}(x_i,y_i)} - 1\right)^2\} \end{aligned} \tag{18}$$

and the Min-Var:

$$\begin{aligned} \min_{\underline{a}}\{ \quad & \textstyle\sum_{i=1}^n P_{\underline{a}}^r(x_i,y_i)^2 + \\ & \textstyle\sum_{i=1}^n \left(\frac{\underline{g}_T'(x_i,y_i)\cdot\underline{a}}{w_{mv}(x_i,y_i)}\right)^2 + \\ & \textstyle\sum_{i=1}^n \left(\frac{\underline{g}_N'(x_i,y_i)\cdot\underline{a}}{w_{mv}(x_i,y_i)} - 1\right)^2\} \end{aligned} \tag{19}$$

Note that the only difference between the above two expressions is in using $w_{mm}(x,y)$ for Min-Max and $w_{mv}(x,y)$ for Min-Var.

A comparison of the performance of the Min-Max and Min-Var algorithms relative to Gradient-one is shown in Figs. 3 and 4. Note, that along with better stability, the first two algorithms provide a more accurate fit.

We discuss now another interesting property of the algorithms proposed in [3]. It can be seen that $w_{mm}(x,y)$ and $w_{mv}(x,y)$ increase as we move away from the origin. The Min-Max and Min-Var algorithms constrain the value of the gradient at the data-set points to be equal to these values, respectively, while the Gradient-one algorithm constrains it to be equal to 1. As a result, the polynomial surface of either Min-Max or Min-Var is flatter than Gradient-one, when it is close to the origin, but steeper when it is far from it. Translating the data-set, so that its center of mass lies at the origin, is part of the data-preprocessing before the polynomial fitting. Thus, generally, the surface of the polynomial derived by Min-Max or Min-Var will be flatter $inside$ the data-set and steeper $outside$ it, as compared to the Gradient-one polynomial surface.

This property has an interesting effect on spurious zero-sets. Usually, these zero-sets don't disturb much, unless they appear close to the zero-set that fits the data-set, since it is then difficult to choose the zero-set describing the original contour. The flatter the surface in the vicinity of the "proper" zero-set, the "easier" it is for the polynomial to intersect the zero-plane (and thus create a spurious zero-set) close to this zero-set. Based on these observations we can conclude, that compared to Gradient-one, Min-Max and Min-Var internal spurious zero-sets (i.e., spurious zero-sets inside the data-set) may appear closer to the fitting zero-set while external zero-sets are expected to appear farther away. This is, of course, rather a tendency than a rule. Another tendency of spurious zero-sets, is that they are more likely to appear outside the data-set than inside it. Therefore, in most cases, Min-Max and Min-Var spurious zero-set will appear farther away from the fitting zero-set, than Gradient-one's. Several examples are shown in Fig. 4.

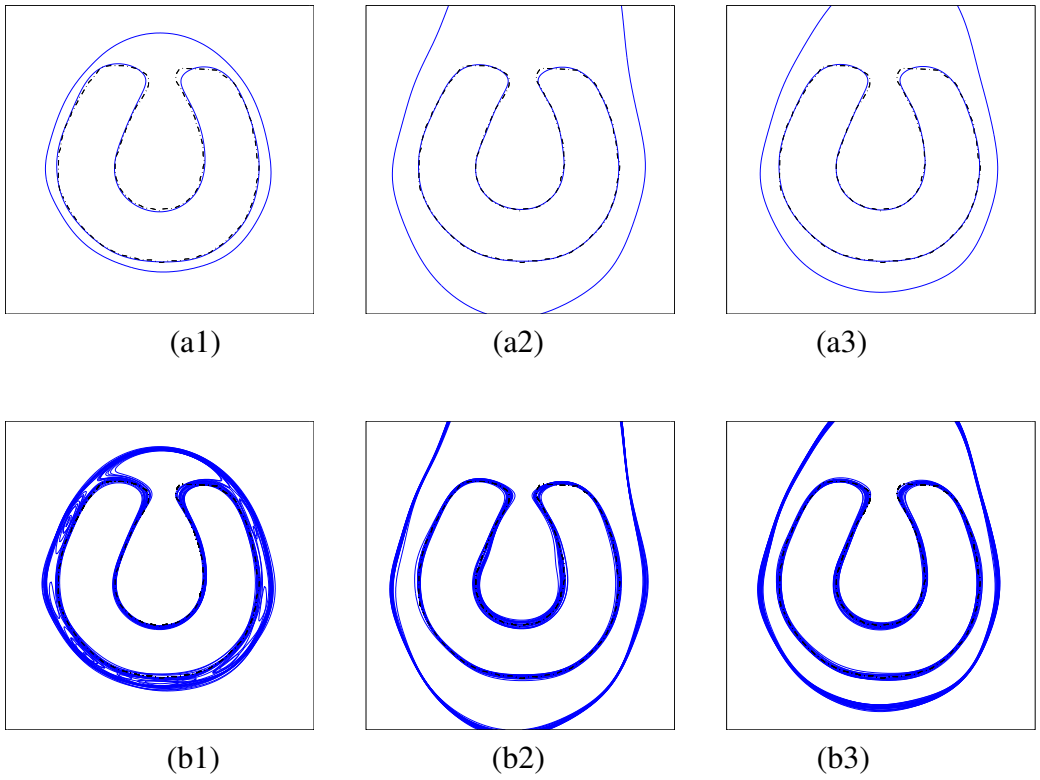



Fig. 3. Fit of a polynomial of degree 8 (solid line) to the "horseshoe" contour (dashed line). (1) – Gradient1, (2) – Min-Max,(3) – Min-Var. The top row depicts the derived zero-set, the bottom one – accumulated zero-sets , when noise is added to each coefficient. The noise is uniformly distributed with mean 0 and absolute maximum value of $2^{-10}$ of the biggest coefficient.

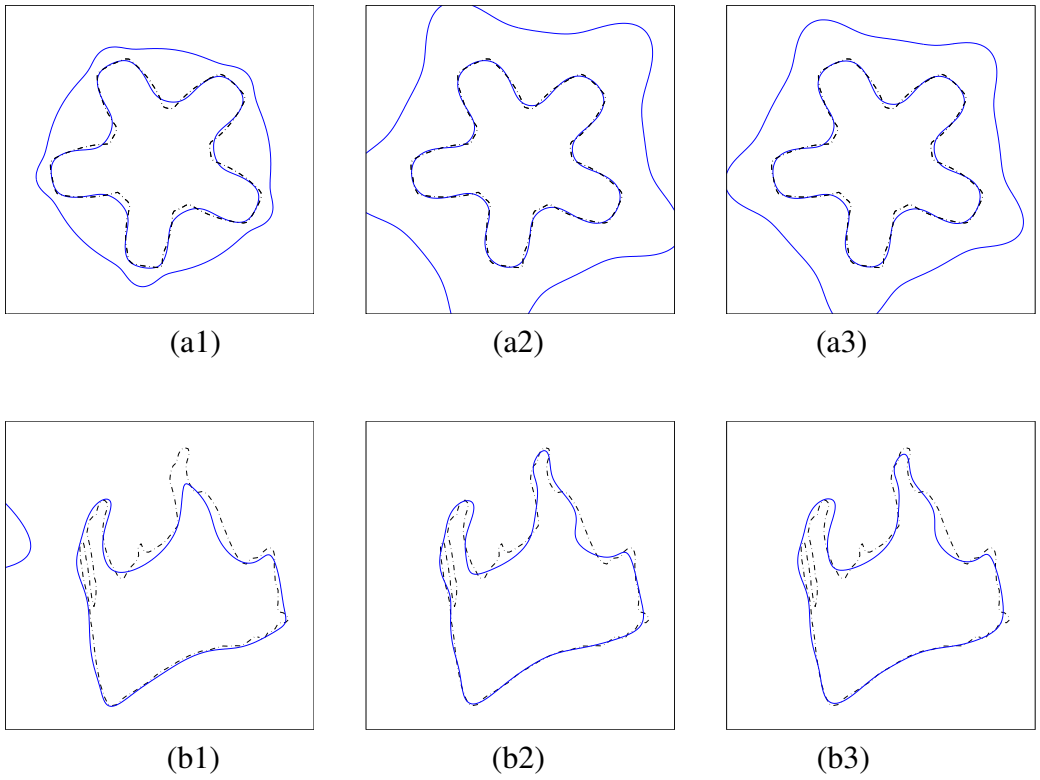



Fig. 4. Comparison of fitting results for Gradient-one (left), Min-Max (center) and Min-Var (right) fit (polynomial of degree 8). The data-set depicted by a dashed line, zero-sets – by solid.

### *C. Modification of Min-Max and Min-Var Algorithms for Improved Fitting Performance*

Although the Min-Var and Min-Max algorithms describe the data quite well and are stable, there is still room for improvement. These algorithms minimize the polynomial value along the data-set instead of minimizing the distance between the data-set and the polynomial zero-sets, i.e.,

$$\min_{\underline{a}} \sum_{k=1}^{n} e_{\underline{a}}(k)^2, \tag{20}$$

where $e_{\underline{a}}(k)$ is a minimal distance between the k-th point of the data-set to the zero-sets of the polynomial (i.e., the fitting error).

It is obvious that it is impossible to cast this problem as a linear LS minimization problem. However, we overcome this problem by considering the minimization of the approximated fitting

error (see [5]):

$$\min_{\underline{a}} \sum_{k=1}^{n} \left( \frac{P_{\underline{a}}(x_k, y_k)}{|| \bigtriangledown P_{\underline{a}}(x_k, y_k)||} \right)^2. \tag{21}$$

Naturally, the value of the gradient at every point of the data-set is needed. Although the exact value of the gradient is not available before the problem is solved, in the case of Min-Max and Min-Var we have a clue of its value. Recall, that in the case of Min-Max a good solution presupposes values of the gradient at a point (x,y) to be close to $w_{mm}(x, y)$ (15), and in the case of Min-Var, close to $w_{mv}(x, y)$ (17). Thus, substituting these values into (21) we get:

$$\min_{\underline{a}} \sum_{i=1}^{n} \left( \frac{P_{\underline{a}}(x_i, y_i)}{w_{mm}(x_i, y_i)} \right)^2 \tag{22}$$

for Min-Max, and

$$\min_{\underline{a}} \sum_{i=1}^{n} \left( \frac{P_{\underline{a}}(x_i, y_i)}{w_{mv}(x_i, y_i)} \right)^2 \tag{23}$$

for Min-Var.

Expressions (22) and (23) replace the first element in equations (18) and (19), respectively.

However, it turns out that the polynomials derived by the solution of the resulting equations don't fit the data-set well. There is a simple explanation to it. Min-Max and Min-Var algorithms minimize a sum of two cost functions: the first one minimizes values of the polynomial along the data-set; the second – constrains the direction and the value of the gradient. In order to obtain a good fit, the relative weight of these cost functions should be chosen carefully. Weakening one of the cost functions (e.g., the "fitting errors" ) by dividing it by a factor bigger than one ( e.g., $w_{mm}$ or $w_{mv}$ ), may harm the result.

This problem can be solved by multiplying the cost function of the polynomial by a weight factor bigger than 1. Thus, we can regain timportance of fitting errors. A reasonable parameter can be the average value (over the data-set points) of $w_{mm}(x, y)$, for Min-Max:

$$W_{mm} = \frac{1}{n} \sum_{i=1}^{n} w_{mm}(x_i, y_i) \tag{24}$$

and average value of $w_{mv}(x, y)$, for Min-Var

$$W_{mv} = \frac{1}{n} \sum_{i=1}^{n} w_{mv}(x_i, y_i) \tag{25}$$

Thus, the equations (18) and (19) become:

$$\min_{\underline{a}} \left\{ \sum_{i=1}^{n} \left( W_{mm} \frac{P_{\underline{a}}(x_i, y_i)}{w_{mm}(x_i, y_i)} \right)^2 + \right.$$

$$\sum_{i=1}^{n}\left(\frac{\underline{g}'_T(x_i,y_i)\cdot\underline{a}}{w_{mm}(x_i,y_i)}\right)^2+\sum_{i=1}^{n}\left(\frac{\underline{g}'_N(x_i,y_i)\cdot\underline{a}}{w_{mm}(x_i,y_i)}-1\right)^2\Bigg\} \tag{26}$$

in case of Min-Max and:

$$\min_{\underline{a}}\Bigg\{\sum_{i=1}^{n}\left(W_{mv}\frac{P_{\underline{a}}(x_i,y_i)}{w_{mv}(x_i,y_i)}\right)^2+$$

$$\sum_{i=1}^{n}\left(\frac{\underline{g}'_T(x_i,y_i)\cdot\underline{a}}{w_{mv}(x_i,y_i)}\right)^2+\sum_{i=1}^{n}\left(\frac{\underline{g}'_N(x_i,y_i)\cdot\underline{a}}{w_{mv}(x_i,y_i)}-1\right)^2\Bigg\} \tag{27}$$

in case of Min-Var.

Figures 5 and 6 demonstrate the difference in fitting between the original Min-Max/Min-Var and the modified Min-Max/Min-Var (Mod. Min-Max/ Mod. Min-Var). If we compare the fittings, we conclude that the original algorithms make more effort to fit points lying far away from the origin, than points lying close to it; while the modified algorithms result in a more uniform solution.

## III. Application of Implicit Polynomials to 2D Object recognition

The ability to recognize objects is very important in many fields, such as medical and robotic fields. In this section we bring a brief overview of existing IP based recognition. Then, we propose to preprocess the data in order to get better recognition and suggest a further modification of the Min-Max and Min-Var fitting techniques, so that we would be able to base the recognition process described in [2] not only on Gradient-one fit, but also on Mod. Min-Max and Mod. Min-Var.

### *A. General Overview*

The recognition problem we are going to deal with can be described by the following scheme: Assume, that we have $L$ different 2D objects, each represented by a set of points describing its boundary. It is possible that each object has several representations (for example, the same object was extracted from different pictures), i.e., object $l$ has several ($S_l$) $views$ or, equivalently, data-sets.

Once we have extracted a new shape, we would like to determine the group, to which this shape belongs to, i.e., to $recognize$ the shape.

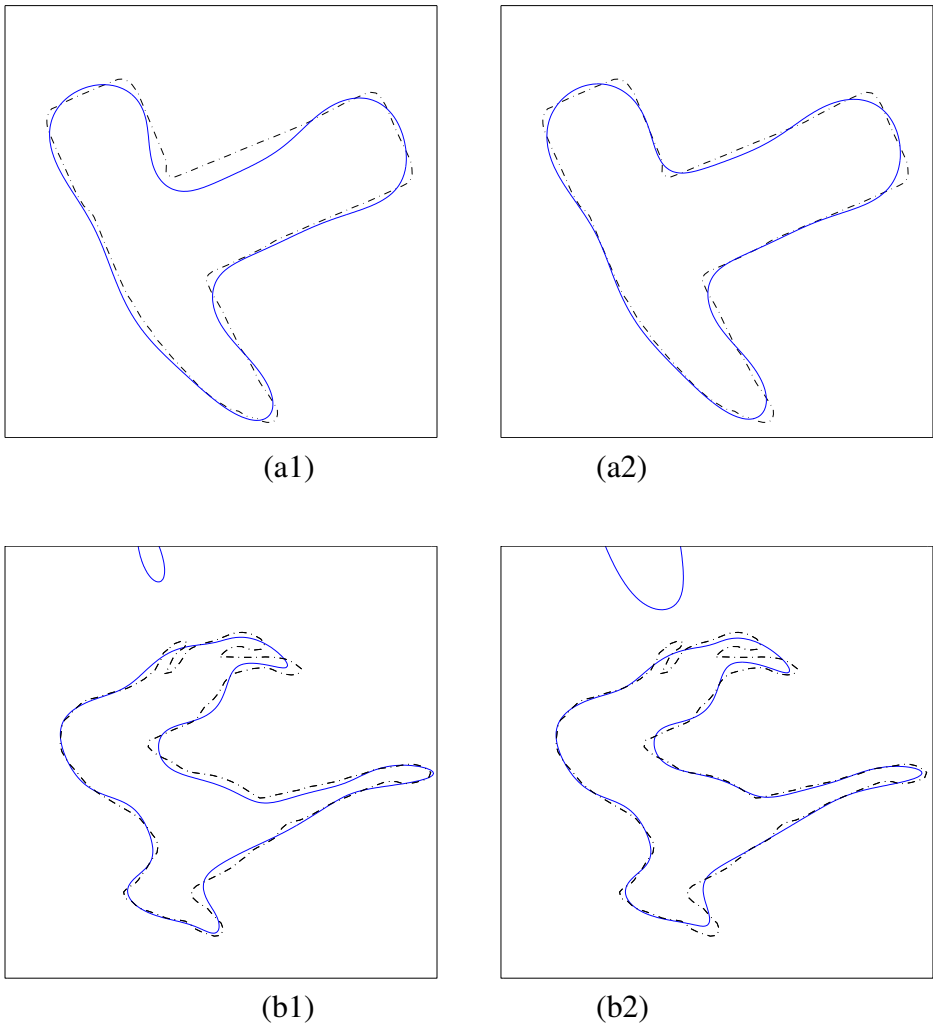



Fig. 5. Min-Max (left) and Mod. Min-Max (right) fit (polynomial of degree 8). Data-set – dashed line, obtained zero-sets – solid.

There are recognition methods that compare the extracted shape to a dictionary of objects directly, like in [11]. Generally, these methods have high computational cost and even simple transformations (like rotation) can result in a very complicated recognition process.

To produce a more effective recognition, indirect methods are used. First, at the learning stage, some special features (algorithm dependent) are extracted from every data-set in the dictionary. Second, at the testing stage, the same features are extracted from the unknown shape, and the comparison is based on these features. Among these we can mention techniques based on Fourier Descriptors [19], Convex Hull [21], Turning Angle [20] and Cuvature Scale Space (CSS) [18].

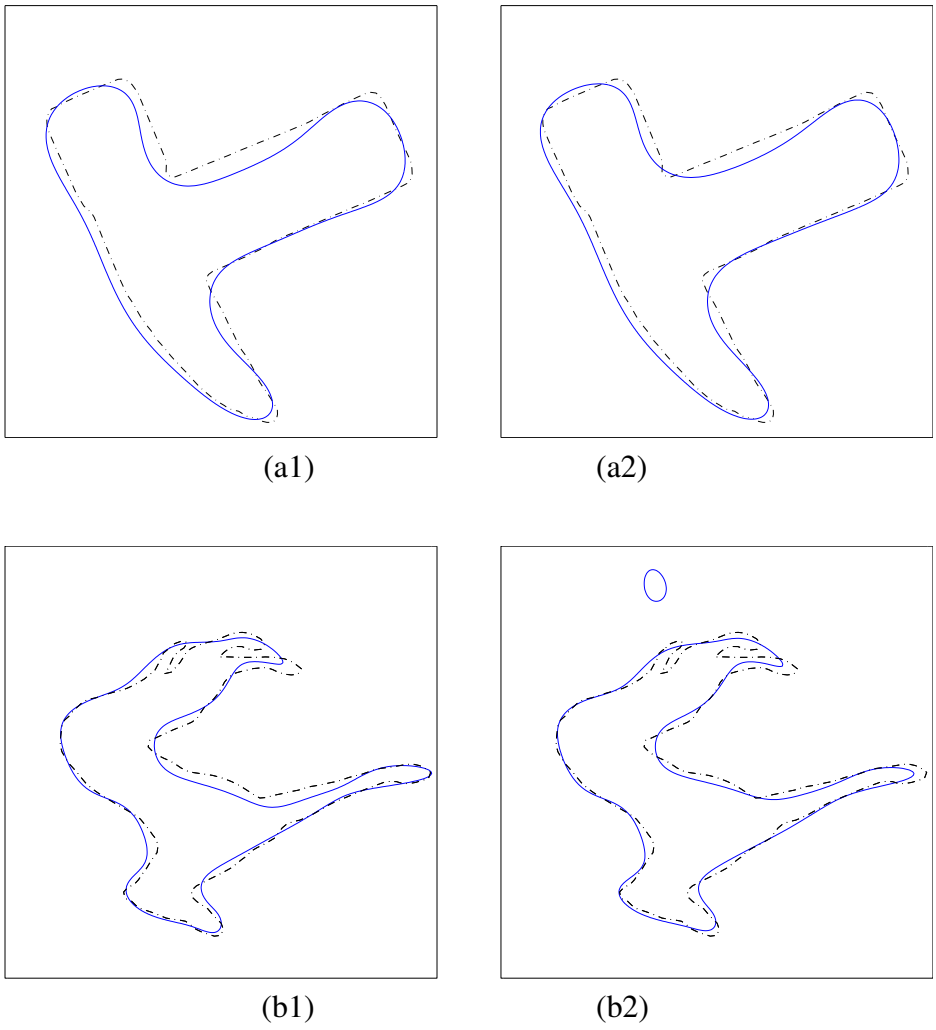



Fig. 6. Min-Var (left) and Mod. Min-Var (right) fit (polynomial of degree 8). Data-set – dashed line, obtained zero-sets – solid.

IP-s are also used to recognize objects indirectly [2]. In this case the extracted features are based on the coefficients of the fitted polynomial.

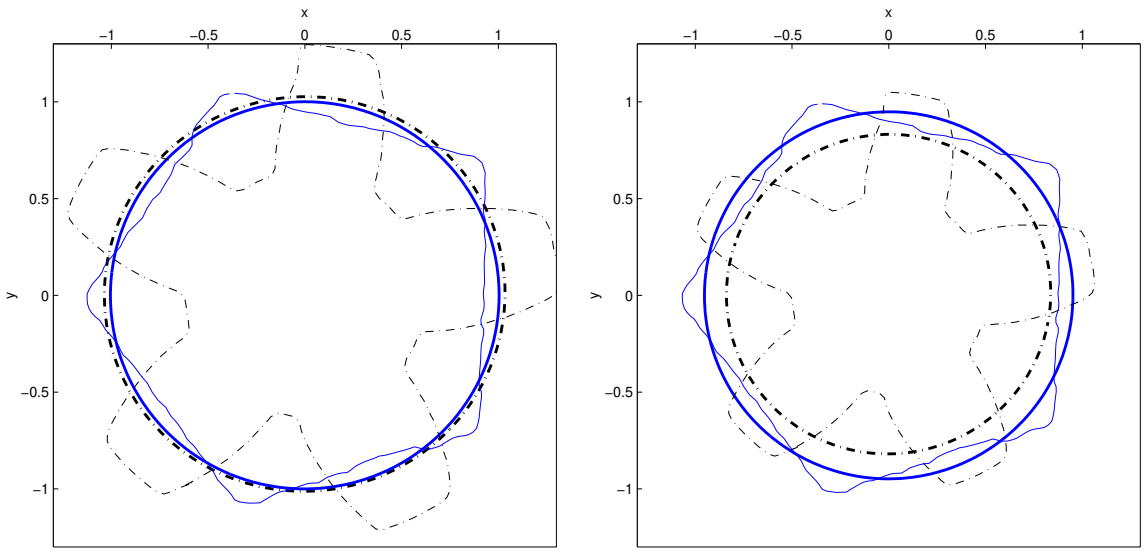
### *B. Data preprocessing*

In this subsection we show that preprocessing the data-set, before applying the LS minimization, is very important for the recognition process. The preprocessing can be expressed as some arbitrary transformation of the data-set to another data-set. The aim is that as a result of the transformation, the features extracted from the data-set would become both more stable (i.e.,

distance between features extracted from different views of the same object would be small) and more distinguishable (i.e., distance between features extracted from different objects would be big).

As shown in [2], putting the data-set close to the unit circle, helps to avoid large perturbations in the coefficients as a result of data-set noise. Hence, the preprocessing proposed in [2] is first to translate the data-set so that its center of the mass would be at the origin and then scale it so that the average distance of the data-set points from the origin is equal to 1. We denote this scaling factor by $\mathbb{S}_{av}$.

In this work, we propose another scaling factor $\mathbb{S}_{75\%}$ that scales the data-set so that 75% of its points would appear inside the unit circle and the rest – outside of it. Generally, we found that zero-sets (and coefficients) of polynomials, fitted to data-sets of different objects, are more distinguishable if the data-set is scaled by $\mathbb{S}_{75\%}$ than if it scaled by $\mathbb{S}_{av}$. An example of this phenomenon is shown in Fig. 7. As can be seen, the zero-sets of the IP-s belonging to "pentagon" and "star" are quite the same in the case of the $\mathbb{S}_{av}$ scaling factor, but differ in the case of $\mathbb{S}_{75\%}$. That means that the recognition will be better in the second case.




(a) (b)

Fig. 7. Examples of normalization with $\mathbb{S}_{av}$ (a) and $\mathbb{S}_{75\%}$ (b) scaling factors. The "pentagon" (solid) and "star" (dash-dot) data-sets. The zero-sets of degree 2 polynomial are drawn by heavy lines (solid and dash-dot, respectively).

In addition, we propose that before translating and scaling the data-set, the measurement noise

should be filtered out by passing the data-set through a low-pass filter (the procedure appears in [3]).

### *C. Further modification of Min-Max and Min-Var Algorithms for rotation invariant recognition*

In many recognition tasks, the object to be recognized may appear in any orientation. Pose estimation, based on highest homogenous form of the IP was introduced in [7]. It allows pose estimation of a $known$ object, but has poor performance in recognition. Other, more successful methods, are based on rotation invariant features of the object. Rotation invariants that are based on the polynomial coefficients, were first derived by Keren [8]. These invariants have a form of sum of coefficient products. These products may cause instability, especially for high degree polynomials (requiring multiplication of more coefficients). Therefore, we prefer using another set of invariants, those derived by Tarel [9]. This set contains invariants that depend linearly, quadratically, and angularly on the polynomial coefficients. In our experiments we used only the linear invariants (which due to their simple dependence on polynomial coefficients are more likely to be stable). Furthermore, in our experiments using additional invariants did not improve the recognition rate.

In order to take advantage of these invariants, we should make sure that the relation between polynomials fitted to different orientations of the same data-set is characterized by rotation only. All the aforementioned fitting algorithms demand that at the data-set points:

1) The polynomial value equals zero.
2) The gradient direction coincides with the normal to the data-set at each point.
3) The gradient value equals to a constant, or is a function of the point coordinates (algorithm dependent).

If we have two different polynomials related by rotation only (by an angle $\theta$), $P^r_{\underline{a}}(x,y)$ and $P^r_{\underline{\tilde{a}}(\theta)}(x,y)$, then

$$P^r_{\underline{\tilde{a}}(\theta)}(\underline{\tilde{x}}(\theta),\underline{\tilde{y}}(\theta)) = P^r_{\underline{a}}(\underline{x},\underline{y}), \tag{28}$$

$$\measuredangle\nabla P^r_{\underline{\tilde{a}}(\theta)}(\underline{\tilde{x}}(\theta),\underline{\tilde{y}}(\theta)) = \measuredangle\nabla P^r_{\underline{a}}(\underline{x},\underline{y}) + \theta, \tag{29}$$

$$|\nabla P^r_{\underline{\tilde{a}}(\theta)}(\underline{\tilde{x}}(\theta)\underline{\tilde{y}}(\theta))| = |\nabla P^r_{\underline{a}}(\underline{x},\underline{y})|, \tag{30}$$

where $\underline{\tilde{x}}(\theta)$ and $\underline{\tilde{y}}(\theta))$ are $\underline{x}$ and $\underline{y}$ rotated by $\theta$.

It is clear that the first two conditions fulfill (28) and (29). The value of the gradient depends on the algorithm we apply. In the case of Gradient-one, this value is set to 1 for each data-set point and, therefore, (30) is met. Thus, Gradient-one can be used for rotation invariant recognition. However, for the other algorithms considered, some modifications need to be done in order to meet (30).

*1) Min-Max and Mod. Min-Max:* For these algorithms, the value of the gradient at a data-set point $(x_k, y_k)$, $|\nabla P^r_{\underline{a}}(x_k, y_k)|$, is required to be equal to $w_{mm}(x_k, y_k) = \sum_{i=1}^{\frac{(r+1)(r+2)}{2}} |p^r_i(x_k, y_k)|$ (see (15)). Clearly, this value is not rotation invariant. However, if we replace it by $w^{ri}_{mm}(d_k)$:

$$w^{ri}_{mm}(d_k) = \underset{x,y,\ x^2+y^2=d_k^2}{mean} \left(w_{mm}(x, y)\right), \tag{31}$$

where $d_k = \sqrt{x_k^2 + y_k^2}$ is the distance of point $(x_k, y_k)$ from the origin, then the gradient value requirements would be rotation invariant (indicated by $(\cdot)^{ri}$). We denote this modification of Min-Max by RI Min-Max. Note that the $w^{ri}_{mm}(d_k)$ is based on a circle of radius $d_k$ and not on data-set points laying at this distance from the origin., i.e., this parameter is independent on data-set.

There is a need to check whether this approximation is good enough. Fig. 8 shows the value of $w^{ri}_{mm}(d)$ as a function of $d$ along with the minimum and the maximum values of $w_{mm}(x, y)|_{x^2+y^2=d^2}$. It can be seen that for high degree polynomials these three quantities hardly differ. Therefore, the use of the average value is not expected to harm much the Min-Max/Mod. Min-Max properties. Actually, we did not find any degradation, neither in fitting performance, nor in sensitivity to perturbations in the polynomial coefficients. Note that if Mod. Min-Max is changed to RI Mod. Min-Max we also replace $W_{mm}$ (24) by

$$W^{ri}_{mm} = \frac{1}{n} \sum_{i=1}^{n} w^{ri}_{mm}(\sqrt{x_i^2 + y_i^2}). \tag{32}$$

Note also that the values $w^{ri}_{mm}(d)$ (which is data-set independent) can be computed in advance (in our experiments we discretized $d$ using small steps, e.g., 0.01). Thus, the online computational cost is not increased.

*2) Min-Var and Mod. Min-Var:* For these algorithms, the value of the gradient at a data-set point $(x, y)$, $|\nabla P^r_{\underline{a}}(x, y)|$, is required to be equal to $w_{mv}(x, y) = \sqrt{\underline{p}^r(x, y)' \underline{p}^r(x, y)}$ (see (15)). This expression can be rewritten as $\sqrt{\sum_{l=0}^{r} \sum_{i}^{l} (x^{(l-i)} y^i)^2}$, and it is obvious that it is not invariant

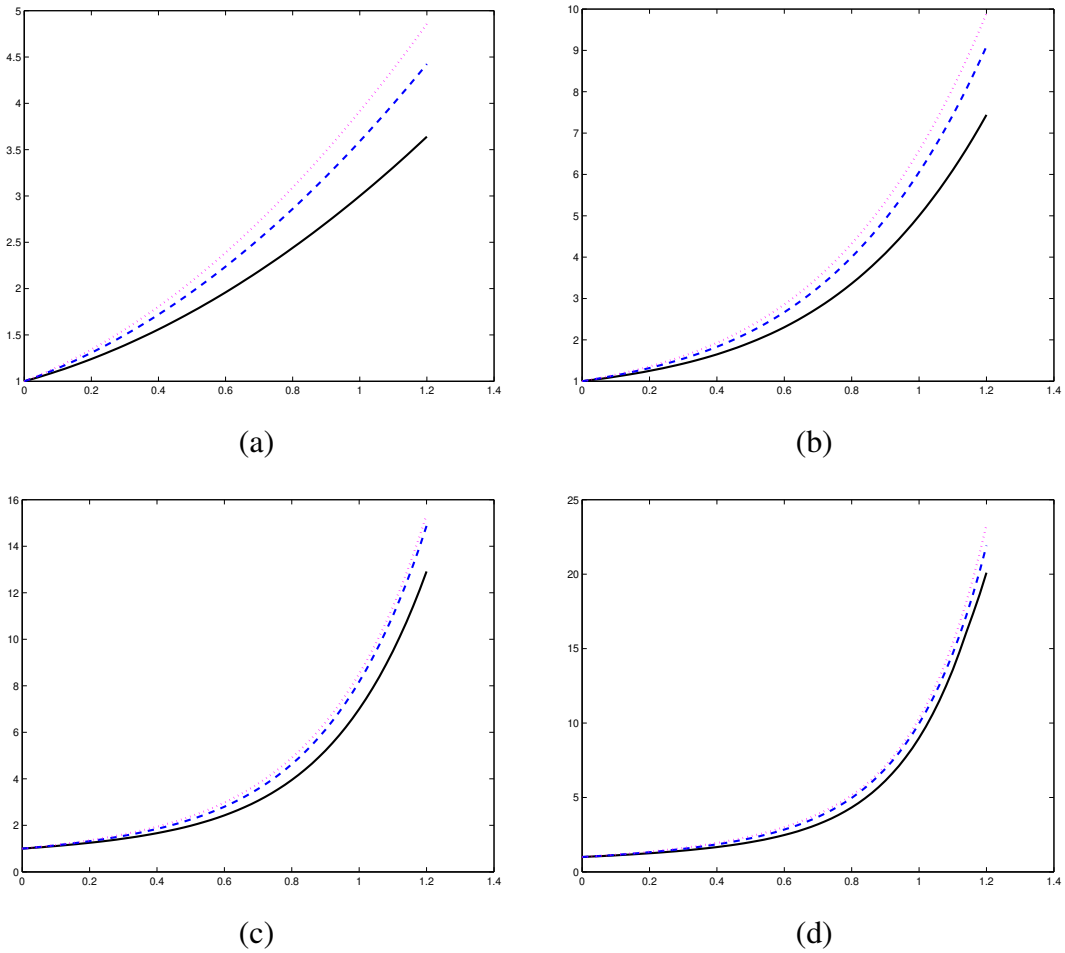



Fig. 8. $w^{ri}_{mm}(d)$ (dashed line) along with the minimum (solid line) and the maximum (dotted line) value of $w_{mm}(x,y)|_{x^2+y^2=d^2}$ as a function of the distance from the origin $d$ for different degrees of the polynomial. (a) 2-nd degree, (b) 4-th degree, (c) 6-th degree, (d) 8-th degree.

to rotation. However, for rotation invariance it can be replaced by

$$\begin{aligned} w^{ri}_{mv}(x,y) &\stackrel{\triangle}{=} \sqrt{\textstyle\sum_{l=0}^{r}(x^2+y^2)^l} \\ &= \sqrt{\textstyle\sum_{l=0}^{r}\sum_{i}^{l}\binom{l}{i}x^{2\cdot(l-i)}y^{2i}} \end{aligned} \tag{33}$$

This is done by redefining the polynomial as follows:

$$\begin{aligned} P_{\underline{a}}(x,y) &= \sum_{l=0}^{r}\sum_{i=0}^{l} a_{l-i,j}x^{l-i}y^{i} \\ &= \sum_{l=0}^{r}\sum_{i=0}^{l} \underbrace{\frac{a_{l-i,i}}{\sqrt{\binom{l}{i}}}}_{b_{l-i,i}} \cdot \sqrt{\binom{l}{i}}x^{l-i}y^{i} \\ &= \sum_{l=0}^{r}\sum_{i=0}^{l} b_{l-i,i}\sqrt{\binom{l}{i}}x^{l-i}y^{i} \\ &\triangleq \mathcal{P}_{\underline{b}}^{\,r}(x,y), \end{aligned} \tag{34}$$

where $\mathcal{P}_{\underline{b}}^{\,r}(x,y)$ denotes a polynomial of degree $r$, and the vector $\underline{b}$ is comprised of coefficients $b_{l-i,i}$ that multiply the $factorized$ monomials of the form $\sqrt{\binom{l}{i}}x^{l-i}y^{i}$.

The Jacobian , $\mathcal{J}_{ab}$, relating $\underline{a}$ and $\underline{b}$, is a diagonal matrix with the main diagonal consisting of: $(\sqrt{\binom{0}{0}}\ \sqrt{\binom{1}{0}}\ \sqrt{\binom{1}{1}}\ \sqrt{\binom{2}{0}}\ \sqrt{\binom{2}{1}}\ \sqrt{\binom{2}{2}}\ \cdots\ \sqrt{\binom{r}{r}})$.

Hence, (13) can be rewritten as:

$$du(x,y) = \frac{\underline{p}^{r}(x,y)'}{||\nabla P_{\underline{a}}(x,y)||} \cdot \mathcal{J}_{ab} \cdot d\underline{b} \tag{35}$$

Then, denoting by $\underline{\eta}^{r}(x,y)$ the vector consisting of factorized monomials $\sqrt{\binom{l}{i}}x^{l-i}y^{i}$, and replacing $||\nabla P_{\underline{a}}(x_i,y_i)||$ by $||\nabla \mathcal{P}_{\underline{b}}^{\,r}(x_i,y_i)||$ (these are just different notations for the same function) we get:

$$du(x,y) = \frac{\underline{\eta}^{r}(x_i,y_i)'}{||\nabla \mathcal{P}_{\underline{b}}^{\,r}(x_i,y_i)||} \cdot d\underline{b} \tag{36}$$

We assume that the polynomial is represented by $\underline{b}$ and, therefore, any processes perturbing the polynomial coefficients (like quantization) is applied directly to this vector. Under this assumption, we require that the value of the gradient should be $\sqrt{\underline{\eta}^{r}(x,y)'\underline{\eta}^{r}(x,y)}$, which is equal to $w_{mv}^{ri}$ and get the RI Min-Var algorithm. In the case of Mod. Min-Var one needs to replace $W_{mv}$ in (25) by

$$W_{mv}^{ri} \triangleq \frac{1}{n}\sum_{i=1}^{n} w_{mv}^{ri}(x_i,y_i) \tag{37}$$

A comparison of the Mod. Min-Var and the RI Mod. Min-Var is shown in Fig. 9. Note that both algorithms have the same fitting performance. RI Mod. Min-Var, however, is less sensitive to coefficients noise. The reason for it is that the noise is added to the coefficients of $\underline{b}$ instead of to $\underline{a}$. The STD of the noise we added is relative to the $biggest$ coefficients. Because the data-set lies mostly inside a circle of radius 1, the higher the monomial power the bigger its coefficient. On the other hand, the factors, $\sqrt{\binom{l}{i}}$ also tend to grow with monomials power. Therefore, the

biggest value of $\underline{b}$ may be considerably smaller than that of $\underline{a}$, while the smallest values of these vectors would remain almost the same. Consequently, the dynamic range of $\underline{b}$ is smaller than the dynamic range of $\underline{a}$ and, hence, $\underline{b}$ is less sensitive to coefficient noise.

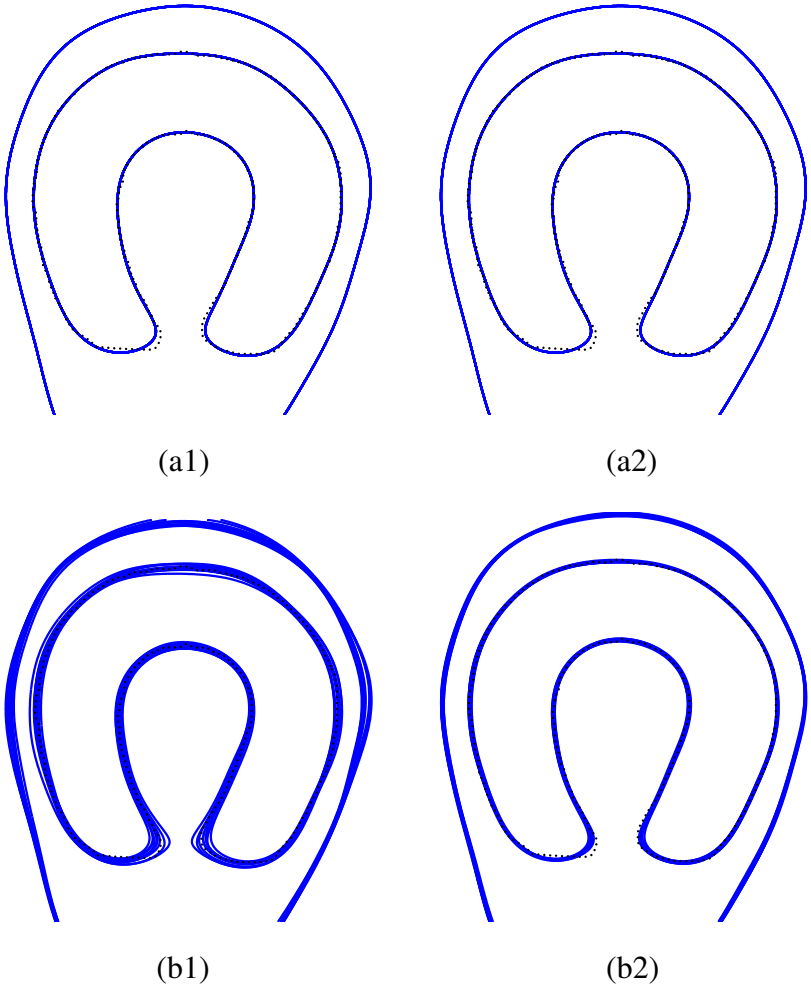



Fig. 9. Left column: Application of the original Mod. Min-Var of degree 8 to the "Horseshoe" shape. Right column: Application of the RI Mod. Min-Var of degree 8 to the "Horseshoe" shape. (a) Heavy-dotted line: the original data-set, solid line – the fitted polynomial zero-set. (b) Heavy-dotted line: the original data-set. Solid lines – accumulated zero-sets, when noise added to each coefficient is uniformly distributed with mean 0 and absolute maximum value of $2^{-10}$ of the biggest coefficient.

# IV. Multi Order (degree) and Fitting Error Technique (MOFET) for data-set recognition

## A. Drawbacks of single-degree recognition

All the aforementioned algorithms use coefficients of a polynomial of a predefined degree (typically, 6 or 8) to recognize objects. However, generally, the dictionary would contain contours of different complexity. Thus, we can find some contours that cannot be fitted well by the polynomial of the chosen degree, and, as a result, the resulting zero-set may resemble other shapes in the dictionary, making them indistinguishable. On the other hand, we might have some simple contours that can be fitted by a variety of polynomials. For example, if the contour is a circle $x^2+y^2=1$ and the degree is 4, then any polynomial of the form $(ax^2+bxy+cy^2+dx+ey+f)(x^2+y^2-1)=0$ would match. As a result, adding even small noise to such simple data-set may result in a totally different polynomial, which complicates the recognition.

We demonstrate this problem by the next example: assume three different data-sets: "Orange", "Dog" and "Cow", shown in Fig.10. We add multiple times colored noise with STD = 0.05 to each of the data-sets and fit to the obtained data-sets a polynomial of a "candidate" degree. Each point in Fig. 11 represents the coefficients $a_{0,0}$ and $a_{1,0}$ of one of the fitted polynomial. Note that when a polynomial of degree 6 was fitted to the data-sets, Fig. 11a, then the "clouds" belonging "Cow" and "Dog" shapes were contiguous and even overlapping. This means that recognition based on coefficients of a degree 6 polynomial will have very low recognition rate on these data-sets, i.e., the data-sets are too $complicated$ for a polynomial of degree 6. The conclusion is that we need to increase the degree of the polynomial, to improve recognition performance. Then, we fitted polynomials of degree 8, Fig. 11b-c. This time, the "Dog" and the "Cow" were separated well, but the "Orange" shape cloud is very dispersed, which means that the fitting polynomial is highly unstable.

## B. Recognition based on several polynomials

We propose the following solution to the single degree recognition problem: *Matching polynomials of several different degrees to each of the data-sets*.

This way we make sure that for each object there exists a set of reliable and valuable parameters. To demonstrate how this technique can solve the previously depicted problem, we return to the example of "Orange", "Dog" and "Cow" shapes.

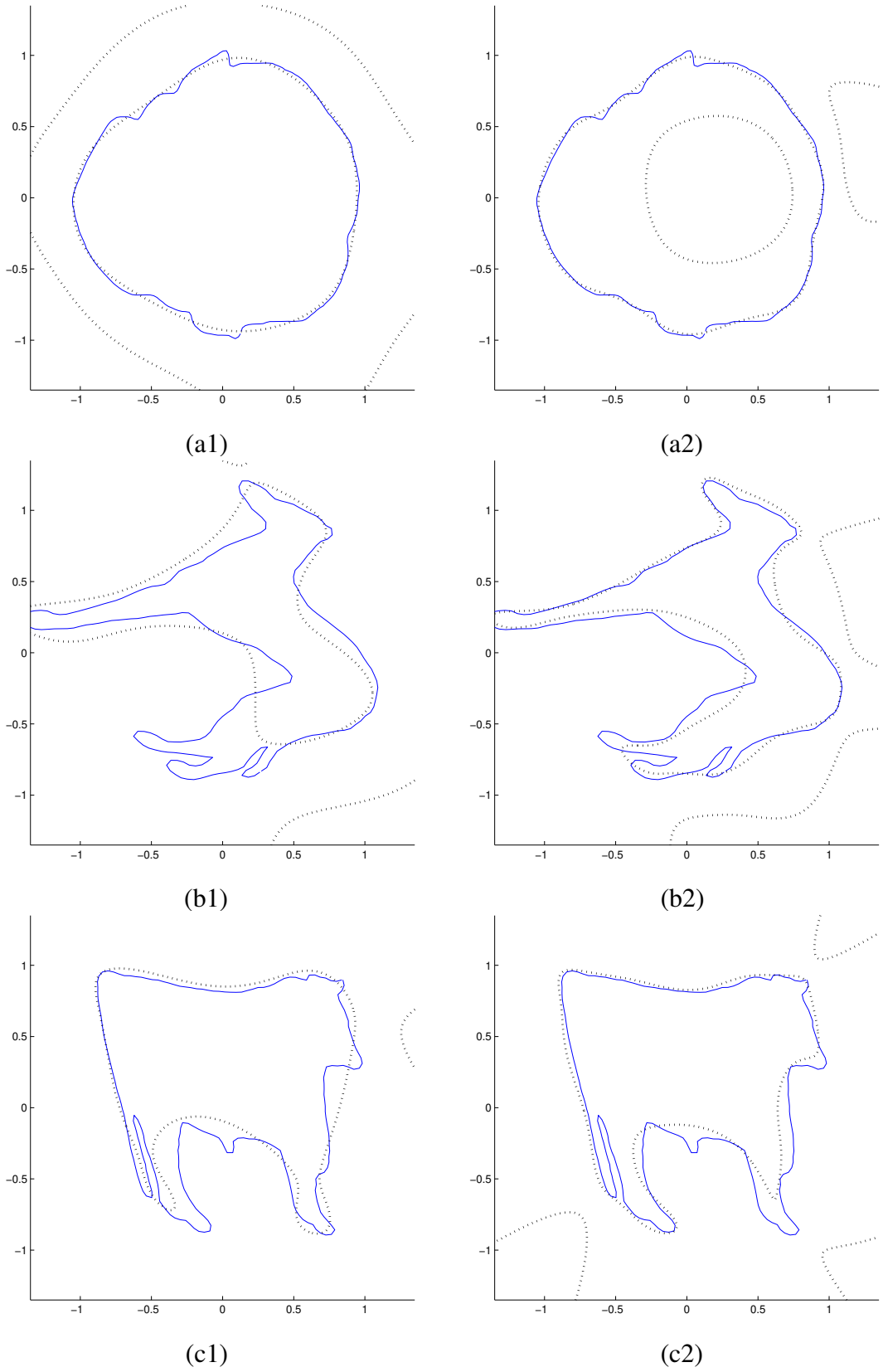
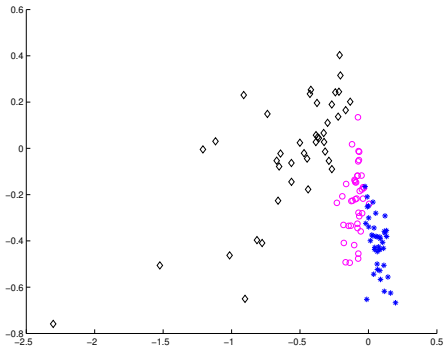
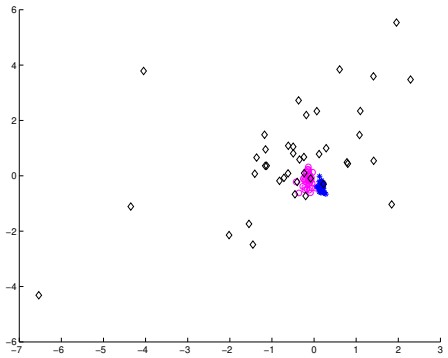


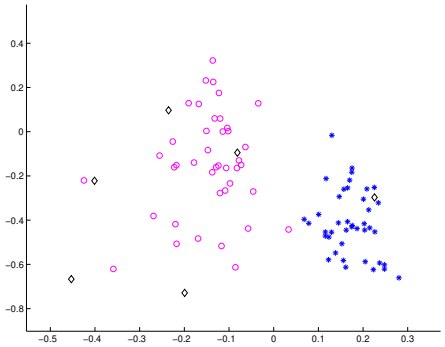

Fig. 10. (a) ”Orange”, (b) ”Dog”, (c) ”cow”. Solid line: Original data-set, Dotted: Zero-set of fitting polynomial. Left column: matching polynomial is of degree 6. Right: of degree 8.



(a)



(b)



(c)

Fig. 11. The projections of coefficient "clouds" : $a_{1,0}$ (y-axis) versus $a_{0,0}$ (x-axis). Diamonds represent "Orange" data-sets. Asterisks: "Dog" data-sets. Circles: "Cow" data-sets. (a) Coefficients of Gradient1 polynomials of degree 6, (b) Coefficients of Gradient-one polynomials of degree 8, (c) Coefficients of Gradient1 polynomials of degree 8 zoomed around "Dog" and "Cow" clouds.

Assume that at the learning stage the recognizer matched polynomials of degrees 6 and 8 to the data-sets belonging to each of 3 categories and extracted information about the data-sets "clouds". Then, at the recognition stage, the recognizer gets an unknown object and fits to it polynomials of degree 6 and 8. Let's denote by $\underline{a}_6$ the coefficient vector of the polynomial of degree 6 and by $\underline{a}_8$ the corresponding vector for the degree 8 polynomial. The recognizer has to decide whether the object is an "Orange", a "Dog" or a "Cow", i.e., it has to check 3 hypotheses.

Assume that the unknown data-set is "Orange". We would expect that $\underline{a}_6$ will fall inside the cloud belonging to the "orange" shape, which is clearly distinguishable from other objects clouds (see Fig. 11(a)). On the other hand, $\underline{a}_8$ may fall in any place of the coefficients space (see Fig. 11(b)), i.e., it does not carry reliable information. Therefore, if the recognizer will base its decision mostly on $\underline{a}_6$, *i.e., the reliable features,* it would mostly make a correct decision.

Now, assume that the unknown data-set is "Dog". Our expectation is that $\underline{a}_6$ will fall inside the cloud of "Dog" (which is mixed with the cloud of "Cow"), Fig. 11(a). At this stage we can already decide that the object is not an "Orange". Since $\underline{a}_8$ will most probably fall inside the "Dog" cloud (Fig. 11(c)) we expect to make a correct decision. Analysis of the "Cow" shape is identical to the analysis of "Dog".

The conclusion is that in order to improve recognition performance, several polynomials (of different degree) should be matched to each object. In our experiments we matched polynomials of degree 2, 4, 6 and 8. At first sight, it appears that there is a need to extract from an object a large amount of features, However, as it will be shown in the sequel, not each and every polynomial coefficient is used, and the final number of features is not very large.

### *C. The final touch – adding the fitting error to the feature vector*

The technique described above can improve significantly the performance, however, we can indicate some $specific$ shapes that cause even this technique to fail. We show an example of such shapes below.

Consider the objects "Pillow" and "Square" that appear in Fig. 12. We can see that a polynomial of degree 4 is adequate for modeling "Square" (Fig. 12(a2)). Therefore, we would expect that the coefficients of a polynomial of degree 6 matched to this object (Fig. 12(b2)) to be unstable. A polynomial of degree 4 describing the "Pillow" shape does not differ significantly from the "Square" polynomial (Figs. 12(a1-2)), consequently, the coefficients of these polynomials are

very similar. We perturbed the objects in the same manner as in the previous example, and show in Fig. 13(a) the projections of the resulting clouds. We see clearly that the points belonging to these two objects are mixed. Although the polynomials of degree 6 belonging to ”Square” and ”Pillow” look quite different, their clouds are mixed due to instability (Fig. 13(b)). Therefore, we cannot base the recognition on this degree either.

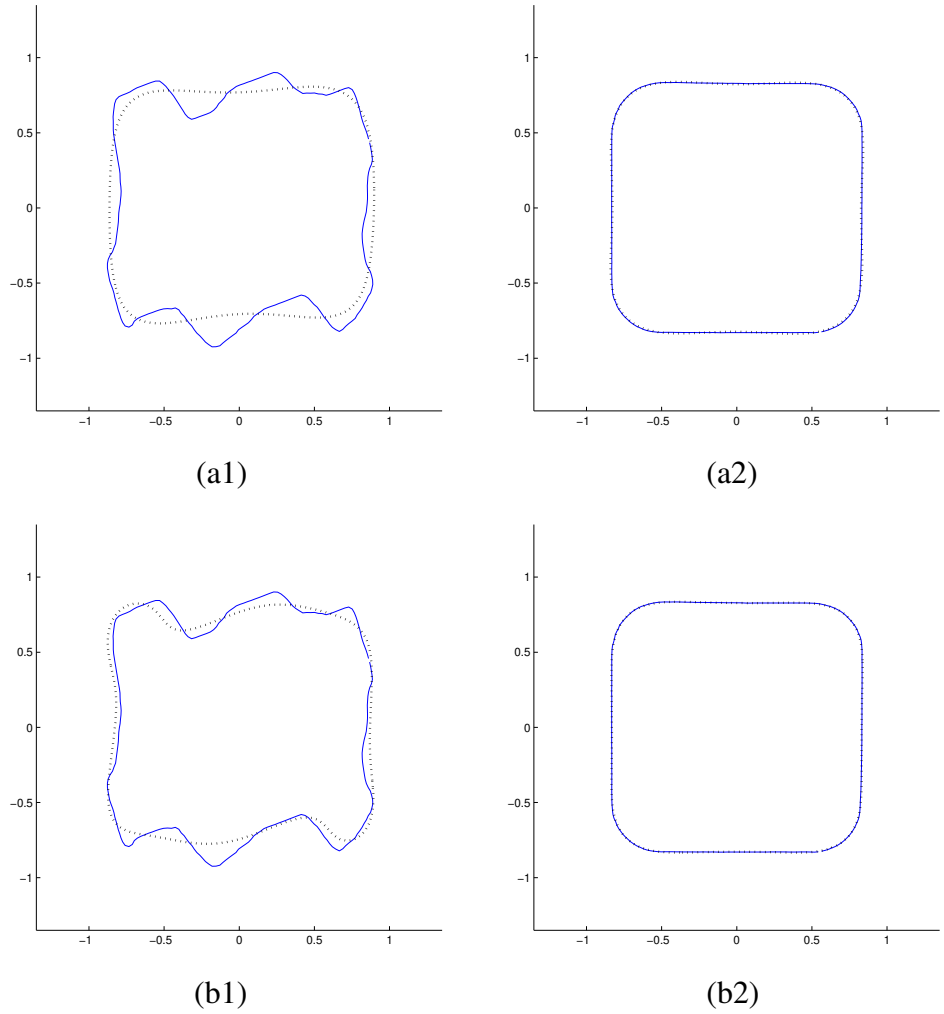
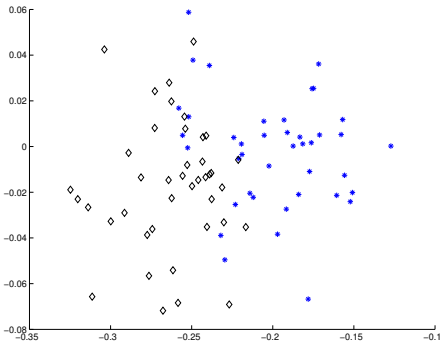
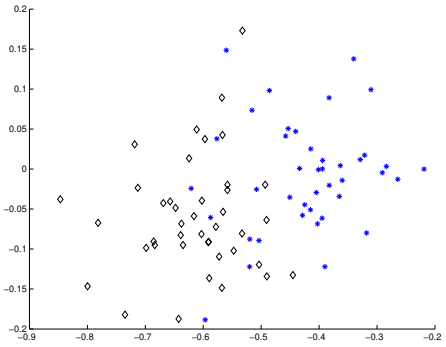



Fig. 12. Left column: ”Pillow”. Right: ”Square”. Solid line: Original data-set. Heavy dotted: Zero-set of fitting polynomial. (a) Matching polynomial is of degree 4, (b) Matching polynomial is of degree 6.

To overcome this obstacle, we propose to extract another feature from the objects. We can see that the polynomial of degree 4 fits well the ”Square” shape, but not so well the ”Pillow”.




(a)




(b)

Fig. 13. Clouds of the coefficients projections: $a_{1,0}$ (y-axis) versus $a_{0,0}$ (x-axis). Diamonds represent "Pillow" data-sets. Asterisks: "Square" data-sets. (a) Coefficients of Gradient-one polynomial of degree 4, (b) Coefficients of Gradient-one polynomial of degree 6.

Therefore, while recognition relying only on polynomial coefficients would have low recognition rate (the resulting polynomials are very similar indeed), a parameter based on *fitting errors* can do this job. After fitting a polynomial $P^r_{\underline{a}}(x,y)$ to the given data-set, we get the vector of the polynomial fitting errors $\underline{e}_{\underline{a}}$ (the length of this vector equals the number of the data-set points). Our aim is to derive from this vector a $robust$ and $informative$ parameter. The measure we propose is $\varepsilon_{75\%}(\underline{e}_{\underline{a}_r})$. Among the elements comprising $\underline{e}_{\underline{a}_r}$, we choose a value greater than 75% of the elements, but smaller than the rest 25%. This way, if there are only a few badly fitted data-set points (as a result of noise), they would not be taken into account, achieving $robustness$. However, if more than 25% of the data-set points were badly fitted, then the $\varepsilon_{75\%}(\underline{e}_{\underline{a}_r})$ measure

would reveal it, providing the $informativeness$. We name the proposed technique which takes advantage of both the coefficients of fitting polynomials of different degrees and fitting errors, Multi Order and Fitting Error Technique (MOFET). The detailed description of the algorithm is given in section VI.

## V. Affine Transform Invariance

Generally, the 2D objects we wish to recognize are projections of 3D objects. Such a projection can be described by a non-linear perspective transform. In general, the relation between two different projections of the same object is not linear. However, in many practical cases it can be approximated by an *affine* transform. That is:

$$\begin{bmatrix} \check{x}_i \\ \check{y}_i \end{bmatrix} = A \begin{bmatrix} x_i \\ y_i \end{bmatrix} + T, \tag{38}$$

where A is an arbitrary 2x2 matrix, T a 2x1 vector and $(\check{x}_i, \check{y}_i)$ and $(x, y)$ are the corresponding points belonging to two different projections of the 3D object. Hence, the ability to recognize a 2D object, even if it undergoes an affine transform, is of high importance.

Examples of using IP-s for affine-invariant recognition appear in [10] and [16]. In [10], a quartic (i.e., a polynomial of degree 4) was fitted to the data-set and then decomposed into three covariant conics (polynomials of degree 2). The resulting invariants were based on intrinsic properties of the conics. In [16] the recognition was based on algebraic affine invariants and was limited to low degree polynomials.

In this work we propose to eliminate the effect of the affine transform (up to rotation) by preprocessing the data-set. We then apply polynomial fitting and use the earlier mentioned rotation invariants [9], achieving full affine-invariance.

The elimination of the effects of the affine transform (up to rotation) is done by normalization of the Scatter Matrix $\mathcal{S}(\underline{x}, \underline{y})$ of the translated to origin data. The use of this concept in combination with different recognition algorithms can be found, for example, in [12] and [13].

The Scatter Matrix is defined as:

$$\mathcal{S}(\underline{x}, \underline{y}) \triangleq \frac{1}{n} \begin{bmatrix} \underline{x}' \\ \underline{y}' \end{bmatrix} \cdot \begin{bmatrix} \underline{x} & \underline{y} \end{bmatrix} = \begin{bmatrix} E(x^2) & E(xy) \\ E(xy) & E(y^2) \end{bmatrix}, \tag{39}$$

where $E(x^2)$, $E(xy)$ and $E(y^2)$ denote second-order $moments$ of the data-set.

The normalization of the matrix $\mathcal{S}$ can be done by multiplying the data-set ($\underline{x}$,$\underline{y}$) by $\mathcal{N}$, where $\mathcal{N}$ is a 2×2 *normalization* matrix such that,

$$\mathcal{N}'\mathcal{N} = \mathcal{S}^{-1} \tag{40}$$

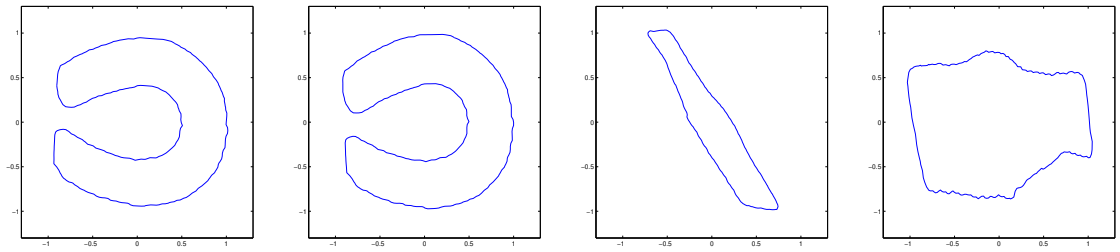
Denoting by $\mathcal{S}_{norm}$ the normalized scatter matrix, we have:

$$\begin{aligned} \mathcal{S}_{norm}(\underline{x},\underline{y}) &= \tfrac{1}{n}\mathcal{N}\begin{bmatrix} \underline{x}' \\ \underline{y}' \end{bmatrix} \cdot \begin{bmatrix} \underline{x} & \underline{y} \end{bmatrix} \mathcal{N}' \\ &= \mathcal{N}\mathcal{S}(\underline{x},\underline{y})\mathcal{N}' = \mathcal{N}(\mathcal{N}'\mathcal{N})^{-1}\mathcal{N}' \\ &= \mathcal{N}(\mathcal{N})^{-1}(\mathcal{N}')^{-1}\mathcal{N}' = I \end{aligned} \tag{41}$$

We denote a shape that has a Scatter Matrix equal to $I$ as a *Mother Shape*.

Note, that the normalization matrix $\mathcal{N}$ is not unique. Multiplication of this matrix by any 2x2 mirroring or rotation matrix R also satisfies equation (40).

In Fig. 14 we show examples of the proposed data-set normalization.



(a) (b) (c) (d)

Fig. 14. (a) "Horseshoe" data-set, (b) A Mother Shape of the "Horseshoe" data-set. (Note that the original Scatter matrix of this data-set is almost $I$, therefore this data-set barely changes.) (c) "Bottle" data-set, (d) A Mother Shape of the "bottle" data-set.

## VI. Recognition process and experiments

In this section we present the proposed recognizer that involves the computation of a Mother-Shape of each object followed by the application of MOFET. We then report the results of several tests that were carried out to examine the effect of different recognizer parameters on recognition performance. Finally, we compare our proposed technique with the Curvature Scale Space (CSS) [18] recognition method adopted by the MPEG7 standard [26].

*A. Recognition process*

Generally, a recognition task consists of two sub-tasks: Recognizer design (i.e., learning) and Recognizer application (i.e., testing). In our work it is assumed that the feature vectors from each set of objects have a Gaussian probability density function (PDF) of feature vectors, where the parameters of each PDF (i.e., the mean vector and covariance matrix) are estimated from feature vectors belonging to Mother Shapes of different views of a corresponding object in the dictionary. Thus, each dictionary object is represented by its PDF. Then, we apply the Maximum Likelihood principle for recognition. I.e., we extract the feature vector from the shape we want to recognize and substitute it into each object's PDF. We then recognize the shape as an object having highest likelihood value. The feature vector we used in our experiments was extracted as follows:

**Feature vector extraction process**

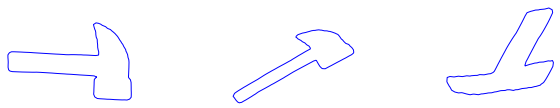
1) Translate the object data-set to the origin and transform it to get a representative Mother-Shape (according to section V).
2) Scale and smooth the derived data-set as described in section III-B
3) Fit degree 2, 4, 6 and 8 polynomials to the obtained data-set and find the fitting error measure $\varepsilon_{75\%}(\underline{e}_{\underline{a}_r})$ (see section IV-C) for each degree. The fitting polynomials can be computed by any one of the following algorithms: Gradient-one, Rotation Invariant Min-Max (RI-Min-Max), Rotation Invariant Min-Var (RI-Min-Var), Modified Rotation Invariant Min-Max (Mod. RI-Min-Max), and Modified Rotation Invariant Min-Var (Mod. RI-Min-Var).
4) Calculate the linear geometric invariants from the obtained polynomial coefficients.

It can be concluded from [9] that a *homogeneous* polynomial of even degree provides one linear invariant. Therefore, an $r$-degree polynomial ($r$ is even), containing $r/2+1$ homogeneous polynomials of even degree, would provide $r/2+1$ linear invariants. Consequently, we can derive 2+3+4+5 = 14 linear invariants from the polynomials fitted to the data-set. We add to these invariants four $\varepsilon_{75\%}(\underline{e}_{\underline{a}_r})$ parameters (one for each degree), obtained in step 3, and get an 18-dimentional *feature vector*.

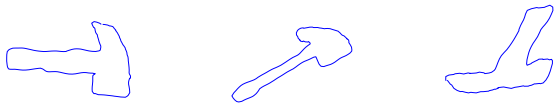
### B. Experimental Database

In our experiments we used the ”Multiview Curve Database” (MCD) created by M. Zuliani [23], containing 40 different shapes taken from 7 different points of view, giving a total of 280 basic shapes. Examples of different views of the same object are shown in Fig. 15a. In order to expand this data-base, we add colored noise to the coordinates of each shape. Each basic shape was perturbed 20 times, thus a database of $40 \times 7 \times 20 = 5600$ different objects was created. Colored noise was added in order to better mimic the changes in boundary shapes due to data acquisition (e.g., segmentation). White noise can model the pixelization noise (i.e., quantization of data-set points location), which is negligible relative to the acquisition noise.

The noise was modeled as follows. First, a vector of Gaussian i.i.d. values, with STD of 0.02 (noise of this strength added relatively strong perturbations to the objects, but the objects could still be distinguished by human observers), having the same length as the data-set, was created. Then, it was ”colored” by a Gaussian shaped FIR window (having a length of 15 and a STD of 0.4 ). This procedure was carried out for each of x and y coordinates of the shape. An example of the effect of these perturbations appear in Fig 15b. Note that because of shape normalization the ”stretched” data-sets are more sensitive to noise then those which undergo a minor normalization.



(a)



(b)

Fig. 15. (a) 3 different views of the shape ”Hammer”, (c) The views perturbed by colored noise with STD = 0.02

### C. Experiments

In our experiments we chose a group of objects from the data-set obtained in VI-B for the recognizer dictionary (i.e., learning data), while the rest (test data ) were recognized using the

obtained dictionary. For learning we used 3 views $\times$ 20 perturbations = 60 data-sets for each of the 40 shapes in the database. The remaining sets (4 sets $\times$ 20 perturbations = 80 data-sets for each shape) were used for the recognition test. Then, the MOFET algorithm was applied and the recognition rate (i.e., the percentage of correctly recognized shapes) was calculated.

In order to get more statistics, we carried out the above process several times, where each time different views were used for learning. Then, the average recognition rate was calculated.

The results of this process for the different recognition algorithms examined are shown in Table I

TABLE I

PERFORMANCE OF MOFET FOR DIFFERENT FITTING ALGORITHMS.

| Grad.One | RI-Min-Max | RI-Min-Var | Mod. RI-Min-Max | Mod. RI-Min-Var |
|---|---|---|---|---|
| 90.2% | 90.1% | 90.5% | 90.1% | **90.6%** |

As can be seen, there is only a slight difference in performance between the different examined algorithms, when applied with MOFET. We choose Mod. RI-Min-Var for other experiments.

We compared the MOFET approach to an algorithm that uses only a single-degree (i.e., using linear invariants of a $single$ fitting polynomial). To do this we performed all four feature-extraction steps of section VI-A and then examined what would happen if we use linear invariants statistics (without the fitting error parameter $\varepsilon_{75\%}(\underline{e}_{\underline{a}_r})$) of only one fitting polynomial (of degree 2, 4, 6 or 8). The results are shown in Table II, proving the effectiveness of the MOFET approach.

In order to show that for effective recognition we need the coefficients of several polynomials as well as the fitting errors, we examined the performance of the recognizer based on a single polynomial along with its fitting error and the performance of the recognizer based on fitting polynomials of degree 2, 4, 6 and 8 , without the fitting error measure (i.e., Multi Order Technique – MOT only). The results appear in Table III

We also performed experiments with 5% missing data. First, we used 6 views $\times$ 20 perturbations = 120 data-sets for each of the 40 shapes in the database. The remaining set (1 set with 20 perturbations for each shape) was randomly occluded and used for the recognition test. As a result we got 91% recognition rate. Then we produced a more complete dictionary, based on

all the 7 views (without occlusion). For the recognition step, the data-sets were perturbed again and randomly occluded. As a result we got 95% recognition rate.

TABLE II

COMPARISON BETWEEN MOFET AND SINGLE-DEGREE APPROACHES.

| MOFET | Single-degree | | | |
|---|---|---|---|---|
| | 2 | 4 | 6 | 8 |
| **90.6%** | 54.5% | **75.1%** | 64.8% | 60.8% |

TABLE III

COMPARISON BETWEEN MOFET, SINGLE-DEGREE ALONG WITH $\varepsilon_{75\%}(\underline{e}_{\underline{a}_r})$ AND MOT APPROACHED.

| MOFET | single degree + $\varepsilon_{75\%}(\underline{e}_{\underline{a}_r})$ | | | | MOT |
|---|---|---|---|---|---|
| | 2 | 4 | 6 | 8 | |
| **90.6%** | 64.1% | **78.5%** | 73.3% | 68.9% | **88.4%** |

Finally, we compared the performance of MOFET with the CSS [18] technique (again, 3 views were used for learning ). We chose this technique, because is has been reported [18] as having better performance then other known techniques, namely, Fourier Descriptors [19], Turning Angle [20], Convex Hull [21], etc. We also compared the techniques in a less noisy environment, i.e., added colored noise of STD = 0.01, which resulted in only slight data-set deformations. The results of the comparisons appear in Table IV. It can be seen that the MOFET algorithm has better performance.

Comparing the computational costs of these techniques, we show in the Appendix that MOFET has also an advantage over CSS, although small, in this aspect too.

TABLE IV
COMPARISON BETWEEN MOFET AND CSS TECHNIQUES.

| STD = 0.02 | | STD = 0.01 | |
|---|---|---|---|
| MOFET | CSS | MOFET | CSS |
| **90.6%** | 87.8% | **99.2%** | 97.0% |

## VII. CONCLUSION

In this work we have improved the performance of existing fitting algorithms in two aspects: representation and recognition. We have shown that replacing the algebraic distances by geometric ones, in the fitting algorithm minimization problem, results in improved fitting of the data-set.

We have also introduced a novel Multi Order (degree) and Fitting Error (MOFET) recognizer that outperforms existing Implicit Polynomial based recognizers. Fitting several polynomials of different degrees and utilization of their coefficients along with the fitting errors made it possible to take advantage both of the stability of low-degree polynomials and informativeness of high-degree polynomials, and thus enabled the design of a high performance recognizer. Making use of the linear invariants designed by Tarel [9], allows the recognition of objects that underwent an Euclidian transform. Although the recognizer based on these invariants does not use all the information provided by the polynomial (i.e., some coefficients don't affect the invariants), it shows good performance.

We addressed also the problem of 3D object recognition via 2D contours. We showed that in circumstances allowing approximations of a projective transform by an Affine transform, exploiting the Mother-shape technique enables effective projection-based recognition of 3D objects.

We used a Gaussian pdf of feature vectors to design a simple low-cost MOFET recognizer based on a ML estimator. This technique showed good performance, although it requires a sufficient amount of different object views in the dictionary.

Finally, we compared the proposed MOFET recognizer to the standard CSS [18] contour based recognition and found MOFET to have a better performance.

## APPENDIX

## COMPARISON BETWEEN CSS AND MOFET COMPUTATIONAL COSTS

In this appendix we compare the computational costs of CSS and MOFET. We first compare the feature extraction computational cost and then – data recognition.

In the CSS approach [18] the data-set is smoothed several times with Gaussians of different kernel width. Assuming that the data consists of N points, the Gaussian has length K and the smoothing is performed Q times, we have *O(NKQ)* computations. This is the leading factor of the CSS features extraction computation. Having 200 points, a Gaussian kernel of length 10 and 600 different kernels, we have about $1.2 \cdot 10^6$ calculations per object data-set.

In the case of MOFET we are dealing with a LS problem, based on N points (samples) and $O(r^2)$ unknown coefficients (r is the order of the polynomial). The computational cost would be $O(Nr^4)$. This is the leading factor of the MOFET features extraction computation. Having N = 200 and r = 8 we get about $0.8 \cdot 10^6$ calculations.

The data recognition in the case of CSS is done by comparison of the CSS image [18] of the unknown data-set with the CSS images of the data-sets in the dictionary. The comparison takes $O(P^3)$, where P is the typical number of $leading\ peaks$ [18] of the data-set. So, the recognition computational cost is $O(RP^3)$, where R is number of the data-sets in the dictionary. In our case we have R = 40 objects $\times$ 60 data-sets in the dictionary per object = 2400. The typical number of leading peaks P was equal to 6. Therefore, in CSS case we need about $5 \cdot 10^5$ calculations.

In the case of MOFET we have an 18-dimentional feature vector , R = 40 candidates and a Gaussian likelihood function. The process would take about $40 \cdot 18^2 \approx 10^4$ calculations.

## REFERENCES


[1] M. M. Blane, Zhibin Lei, "3L Algorithm for Fitting Implicit Polynomial Curves and Surfaces to Data", IEEE Trans. on PAMI, vol.22, No. 3, pp. 298–313, March 2000.

[2] T. Tasdizen, J.P. Tarel and D.B. Cooper, "Improving the Stability of Algebraic Curves for Application", IEEE Trans. on Image Proc., vol.9, No. 3, pp. 405–416, March 2000.

[3] A. Helzer, M. Barzohar and D. Malah, "Robust Fitting of 2D Curves and 3D Surfaces by Implicit Polynoms", IEEE Trans. on PAMI., vol. 26, no. 10, pp. 1283–1294, Oct. 2004.

[4] S. Sallivan, L. Sandford and J. Ponce, "Using geometric distance fits for 3-D object modeling and recognition", IEEE Trans. PAMI, vol. 16, pp. 1183-1196, Dec. 1994.

[5] G. Taubin, "Estimation of Planar Curves, Surfaces and Nonplanar Space Curves Defined by Implicit Equations, with Applications to Edge and Range Image Segmentation", Trans. on PAMI, Vol. 13, No. 11, pp. 1115-1138, Nov. 1991.

[6] D Keren, C Gotsman, ”Fitting curves and surfaces with constrained implicit polynomials ”, IEEE Trans. PAMI, vol. 21, pp. 31-41, Jan. 1999.

[7] C. Unsalan, ”A model based approach for pose estimation and rotation invariant object matching”, Pattern Recognition Letters 28 (2007), pp. 49-57.

[8] D. Keren, ”Using Symbolic Computation to Find Algebraic Invariants”, IEEE Trans. on PAMI, vol. 16, no. 11, pp. 1143-1149, Nov. 1994.

[9] J.-P. Tarel, D.B. Cooper, ”The complex representation of algebraic curves and its simple exploitation for pose estimation and invariant recognition”, Trans. on PAMI, vol. 22, no. 7, pp. 663 - 674, July 2000.

[10] J.-P. Tarel, W. A. Wolovich, D.B. Cooper, ”Covariant-Conics Decomposition of Quartics for 2D Shape Recognition and Alignment”, Journal of Mathematical Imaging and Vision archive Volume 19 , Issue 3, pp. 255 - 273, Nov. 2003.

[11] A. Goshtasby, “Description and Discrimination of Planar Shapes Using Shape Matrices ”. IEEE Trans. on PAMI, vol. 7, pp.738-743, Nov. 1985.

[12] Y. Avrithis, Y. Xirouhakis, S. Kollias, ”Affine-invariant curve normalization for object shape representation, classification, and retrieval”, Machine Vision and Applications, vol. 13, no. 2, pp. 80-94, Nov. 2001.

[13] M. Zuliani, S. Bhagavathy, B. S. Manjunath, C. S. Kenney, ”Affine-Invariant Curve Matching” IEEE International Conference on Image Processing, Singapore, Oct. 2004.

[14] J. Subrahmonia, D. B. Cooper, D. Keren ”Practical reliable Bayesian recognition of 2D and 3D objects usingimplicit polynomials and algebraic invariants”, IEEE Trans. on PAMI, vol. 18, Issue 5, pp. 505-519, May 1996.

[15] L. Zhibin, T. Tasdizen, D. B. Cooper, ”PIMs and invariant parts for shape recognition,” in Proc. 6th Int. Conf. Computer Vision (ICCV’98), Mumbai, India, 1998, pp. 827-832.

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

[17] Daniel Keren, ”Topologically Faithful Fitting of Simple Closed Curves”, IEEE Trans. on PAMI, vol. 26 no.1, pp.118-123, Jan. 2004.

[18] F. Mokhtarian, S. Abbasi, and J. Kittler, ”Robust and Efficient Shape Indexing through Curvature Scale Space”, In BMCV, Edinburgh, UK, 1996.

[19] F. Chaker, MT. Bannour, F. Ghorbel, ”A complete and stable set of affine-invariant Fourier descriptors”, Proc. of International Conference on Image Analysis and Processing, pp. 578-581, Mantova, Italy (2003).

[20] W Niblack, J Yin, ”A pseudo-distance measure for 2D shapes based on turning angle”, Proc. IEEE Int. Cont on Image Processing. pp 352-355, Washington, DC (1995).

[21] Z Yang, FS Cohen, ”Image registration and object recognition using affine invariants and convex hulls”, IEEE Trans. on PAMI, vol. 8, pp. 934-946, July 1999.

[22] A. Blake and M. Isard, *Active Contours*, London, U.K.: SpringerVerlag, 1998.

[23] http://vision.ece.ucsb.edu/∼zuliani/MCD/MCD.shtml

[24] R. Hartley, A. Zisserman, *Multiple View Geoometry in computer vision*, Cambridge University Press, 2000.

[25] A. E. Hoerl and R.W. Kennard, “Ridge Regression: Biased Estimation for Nonorthogonal Problems”, Technometrics, vol. 12, No. 1, pp. 55-66, Feb. 1970.

[26] B. Manjunath, P. Salembier, T. Sikora, *Introduction to MPEG-7*, Wiley, 2002.