Recognizing Mice, Vegetables and Hand Printed Characters Based on Implicit Polynomials, Invariants and Bayesian Methods *

Jayashree Subrahmonia, Daniel Keren and David B. Cooper
Laboratory for Engineering Man/Machine Systems,
Division of Engineering, Brown University, Providence, RI 02912, USA
email: js@lems.brown.edu

Abstract

This paper presents a new robust low-computational-cost system for recognizing freeform objects in 3D range data or in 2D curve data in the image plane. Objects are represented by implicit polynomials (i.e., 3D algebraic surfaces or 2D algebraic curves) of degrees greater than 2, and are recognized by computing and matching vectors of their algebraic invariants (which are functions of their coefficients that are invariant to translations, rotations, and general linear transformations). Implicit polynomials of 4th degree can represent complicated asymmetric free-form shapes. This paper deals with the design of Bayesian (i.e., minimum probability of error) recognizers for these models and their invariants that results in low computational cost recognizers that are robust to noise, partial occlusion, *and other perturbations of the data sets. This work extends the work in [5] by developing and using new invariants for 3D surface polynomials and applying the Bayesian recognizer to operating on invariants.*

1 Introduction

The simplest 2D or 3D recognition problem is that a boundary model is stored in a database for each of L rigid objects. (By an object *boundary*, we mean the 3D object surface, and external and internal boundary curves for a 2D object.) Data along the entire boundary or over a portion of the boundary of an object to be recognized is collected from a sensor. Object recognition is to be realized by determining the stored boundary model that fits the sensed data the best. By best, we mean in the sense of minimum mean squared distance from the data points to a boundary model. This should produce the recognizer functioning with the highest relative frequency of correct recognition. What are the drawbacks to this approach? There are two, both computational. First is that, if there are N data points, order of NL computations must be made for checking on the mean square fits of L stored object boundaries to N data points. This can be considerable if L is large. Second is that the position of the object being sensed will be different than the position of the object in the database and the sensed data may also have undergone a general linear transformation, due, e.g., to the viewing of a target boundary on the ground from an arbitrary aerial viewing direction. Hence, an object in the database has to be rotated and translated in checking its match to the sensed data. This usually means checking a boundary model fit to the data for the boundary model moved to many different positions and linearly transformed, thus incurring a huge amount of computation. The approach presented in this paper avoids both these drawbacks.

*This work was partially supported by NSF Grant #IRI-8715774 and NSF-DARPA Grant #IRI-8905436

2 Recognition Approach

The approach in this paper is to model 3D and 2D objects of interest by algebraic surfaces or curves, respectively, i.e., by the *zero sets* of implicit polynomials. The *zero set* is the set of points (x, y) in 2D (or (x, y, z) in 3D) for which the polynomial function $f(x, y)$ on 2D (or $f(x, y, z)$ on 3D) is zero. Then, a stored model is simply the set of coefficients for the polynomial model. These are global 3D models, unlike explicit polynomials where z is given as an explicit function of x and y as in a depth map. Most of the early work on implicit polynomial curves and surfaces was limited to quadrics, thus dealing with representations that had modest expressive power. Implicit polynomials of degree greater than 2, on the other hand, have great modeling power for complicated objects and can be fit to data very well. In [8] there is presented a very well organized and understandable introduction to these polynomials and some of their properties, and very effective approaches to low computational cost algorithms for fitting these polynomials. Hence we can now use these high degree implicit polynomials for representing complicated objects. In addition, we have developed a technique for fitting polynomials with bounded zero sets, which results in better and more stable description of objects [4].

For an implicit polynomial model, checking the fit of a stored surface or curve to data involves fitting an implicit polynomial to the data, and then comparing the resulting polynomial coefficients with the L coefficient vectors (one for each object) stored in the database. Unfortunately however, there are two problems that need to be solved. This paper presents the solutions to them and the resulting recognizer.

The first problem is that if the object to be recognized is in a different position than the object in the database, the coefficients for the best fitting polynomial to the data will be different than the coefficients

0-8186-3870-2/93 $3.00

for the same object in the database, and hence, one cannot compare the set of coefficients for the best fitting implicit polynomial to the data set with the stored coefficient vectors. Our solution to this problem is to use a vector of algebraic invariants for the recognizer. An algebraic invariant is a function of the implicit polynomial coefficients that is invariant to rotations and translations for 3D surfaces and is invariant to translations and general linear transformations for 2D curves. Thus, the recognizer compares the vector of invariants for the best fitting polynomial to the data set with the stored vectors of invariants. The additional computation for computing the invariants is negligible compared to the time required for fitting the implicit polynomial.

For an extensive survey of invariants in computer vision, see [1, 9]. Here we concentrate on finding *algebraic invariants*, which are related to the description of objects as the zero sets of implicit polynomials [6, 4, 8]. In order to use invariants in the Bayesian based recognition system described in this paper, it was necessary to have invariants that are expressed as *simple explicit* functions of the all the coefficients [7]. In this paper, we describe a new symbolic method that is bound to find all the invariants of the type it is looking for. We know of no other method which can accomplish this. More generally, the only other method we are aware of for finding new algebraic invariants of all the coefficients of high degree implicit polynomials is that of [9].

The second problem that had to be solved is that small changes in a data set often result in large changes in the coefficients of the best fitted polynomial, and, hence, large changes in the algebraic invariants. The reason for this variability is due to the fact that the data used in fitting the polynomials provides constraints among the coefficients of a fitted polynomial, but the data may be insufficient to uniquely determine the coefficients. Hence, since the fitted curve and stored curve coefficients may differ greatly, we cannot compare the curves over the local region of interest based on their coefficients or the invariants, which are functions of these coefficients. Our solution to this problem is to treat recognition as Bayesian statistical recognition in the presence of noisy data, and to use certain asymptotic results which permit a computationally low cost recognizer. The resulting recognizer involves comparison of the measured vector of invariants, but not using the Euclidean distance. *Rather, the error measure requires the use of a weighting matrix which is a function of the specific data set being recognized.* The required computation for computing this matrix is of the order of the number of data points, and hence, modest and suitable for real time recognition. The beauty of this approach is that even though it uses global models — implicit polynomials and coefficient vectors — it behaves as though recognition is based on the matching of a local data set to a boundary model. *Hence, it works excellently even if the data set is over only a portion of the object boundary, which will be the case due to self occlusion if range data is taken for a 3D object from one direction, or which may be the case if one or more objects are partially occluding the object to be recognized.*

3 Using Symbolic Computation to find Invariants

Formally, let a polynomial be denoted by $f(x,y) = \sum_{0\le i<j\le n}^{n} a_{ij}x^iy^j$. Let x and y be subjected to some kind of transformation $(u,v)^t = T(x,y)^t$. Because of space limitations, we address only the case of T = rotation by θ. $f(x,y)$ transforms into a polynomial $g(u,v)$, where g's coefficients, b_{ij}, are functions of the a_{ij}'s and θ. From here on, it will be more convenient to look at the coefficients a_{ij} and b_{ij} as being indexed by a single variable. So let us revise the notations as follows: $f(x,y)$ is determined by the coefficients $\{a_i\}_{i=1}^{i=N}$, and $g(u,v)$ by the coefficients $\{b_i\}_{i=1}^{i=N}$, where each b_i is a function of the a_i's and θ, and $N = \frac{1}{2}(d+1)(d+2)$, where d is the degree of $f(x,y)$.

Now, we *guess* for an invariant I, of a high degree implicit polynomial curve, which is a homogeneous polynomial (or *form*) Φ in the a_i's, e.g. of second degree, so $I = \sum_{0\le i<j\le N} \Phi_{ij}a_ia_j$, which has to be equal to $\sum_{0\le i<j\le N} \Phi_{ij}b_ib_j$.

Let us denote the relation between the a_i's and θ and the b_i's by Λ. Formally,

$$\Lambda : \mathcal{R} \times \mathcal{R}^N \rightarrow \mathcal{R}^N$$

Here $\Lambda(\theta, \mathbf{a}) = \mathbf{b}$, where θ is the rotation angle, $\mathbf{a}$ the vector of coefficients of the polynomial $f(x,y)$, and $\mathbf{b}$ the vector of coefficients of $g(u,v)$.

The following has to hold for every coefficient vector $\mathbf{a}$ and every angle θ:

$$\Phi(\mathbf{a}) = \sum_{0\le i<j\le N} \Phi_{ij}a_ia_j = \sum_{0\le i<j\le N} \Phi_{ij}b_ib_j = \Phi(\mathbf{b})$$

Theorem 1 *For the above to hold – e.g., for the form Φ to define an invariant – it is necessary and sufficient that for every* $\mathbf{a}$,

$$(\frac{\partial}{\partial\theta}\Phi[\Lambda(\theta,\mathbf{a})])_{\theta=0} = 0 \qquad (1)$$

Proof : [3, 5].

The following is a simple program written in the *Mathematica* language [10] to find invariants under rotation for a second degree form. Similar programs for finding more general invariants of higher degree polynomials consist of thousands of lines and were written using a lexical analyzer [3, 5]..

1) $a20x^2 + a11xy + a02y^2$
2) $\%/.x- > u - v * z$ (% *means "previous expression"*).
3) $\%/.y- > v + u * z$
(/. *means "substitute", here "substitute* $v + u * z$ *for* y).
4) $expl = Expand[\%]$
5) $b20 = D[expl, \{u, 2\}]/2$
6) $b02 = D[expl, \{v, 2\}]/2$

7) $b11 = D[exp1, \{u, 1\}, \{v, 1\}]$
8) $exp2 = Expand[A * b20^2 + B * b11^2 + C * b02^2 + D * b20 * b11 + E * b20 * b02 + F * b11 * b02]$
9) $exp3 = A * a20^2 + B * a11^2 + C * a02^2 + D * a20 * a11 + E * a20 * a02 + F * a11 * a02$
10) $exp4 = Expand[exp2 - exp3]$
11) $D1 = Expand[D[exp4, \{a20, 2\}]]$
12) $D2 = Expand[D[exp4, \{a11, 2\}]]$
13) $D3 = Expand[D[exp4, \{a02, 2\}]]$
14) $D4 = Expand[D[exp4, \{a20, 1\}, \{a11, 1\}]]$
15) $D5 = Expand[D[exp4, \{a20, 1\}, \{a02, 1\}]]$
16) $D6 = Expand[D[exp4, \{a11, 1\}, \{a02, 1\}]]$
17) $eq1 = (Coefficient[D1, z, 1] == 0)$
18) $eq2 = (Coefficient[D2, z, 1] == 0)$
19) $eq3 = (Coefficient[D3, z, 1] == 0)$
20) $eq4 = (Coefficient[D4, z, 1] == 0)$
21) $eq5 = (Coefficient[D5, z, 1] == 0)$
22) $eq6 = (Coefficient[D6, z, 1] == 0)$
23) $Solve[\{eq1, eq2, eq3, eq4, eq5, eq6\}, \{A, B, C, D, E, F\}]$

First, the original form is given (line 1). Then, x and y are replaced by u and v, which are the first order Taylor approximation of the rotated x, y coordinate system (these are enough, as we need to compute only the first derivative at zero). The angle of rotation is denoted by z. In Lines 4-7, the coefficients of the new form are computed. In lines 8-9, we guess for invariants which are second degree forms in the coefficients; their difference – denoted $exp4$ – has to be zero. Next, the coefficients of z are isolated by differentiating $exp4$ first by the $p_{i,j}$'s and then by z (lines 11-22). Equating these coefficients to zero gives a set of equations which is solved in line 23; the result *Mathematica* outputs is the following:

$$\{A \rightarrow \mathrm{C}, B \rightarrow \frac{\mathrm{C}}{2} - \frac{E}{4}, \mathrm{D} \rightarrow 0, F \rightarrow 0\}$$

which correspond to the invariants $p_{20}^2 + \frac{1}{2}p_{11}^2 + p_{02}^2$ and $p_{20}p_{02} - \frac{1}{4}p_{11}^2$. The great advantage of this method is that it is bound to find *all* invariants of the type it is looking for. In [3, 5], we generalize this approach to finding Euclidean and affine invariants that are higher degree polynomials in the coefficients.

4 Asymptotic parameter distribution and Bayesian Recognition

Let $\boldsymbol{\alpha}$ denote the vector of coefficients of the polynomial $f(x, y, z)$ that describes the given object. We assume that the range data points $Z_1, Z_2, \ldots, Z_n$ are statistically independent, noisy Gaussian measurements of the object surface in the direction perpendicular to the boundary at its closest point. This model is introduced and discussed in [2, 7, 5]. Thus, the joint probability of the data points, $\mathbf{Z}^n = (Z_1, Z_2, \ldots, Z_n)$, is

$$p(\mathbf{Z}^n \mid \boldsymbol{\alpha}) = \frac{1}{(2\pi\sigma^2)^{\frac{n}{2}}} \exp[-\frac{1}{2\sigma^2} \sum_{i=1}^{n} \frac{f^2(Z_i)}{\| \nabla f(Z_i) \|^2}] \quad (2)$$

Being able to write this joint probability for a data set in terms of a complicated curve or surface is an important result and permits the application of a large range of tools from statistics and probability theory. The maximum likelihood estimate $\hat{\boldsymbol{\alpha}}_n$ of $\boldsymbol{\alpha}$ given the data points is the value of $\boldsymbol{\alpha}$ that maximizes (2). A very useful tool for solving the problems of object recognition and parameter estimation is an asymptotic approximation to the joint likelihood function, (2), which can be shown to have a Gaussian shape in $\boldsymbol{\alpha}$ [2, 7], i.e.,

$$p(\mathbf{Z}^n \mid \boldsymbol{\alpha}) \approx [p(\mathbf{Z}^n \mid \hat{\boldsymbol{\alpha}}_n)] \exp\{-\tfrac{1}{2}(\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}_n)^t \Psi_n (\boldsymbol{\alpha} - \hat{\boldsymbol{\alpha}}_n)\} \quad (3)$$

where Ψ_n is the second derivative matrix having i, jth component $-\frac{\partial^2}{\partial\alpha_i \partial\alpha_j} ln\, p(\mathbf{Z}^n \mid \boldsymbol{\alpha}) \mid_{\boldsymbol{\alpha}=\hat{\boldsymbol{\alpha}}_n}$. Hence, all the useful information about $\boldsymbol{\alpha}$ is summarized in the quadratic form in the exponent of equation (3). If Ψ_n is not singular, then it is the inverse covariance matrix of $\hat{\boldsymbol{\alpha}}_n$. The matrix Ψ_n is called the Fisher Information matrix of $\hat{\boldsymbol{\alpha}}_n$. Various extremely useful generalizations of (3) are developed in [7].

The asymptotic approximation (3) gives an understanding of the extent to which the data constrains the coefficients of the best fitting polynomial [5]. The next section deals with using this approximation for designing a metric based on the geometric invariants for comparing two polynomial zero sets over the region where the data exists.

4.1 Mahalanobis distance between two sets of Invariants

The scenario for recognition that we consider in this paper is one where we have a set of objects labeled $l = 1, 2, \ldots, L$ in the database, all modeled by polynomials of the same degree. Let $\mathbf{G}_l$ denote the vector of invariants for the polynomial describing object l. Then, given a set of range data points, $\mathbf{Z}^n = \{Z_1, Z_2, \ldots, Z_n\}$, the optimum recognition rule is 'choose l for which $p(\mathbf{Z}^n \mid \mathbf{G}_l)$ is maximum'. Thus, the recognition problem reduces to computing the likelihood of the data given $\mathbf{G}$. In [7], we have shown that

$$p(\mathbf{Z}^n \mid \mathbf{G}) \approx [p(\mathbf{Z}^n \mid \hat{\boldsymbol{\alpha}}_n)] (2\pi)^{\frac{d_\mathbf{H}}{2}} \left|\Psi_n^\mathbf{H}\right|^{-\frac{1}{2}} \exp\{-\tfrac{1}{2}(\mathbf{G} - \hat{\mathbf{G}}_n)^t \Psi_n^\mathbf{G} (\mathbf{G} - \hat{\mathbf{G}}_n)\} \quad (4)$$

where $\Psi_n^\mathbf{G}$ and $\Psi_n^\mathbf{H}$ are the Information matrices of the vector of invariants and a vector of nuisance parameters, respectively, and $d_\mathbf{H}$ is the number of nuisance parameters. The matrix $\Psi_n^\mathbf{G}$ is given by

$$\Psi_n^\mathbf{G} = (D\mathbf{G})^{\dagger t} \mid_{\boldsymbol{\alpha}=\hat{\boldsymbol{\alpha}}_n} \Psi_n (D\mathbf{G})^\dagger \mid_{\boldsymbol{\alpha}=\hat{\boldsymbol{\alpha}}_n}$$

where $\dagger$ implies pseudo-inverse and $D(\mathbf{G})$ is the Jacobian of the transformation from $\boldsymbol{\alpha}$ to $\mathbf{G}$.

Using (4) for the simplest case of recognition, the optimum recognition rule becomes – 'Choose l for which the Mahalanobis distance, $(\mathbf{G}_l - \hat{\mathbf{G}}_n)^t \Psi_n^\mathbf{G} (\mathbf{G}_l - \hat{\mathbf{G}}_n)$, is minimum.' This is because, the only part of (4) that is a function of l is $\exp\{-\frac{1}{2}(\mathbf{G}_l - \hat{\mathbf{G}}_n)^t \Psi_n^\mathbf{G} (\mathbf{G}_l - \hat{\mathbf{G}}_n)\}$.

The beauty of this recognizer is that the computational cost is negligible, but the recognizer is equivalent

to checking how well the data fits the models stored in the database for different linear transformations of the models for which the computational cost is enormous.

In summary, object recognition using invariants is done as follows.

1. Fit the best polynomial to the data set.
2. Compute the invariants $\hat{\mathbf{G}}_n$ which are functions of the coefficients of the polynomial.
3. Compute the Mahalanobis distance, $(\mathbf{G}_l - \hat{\mathbf{G}}_n)^t \Psi_n^{\mathbf{G}} (\mathbf{G}_l - \hat{\mathbf{G}}_n)$ to each object in the database and pick the l for which it is a minimum.

The computational cost for steps 1-3 is linear in the number of data points. The computation of the Mahalanobis distance for 200 points and a 4th degree curve is a fraction of a second on a SPARC 10 and can be sped up by orders of magnitude with parallel architectures.

5 Experimental Results

The experiments illustrate the use of the Mahalanobis distance in the space of invariants for recognizing 2D and 3D objects from real data that may be partial and that is noisy. The experiments also illustrate the fact that the Mahalanobis distance has better discriminatory power than does the Euclidean distance.

The first set of experiments illustrate the performance of the recognizer for 3D objects. The objects in this experiment are keyboard mice. Figure 1 shows the four mice used in this experiment. Figures 2(a)-(d) are the data sets and the polynomial fits for the mice in standard position. (The polynomial fits were obtained using our approach for fitting bounded polynomials). The data sets were obtained using the Brown and Sharpe Microval Manual coordinate measuring machine. All the data sets are well fit by fourth degree polynomials in x, y, z. These are the four objects in the database.

Figures 3(a)-(d) are the data sets and polynomial fits for the rotated and translated versions of the mice in the database. We used 7 invariants for a fourth degree polynomial in x, y, z. All of them are listed in [3]. The goal in this experiment is to recognize the mice in Figures 3(a)-(d) using the Mahalanobis distance measure and compare the results with those using the Euclidean distance.

Tables 1 and 2 show the Mahalanobis and the Euclidean distances, respectively, between the vector of invariants for the polynomial fits to the rotated mice in Figure 3 and the vectors of invariants for the four mice in the database. The Mahalanobis distance measure does a great job of discriminating between the right object and the rest. Also, the Mahalanobis distance has much better discriminatory power than does the Euclidean distance.

The next experiment illustrates the use of the Mahalanobis distance for recognizing 2D and 3D objects from partial data.

Figure 4 shows the partial data (with the polynomial fit superimposed) for the mouse in Figure 2(a). The partial data in this experiment is what a stereo sensor would see when looking at the mouse from a point near the bottom left corner. The Mahalanobis and Euclidean distances between the vector of invariants for the polynomial fit to the occluded object and the stored vectors of invariants are :

Mahalanobis distance:
Mouse1 : 1.0 Mouse2 : 1065
Mouse3 : 30.31 Mouse4 : 1.004
Euclidean distance:
Mouse1 : 1.0 Mouse2 : 18.39
Mouse3 : 1.619 Mouse4 : 0.901

The Mahalanobis distance to Mouse1 is the smallest. However, the Mahalanobis distance to Mouse4 is almost the same as that to Mouse1. This is because the occluded data does not contain the curved front part of Mouse1, and since that is the part that really distinguishes Mouse1 from Mouse4, it is hard to distinguish between them based on the partial data. The distances to Mouse2 and Mouse3 are large compared to those to Mouse1 and Mouse4. The Euclidean distance does not give good recognition results with partial data. In fact, the Euclidean distance from the occluded object to Mouse4 is smaller than that to Mouse1. Recognition based on the Mahalanobis distance using invariants should produce good recognition whenever the sensed data points fit the correct stored model significantly better than the other stored models.

The data sets for the 2D examples are handwritten characters. The objects in the database are the handwritten characters, 'a', 'q', 'g' and 'w', shown in Figures 5(a)-(d). The data sets are well fit by fourth degree polynomials in x, y. Figure 6(a) is another instance of the handwritten character 'w' that is a rotated, translated, occluded and noisy version of the one in the database. We fit a fourth degree polynomial to the occluded object in order to compare its invariants with those for the unoccluded database objects. Figure 6(b) is the fourth degree polynomial fit to the data set in 6(a). Three invariants for a fourth degree polynomial in x,y obtained using our approach are

1. $3a_{13}^2 - 8a_{04}a_{22} + 2a_{13}a_{31} + 3a_{31}^2 - 32a_{40}a_{04} - 8a_{22}a_{40}$,
2. $3a_{04}^2 + 2a_{04}a_{22} + a_{13}a_{31} + 2a_{04}a_{40} + 2a_{22}a_{40} + 3a_{40}^2$,
3. $a_{22}^2 - 3a_{13}a_{31} + 12a_{04}a_{40}$,

Since the invariants should be independent of multiplication of the coefficients by a constant, these three functions yield only two invariants. One set of two invariants is $\frac{g_1}{g_3}$ and $\frac{g_2}{g_3}$. Thus, for object recognition, we use the Mahalanobis distance between the ratios of invariants. The Mahalanobis distance from the letter in Figure 6(a) to the letters 'a', 'g', 'q' and 'w' in the database are :

'a':1.00 'q':12.9 'g':12.2 'w':4.81

(The distance to 'a' in the database is normalized to have value 1.0.) The distance to 'w' is minimum. However, the distance is small to 'a' and 'q'. This is because, the data set in 6(a) also fits the model for 'a' and 'q' as shown in Figures 6(c) and 6(d). The experiment illustrates that even under a large amount of occlusion, the recognizer comes up with the best possible results.

References

[1] Geometric Invariance in Machine Vision, J. Mundy and A. Zisserman editors, MIT Press. 1992.

[2] R.M. Bolle and D.B. Cooper. On Optimally Combining Pieces of Information, With Applications to Estimating 3D Complex-object Position From Range Data. IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(5):619–638, September 1986.

[3] D. Keren. Some New Invariants in Computer Vision. 1992. Under review for publication in IEEE Transactions on Pattern Analysis and Machine Intelligence.

[4] D. Keren, D. B. Cooper, and J. Subrahmonia. Describing Complicated Objects by Implicit Polynomials. Technical Report LEMS-102, Brown University, February 1992. Accepted for publication as a regular paper in the IEEE Transactions on Pattern Analysis and Machine Intelligence.

[5] D. Keren, J. Subrahmonia, D. B. Cooper, and G. Taubin. Bounded and Unbounded Implicit Polynomial Curves and Surfaces,Mahalanobis Distances, and Geometric Invariants, for Robust Object Recognition. In Proceedings: Image Understanding Workshop, pages 769–778, San Diego, January 1992.

[6] D.J. Kriegman and J. Ponce. On Recognizing and Positioning Curved 3D Objects from Image Contours. IEEE Trans. on Pattern Analysis and Machine Intelligence, 12:1127–1138, December 1990.

[7] J. Subrahmonia, D. B. Cooper, and D. Keren. Practical Reliable Bayesian Recognition of 2D and 3D Objects Using Implicit Polynomials and Algebraic Invariants. Technical Report LEMS-107, Brown University, May 1992. Under review for publication in the IEEE Transactions on Pattern Analysis and Machine Intelligence.

[8] G. Taubin. Estimation of Planar Curves, Surfaces and Nonplanar Space Curves Defined by Implicit Equations, with Applications to Edge and Range Image Segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 13:1115—1138, November 1991.

[9] G. Taubin and D. B. Cooper. 3D Object Recognition and Positioning with Algebraic Invariants and Covariants. pages 147–182, July 1990. A chapter in Symbolic and Numerical Computations-Towards Integration, pages 147–182, B. R. Donald, D. Kapur and J. Mundy editors, Academic Press, 1992.

[10] S. Wolfram. Mathematica, a System for Doing Mathematics by Computer. Addison-Wesley, 1988.

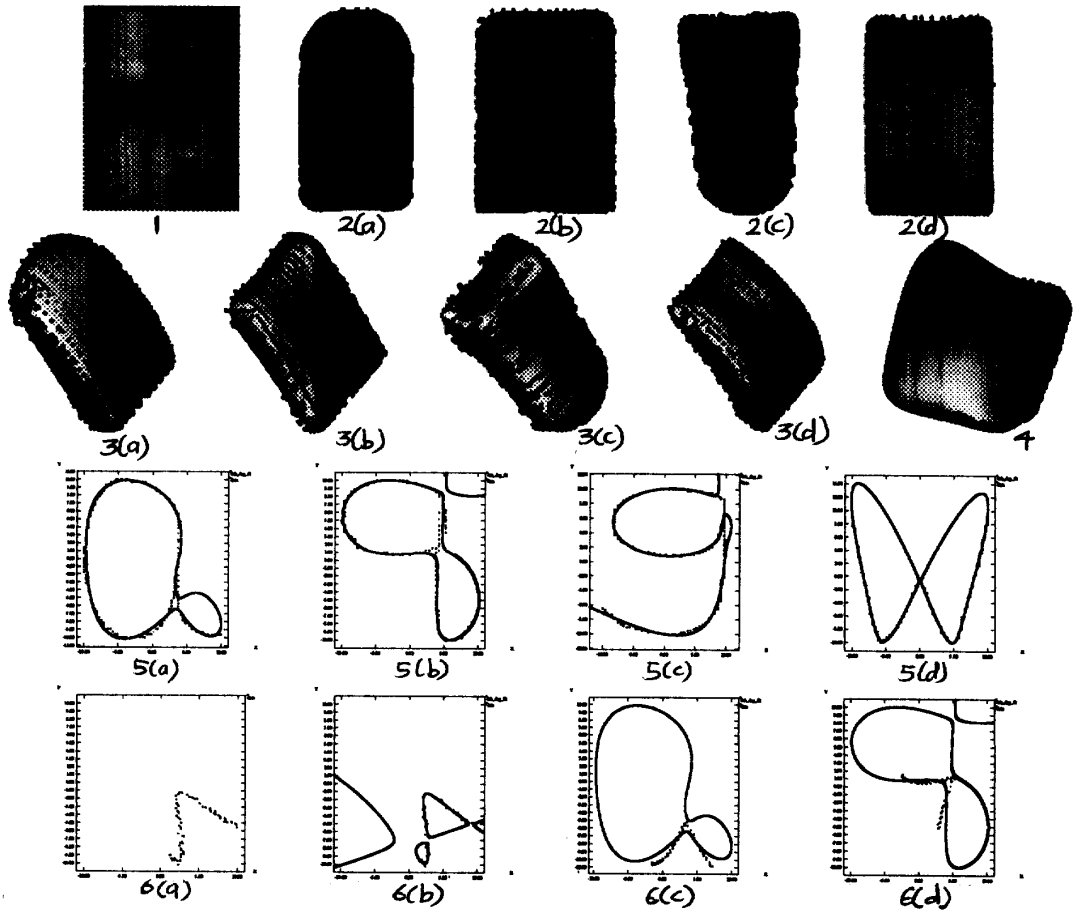

TABLE 1 (Mahalanobis distances) :

	Mouse1 (rotated)	Mouse2 (rotated)	Mouse3 (rotated)	Mouse4 (rotated)
Mouse1	1.000e+00	3.517e+02	1.379e+01	3.519e+00
Mouse2	2.463e+01	1.000e+00	1.560e+02	2.109e+01
Mouse3	2.622e+02	3.717e+03	1.000e+00	1.489e+02
Mouse4	2.872e+00	2.818e+03	6.096e+01	1.000e+00

TABLE 2 (Euclidean distances) :

	Mouse1 (rotated)	Mouse2 (rotated)	Mouse3 (rotated)	Mouse4 (rotated)
Mouse1	1.000e+00	2.535e+00	9.614e+00	2.741e+00
Mouse2	1.822e+00	1.000e+00	1.125e+00	1.547e+00
Mouse3	1.351e+01	2.322e+00	1.000e+00	7.568e+00
Mouse4	1.446e+00	1.327e+00	2.435e+01	1.000e+00