# The Complex Representation of Algebraic Curves and its Simple Exploitation for Pose Estimation and Invariant Recognition

Jean-Philippe Tarel

David B. Cooper

*LCPC*
*58, Boulevard Lefebvre*
*75732 Paris, France*
*tarel@lcpc.fr*

*LEMS, Division of Engineering*
*Brown University, Box D*
*Providence, RI, 02912-9104, USA*
*cooper@lems.brown.edu*

**Abstract**

New representations are introduced for handling 2D algebraic curves (implicit polynomial curves) of arbitrary degree in the scope of computer vision applications. These representations permit fast accurate pose-independent shape recognition under Euclidean transformations with a *complete set of invariants*, and fast accurate pose-estimation based on all the polynomial coefficients. The latter is accomplished by a new centering of a polynomial based on its coefficients, followed by rotation estimation by decomposing polynomial coefficient space into a union of orthogonal subspaces for which rotations within two dimensional subspaces or identity transformations within one dimensional subspaces result from rotations in $x, y$ measured-data space. Angles of these rotations in the two dimensional coefficient subspaces are proportional to each other and are integer multiples of the rotation angle in the $x, y$ data space. By recasting this approach in terms of a complex variable, i.e, $x + iy = z$ and complex polynomial-coefficients, further conceptual and computational simplification results. Application to shape-based indexing into databases is presented to illustrate the usefulness and the robustness of the complex representation

of algebraic curves.

**Keywords:** Complex polynomials, pose estimation, pose-independent curve recognition, Euclidean invariants, complete-sets of rotation invariants, curve centers, implicit polynomial curves, algebraic curves, shape representation, shape recognition.

# 1 Introduction

For shape recognition involving large databases, position-invariant 2D shape-recognition and pose-estimation have to be performed by fast algorithms providing robust accurate estimates subject to noise, missing data (perhaps due to partial occlusion) and local deformations. There is a sizeable literature on alignment and invariants based on moments [1], B-splines [2], superquadrics [3], conics [4], combinations of straight lines and conics, bitangeants [5], differential invariants [6, 7, 8, 9], and Fourier descriptors. Two observations are: these two problems (pose estimation and pose-independent recognition) are often studied independently; though the preceding approaches have their own significant strengths and handle certain situations very well, the two problems –pose-independent recognition and pose-estimation– are unsolved if there is large noise and large shape deformation present, there is missing data, and maximum estimation speed and estimation accuracy are important. This paper presents an approach based on algebraic (also referred to as implicit polynomial) curve models which meets these requirements for pose estimation and for pose-invariant object recognition.

2D algebraic curves of degrees 4 or 6 are able to capture the global shape of curve data of interest (see Fig. 1). However, our primary interest in algebraic curves in this paper is that they have unparalleled features crucial to fundamental computer vision applications. First, we derive a complete set of invariants for fast pose-invariant shape recognition. By a complete set of independent invariants, we mean that it is possible to reconstruct, without ambiguity, the algebraic curve shape from the set of invariants only. Since this set specifies the shape in a unique way, these invariants can be used as "optimal" shape descriptors. (Of conceptual interest is that this set of invariants, defined in the paper, is not necessarily complete algebraically). Second, algorithms are given in the paper which permit single-computation pose estimation, and slightly slower but more accurate iterative pose estimation based on all the polynomial coefficients. These features are due to the following contributions.

1. A complex basis is introduced for the space of coefficient vectors leading to the complex representation of algebraic curves of degree $n$, where $n$ is arbitrary. The components of the basis vectors are complex numbers, even though the resulting polynomial is still real. This provides a representation from which we derive a complete set of rotation-invariants. We fully describe how real and complex vector representations are related.

   The complex basis arises not from consideration of the geometry of the algebraic curve but rather from consideration of the geometry of the transformation of its coefficients and is built on the fact that when the $(x, y)$ data set is rotated, the resulting coefficient vector undergoes an orthogonal transformation [1].

2. A new accurate estimate of an "intrinsic center" for an algebraic curve, which is based on all of the polynomial coefficients. The algebraic curve can then be centered by moving its intrinsic center to the origin of the data coordinate system. This centering is invariant to any prior translations a shape may have undergone. Computing the center requires a single computation followed by a few iterations.

3. Pose-invariant shape recognition is realized by centering an algebraic curve, as in 2., and then basing shape recognition on the complete set of rotation-invariant shape descriptors indicated above in 1.

4. Fast pose-estimation. Estimating the Euclidean transformation, that has taken one shape data-set into another using all the polynomial coefficients is realized by: initial translation estimation as the difference in the estimated intrinsic centers, based on 2., of the two curves; this is followed by rotation estimation, based on 1.; and is completed by one or two iterations of translation estimation followed by rotation estimation, where coefficients for the two polynomials are compared using the representation in 1.

What is most important in the preceding methodology is that estimators used are linear or slightly nonlinear functions, which are iterated a few times, of the original polynomial coefficients, thus being stable. Highly nonlinear functions of polynomial coefficients, which have been used previously, usually are not as robust and repeatable.

Note, the invariant representations we use with algebraic curves are global. At this stage of the research, we do not know if these representations are well suited to highly-accurate finely-discriminating shape recognition and therefore look upon the recognition, when dealing with *very very* large image databases, to be used for the purpose of indexing into these databases. The idea is to reduce the number of images that must be considered in the database by a large factor using our invariant recognition, and then do more careful comparison on the shapes that remain. This more careful comparison would involve pose estimation for alignment followed by careful comparison of aligned shape data. This careful comparison of aligned shapes could then be done through our PIMs measure [10] or through other measures.

How do Fourier descriptors compare with the algebraic curve model? Fourier descriptors, like algebraic curves, provide a global description for shapes from which pose and recognition can be processed. But the Fourier approach has difficulty in general with open patches or is restricted to star shapes, depending on the parameterization used. In particular, when dealing with missing data, small extra components, and random perturbations, heuristic preprocessing must be applied to the curve data in order to close and clean it, and then arc-length normalization problems arise in the comparison between shapes. This is also the case with curvature descriptors [11]. In matching open curves having inaccurately known end points, both of these approaches require extensive computation for aligning starting and stopping points.

For algebraic 2D curves and 3D surfaces, the most basic approach to comparison of two shapes is iterative estimation of the transformation of one algebraic model to the other followed by recognition based on comparison of their coefficients or based on comparing the data set for one with the algebraic model for the other [12, 1, 10]. But the problem of initialize this iterative process still remains. A major jump was the introduction of intrinsic coordinate systems for pose estimation and Euclidean algebraic invariants for algebraic 2D curves and 3D surfaces [1, 6]. These are effective and useful, but as published do not use all the information in the coefficients.

The present paper is an expansion of the complex representation first presented in [13]. A later paper [14] presented a partial complex representation for algebraic curves for obtaining some recognition invariants related to our complete set of invariants, and for pose estimation based on only a few polynomial coefficients. It also uses a different center for an algebraic curve. Moreover, the authors are not concerned with concepts developed in this paper such as complex bases and invariant subspaces, complete sets of invariants, and pose estimation using and combining information available in all the polynomial coefficients.

In Sec. 2, we introduce the decomposition of the coefficient space with two examples: conics and cubics under rotation. This leads to the *complex representation* of algebraic curves. Then, in Sec. 3, the proposed pose estimation technique is described with validation experiments. Sec. 4 is dedicated to recognition with invariants. The proposed recognition algorithm is applied in the context of indexing into a database of silhouettes, where algebraic representations allow us to easily handle missing parts along the contour.

# 2 Algebraic Curve Model

## 2.1 Definition

An algebraic curve is defined as the zero set of a polynomial in 2 variables. More formally, a 2D implicit polynomial (IP) curve is specified by the following polynomial of degree $n$:

$$\begin{aligned} f_n(x,y) = \textstyle\sum_{0\le j,k,j+k\le n} a_{jk}x^j y^k = \underbrace{a_{00}}_{H_0} + \underbrace{a_{10}x + a_{01}y}_{H_1} + \underbrace{a_{20}x^2 + a_{11}xy + a_{02}y^2}_{H_2} + \\ \underbrace{a_{30}x^3 + a_{21}x^2y + a_{12}xy^2 + a_{03}y^3}_{H_3} + \ldots + \underbrace{a_{n0}x^n + a_{n-1\,1}x^{n-1}y + \ldots + a_{0n}y^n}_{H_n} = 0 \end{aligned} \tag{1}$$

Here $H_r(x,y)$ is a *homogeneous binary polynomial (or form)* of degree $r$ in $x$ and $y$. Usually, we denote by $H_n(x,y)$ the leading form. An algebraic curve of degree 2 is a conic, degree 3 a cubic, degree 4 a quartic, and so on.

Polynomial $f_n$ is conveniently represented by coefficient vector $A$, having components $(a_{jk})$, $0 \le j,k, \;\; j+k \le n$ (number of coefficients is $\frac{1}{2}(n+1)(n+2)$), as:

$$f_n(x,y) = Y^T A$$

where

$$\begin{aligned} A &= [\,a_{00} \quad a_{10} \quad a_{01} \quad a_{20} \quad a_{11} \quad a_{02} \quad \ldots \quad a_{n0} \quad \ldots \quad a_{0n}\,]^T \\ Y &= [\,1 \quad x \quad y \quad x^2 \quad xy \quad y^2 \quad \ldots \quad x^n \quad \ldots \quad y^n\,]^T \end{aligned}$$

Superscript $T$ denotes matrix transpose.

## 2.2 A Useful Basis for Conics and Cubics under Rotation

We first consider the representation of conics and cubics under rotation in order to exhibit properties we want to exploit. A cubic curve is defined by 10 coefficients:

$$\begin{aligned} f_3(x,y) = a_{00} + (a_{10}x + a_{01}y) + (a_{20}x^2 + a_{11}xy + a_{02}y^2) \\ +(a_{30}x^3 + a_{21}x^2y + a_{12}xy^2 + a_{03}y^3) = 0 \end{aligned} \tag{2}$$

When a cubic is rotated through angle $\theta$, the 10 coefficients $(a_{ij})$, $0 \le i, j, i+j \le 3$, are transformed as a messy function of $\theta$. The rotation matrix $R(\theta)$ for the data is:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{3}$$

which specifies the counter-clockwise rotation of the curve by $\theta$ radians, equivalently the clockwise rotation of the coordinate system by $\theta$ radians. The original cubic coefficients are vector $A$ and the transformed one is $A'$. We denote with a prime the representation after transformation. By substituting (3) in (2), and after expansion, we obtain the linear relation between the two vectors $A' = LA$, where the $10 \times 10$ matrix $L$ is a function of the rotation angle only. This matrix can be put into a block diagonal form as shown

$$L = \begin{bmatrix} 1 & & & 0 \\ & L_1 & & \\ & & L_2 & \\ 0 & & & L_3 \end{bmatrix}$$

where the block $L_j$ transforms the coefficients of the homogeneous polynomial of degree $j$, i.e the $j^{th}$ form. Therefore, the size of the block $L_j$ is $(j+1) \times (j+1)$. We have $L_1 = R(\theta)$,

$$L_2 = \begin{bmatrix} c^2 & -cs & s^2 \\ 2cs & c^2 - s^2 & -2cs \\ s^2 & cs & c^2 \end{bmatrix}, \; L_3 = \begin{bmatrix} c^3 & -c^2 s & cs^2 & -s^3 \\ 3c^2 s & c^3 - 2cs^2 & -2c^2 s + s^3 & 3cs^2 \\ 3cs^2 & 2c^2 s - s^3 & c^3 - 2cs^2 & -3c^2 s \\ s^3 & cs^2 & c^2 s & c^3 \end{bmatrix}$$

The elements of these blocks are non-linear functions of $c = \cos\theta$ and $s = \sin\theta$. For a second degree form, to put things into a form exhibiting invariance and simple dependence on angle $\theta$, we define a new parameterization, $\alpha_{20}$, $\beta_{20}$, $\gamma_{11}$, of the coefficients $a_{20}$, $a_{11}$, $a_{02}$ of the polynomial, by applying the following matrix transformation, $N_2$:

$$\begin{bmatrix} \alpha_{20} \\ \beta_{20} \\ \gamma_{11} \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}}_{N_2} \begin{bmatrix} a_{20} \\ a_{11} \\ a_{02} \end{bmatrix}$$

These new parameters $\alpha_{20}$, $\beta_{20}$ and $\gamma_{11}$ are linear functions of the original polynomial coefficients. With this new representation, the matrix $L_2$ is mapped into a matrix where $2\theta$ appears:

$$N_2 L_2 N_2^{-1} = \begin{bmatrix} c^2 - s^2 & -2cs & 0 \\ 2cs & c^2 - s^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} R(2\theta) & 0 \\ 0 & 1 \end{bmatrix}$$

That is, $[\,\alpha'_{20}\,\beta'_{20}\,\gamma'_{11}\,]^T = N_2 L_2 N_2^{-1}\,[\,\alpha_{20}\,\beta_{20}\,\gamma_{11}\,]^t$. The reason for this $\alpha$, $\beta$ notation is that $\alpha_{jk}$ and $\beta_{jk}$ are the real and imaginary parts, respectively, of the complex coefficient $c_{jk}2^{j+k}$ introduced in the next section. When $k = j$, the complex coefficient $c_{jj}$ is always real, and we have $\gamma_{jj} = c_{jj}2^{2j-1}$. For $L_3$, it turns out that a similar simplification is possible with the transformation $N_3$:

$$\begin{bmatrix} \alpha_{30} \\ \beta_{30} \\ \alpha_{21} \\ \beta_{21} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 3 & 0 & 1 & 0 \\ 0 & 1 & 0 & 3 \end{bmatrix} \begin{bmatrix} a_{30} \\ a_{21} \\ a_{12} \\ a_{03} \end{bmatrix}$$

and $L_3$ is mapped into:

$$N_3 L_3 N_3^{-1} = \begin{bmatrix} R(3\theta) & 0 \\ 0 & R(\theta) \end{bmatrix}$$

In summary, when a cubic is rotated, there exists a natural basis determined by the square matrices $(N_j)$, $1 \leq j \leq 3$ where, $L$ is mapped into diagonal $2 \times 2$ and $1 \times 1$ sub block form:

$$B' = \begin{bmatrix} 1 & & & & & \\ & R(\theta) & & & 0 & \\ & & R(2\theta) & & & \\ & & & 1 & & \\ & 0 & & & R(3\theta) & \\ & & & & & R(\theta) \end{bmatrix} B \tag{4}$$

The coefficient vector of the cubic in the new basis is $B$, and $B'$ after rotation $R(\theta)$. It is clear that in this new basis, the coefficient space is decomposed into a union of orthogonal one or two dimensional subspaces invariant under rotations. More specifically, the vector $B = [\gamma_{00}\, \alpha_{10}\, \beta_{10}\, \alpha_{20}\, \beta_{20}\, \gamma_{11}\, \alpha_{30}\, \beta_{30}\, \alpha_{21}\, \beta_{21}]^T$ is decomposed into 2D vectors $[\alpha_{jk}\, \beta_{jk}]^T$ which rotate with angles $\theta$, $2\theta$, or $3\theta$. This leads directly to a simple and stable way to compute the relative orientation between cubics, namely, estimate $\theta$ by comparing the angle $\phi_{ij}$ of the pairs of coefficients $[\alpha_{jk}\, \beta_{jk}]^T$ in $B$ with their transformations in $B'$. Moreover, it is easy to compute a complete set of independent invariants under rotation for a conic:

- 2 linear invariants (i.e., linear functions of the IP coefficients): coefficients $\gamma_{00} = a_{00}$ and $\gamma_{11} = a_{20} + a_{02}$,
- 2 quadratic invariants (i.e., second degree functions of the coefficients): squared radiuses $\alpha_{10}^2 + \beta_{10}^2 = a_{10}^2 + a_{01}^2$, and $\alpha_{20}^2 + \beta_{20}^2 = (a_{20} - a_{02})^2 + a_{11}^2$, of the 2D vectors $[\alpha_{10}\, \beta_{10}]^T$ and $[\alpha_{20}\, \beta_{20}]^T$, respectively.
- and 1 relative angle: the angle between $\phi_{20}$ and $2\phi_{10}$, i.e, $arctan(\beta_{20}/\alpha_{20}) -$

$2arctan(\beta_{10}/\alpha_{10})$, which is $arctan(a_{11}/(a_{20}-a_{02})) - 2arctan(a_{01}/a_{10})$.

Note, to understand the angular invariant, under coordinate system rotation of $\theta$ we see from (4) that $\phi_{20}$ transforms to $2\theta+\phi_{20}$ and $2\phi_{10}$ transforms to $2\theta+2\phi_{10}$. Hence, the angular difference is invariant to rotations. For a cubic, we complete the set of independent invariants with:

- 2 quadratic invariants: squared radiuses $\alpha_{30}^2+\beta_{30}^2 = (a_{30}-a_{12})^2+(a_{21}-a_{03})^2$, and $\alpha_{31}^2+\beta_{31}^2 = (3a_{30}+a_{12})^2+(a_{21}+3a_{03})^2$,
- and 2 relative angles: the angle between $\phi_{30}$ and $3\phi_{21}$, and between $\phi_{10}$ and $\phi_{21}$.

In order to generalize this approach to IPs of arbitrary degree, we turn to complex numbers and thus the *complex representation* of IPs.

### 2.3 Complex Representation of Algebraic Curves

Since we are dealing with rotations and translations of 2D curves, complex representation provides a simplification in the analysis and implementation of pose estimation or pose-invariant object recognition. Given the polynomial $f_n(x,y)=\sum_{0\le j,k,j+k\le n} a_{jk}x^jy^k$, the main idea is to rewrite $f_n(x,y)$ as a real polynomial of complex variables $z=x+iy$ and $\bar{z}=x-iy$:

$$f_n(x,y)=f_n(z)=\sum_{0\le j,k,\ j+k\le n}\frac{a_{jk}}{i^k2^{j+k}}(z+\bar{z})^j(z-\bar{z})^k$$

Using binomial expansions for $(z+\bar{z})^j$ and $(z-\bar{z})^k$, we rewrite $f_n(z)$ with new complex coefficients $c_{jk}$:

$$f_n(z)=\sum_{0\le j,k,\ j+k\le n} c_{jk}\bar{z}^jz^k \tag{5}$$

Notice that coefficients $(c_{jk})$ are complex linear combinations of the $(a_{jk})$, and that $\bar{c}_{jk} = c_{kj}$ since the polynomial is real. We call the vector $C = (c_{jk})$ the *complex vector representation* of an algebraic curve which is defined by a real polynomial in $\bar{z}$ and $z$:

$$f_n(z) = \bar{Z}^T C = Y^T A = 0$$

where $Z = [1\ z\ \bar{z}\ z^2\ z\bar{z}\ \bar{z}^2\ z^3\ z^2\bar{z}\ z\bar{z}^2\ \bar{z}^3\ z^4 \ldots \bar{z}^n]^T$ is the vector of complex monomials. With this notation, $j+k$ specifies the degree of the multinomial in $z$ and $\bar{z}$ associated with $c_{jk}$. Notice that the sub-set of polynomials in $z$ only is the well-known set of harmonic polynomials.

For example, the complex representation of a conic is $f_2(z) = c_{00} + c_{10}\bar{z} + \bar{c}_{10}z + c_{20}\bar{z}^2 + c_{11}\bar{z}z + \bar{c}_{20}z^2$, since $c_{00}$ and $c_{11}$ are real numbers. From the previous section, it is easy to show that $\gamma_{00} = c_{00}$, $\alpha_{10} = 2Re(c_{10})$, $\beta_{10} = 2Im(c_{10})$, $\alpha_{20} = 4Re(c_{20})$, $\beta_{20} = 4Im(c_{20})$, and $\gamma_{11} = 2c_{11}$, where $Re(.)$ and $Im(.)$ are the real and imaginary parts of the expression within parenthesis. The complex representation of a cubic is $f_3(z) = c_{00} + 2Re(c_{10}\bar{z}) + Re(c_{20}\bar{z}^2) + c_{11}|z|^2 + Re(c_{30}\bar{z}^3) + Re(c_{31}\bar{z}|z|^2)$, and so on.

The principal benefit of the vector complex representation is the very simple way in which complex coefficients transform under a rotation of the polynomial curve. Indeed, we see that if the IP shape is rotated through angle $\theta$ (see (3)), $z$ transforms as $z' = e^{i\theta}z$, so that $z = e^{-i\theta}z'$, and by substituting in (5):

$$f_n(z) = \sum_{0 \le j,k,j+k \le n} e^{i(j-k)\theta} c_{jk} \bar{z}'^j z'^k = f'_n(z')$$

Hence, the coefficients of the transformed polynomial are

$$c'_{jk} = e^{i(j-k)\theta} c_{jk} \tag{6}$$

Moreover, as presented in the appendix, there is a recursive and thus fast way to compute the matrix providing the transformation of a given polynomial coefficient vector $A$ to the new basis $C$ for any degree (it is the $N_j$ matrices introduced in the previous section).

# 3 Pose Estimation

As described in the previous section, the relation between the coefficients $C$ of a polynomial and $C'$ of the polynomial rotated is particularly simple when using the complex representation, allowing us to compute the difference of orientation between two given polynomial curves, $C$ and $C'$ (see (6)). It turns out that the complex representation also has nice properties under translation, allowing pose estimation under Euclidean transformation in a very fast way and using *all the polynomial coefficients*. At present, we view the most computationally-attractive approach to pose estimation to consist of two steps. 1) Compute an intrinsic center for each algebraic curve based on the coefficients of its IP representation. This generalizes [1] to optimally use all the information in the coefficients. It is an iterative process with only a few iterations. 2) Center each algebraic curve at the origin of the coordinate system (i.e., move the intrinsic center to the origin of the coordinate system), and then compute the rotation of one algebraic curve with respect to the other based on the coefficients of their IP representations. This optimally uses all the information about the rotation in the coefficients. It is not an iterative process. Note, the translation of one curve with respect to another is then given in terms of the difference in their intrinsic centers. Hence, we are treating location and rotation estimation separately in different ways. When processing speed is less important, maximum accuracy under Euclidean transformations is achieved by following the preceding by a few iterations as discussed in Sec. 3.4.

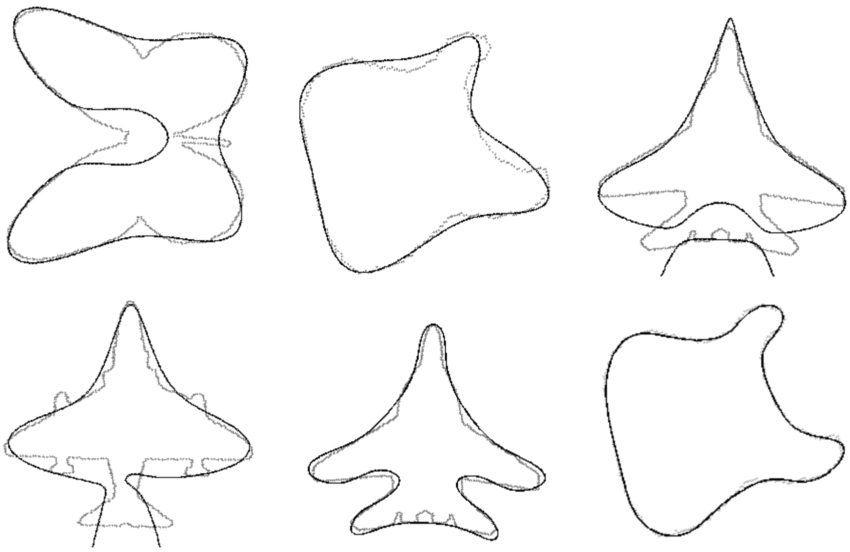
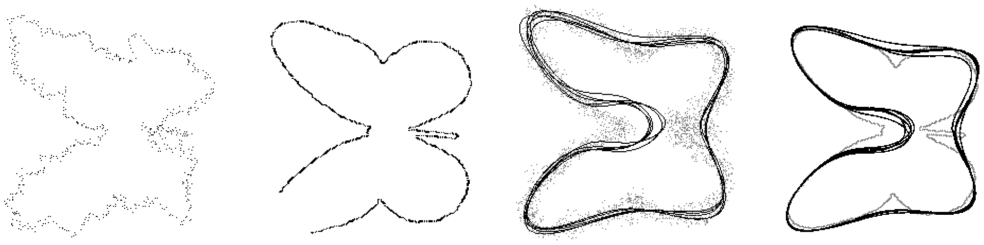


Figure 1: 3L fits of $4^{th}$ degree polynomials to a butterfly, a guitar body, a mig 29 and a sky-hawk airplane. These are followed by two $6^{th}$ degree polynomials fits.

### 3.1 Implicit Polynomial Fitting

Since object pose estimation and recognition are realized in terms of coefficients of shape-modeling algebraic curves, the process begins by fitting an 2D implicit polynomial to a data set representing the 2D shape of interest. For this purpose, we use the gradient-one fitting [15], which is a least squares linear fitting of a 2D explicit polynomial to the data set where the gradient of the polynomial at any data point is soft-constrained to be perpendicular to the data curve and to have magnitude equal to 1. This fitting is of lower computational cost and



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

Figure 2: In (a), the original data set is perturbed with a colored Gaussian noise along the normal with a standard deviation of 0.1 for a shape size of 3 (equivalently, the data is contained in a box having side length 375 pixels and the noise has 12.5 pixels standard deviation). In (b), 10% of the curve is removed. In (c), 5 $4th$ degree fits are superimposed with associated noisy data sets each having standard deviation of 0.1. In (d), 5 fits are superimposed when 10% of the curve is removed at random starting points.

has better polynomial estimated-coefficient repeatability than all previously existing IP fitting methods [15]. This algorithm is an improved version of the 3L fitting [16], taking advantage of the ridge regression approach. The algebraic curve is the zero set of this explicit polynomial. A side benefit of use of the gradient-one soft-constraint is that the fitted polynomial is then normalized in the sense that the polynomial multiplicative constant is uniquely determined.

Fig. 1 shows measured curve data and the fit obtained. This fitting is numerically invariant with respect to Euclidean transformations of the data set, and stable with respect to noise and a moderate percentage of missing data as shown in Fig. 2. $4^{th}$ degree fits allows us to robustly capture the global shape, and higher degree polynomials provide more accurate fits as shown in Fig. 1 with $6^{th}$ degree polynomials.

## 3.2 Translation

It is well known that a non degenerate conic has a center. Given two conics, the two centers are very useful for estimating the relative pose since each conic can be centered before computing the relative orientation. The goal of this section is to compute a stable center in the complex representation with similar properties for polynomials having any degrees.

From (5), if $z$ is transformed as $z' = e^{i\theta}(z+t)$ (i.e translated by $t$ and rotated with an angle $\theta$), we have: $z = e^{-i\theta}z' - t$, and

$$f_n(z) = H_n(e^{-i\theta}z' - t) + H_{n-1}(e^{-i\theta}z' - t) + \ldots = f'_n(z')$$

After expansion, we obtain $H'_n(z')$, the transformed leading form:

$$H'_n(z') = \sum_{0 \le j,k \le n,\ j+k=n} c_{jk} e^{i(j-k)\theta} \bar{z}'^j z'^k$$

Consequently, complex coefficients $(c'_{jk})$ of the transformed leading form $H'_n$ are unaffected by translation in the Euclidean transformation. Continuing expansion, we obtain the transformed next-highest degree form $H'_{n-1}(z')$ having the coefficients $(c'_{jk})$, $0 \le j,\ k \le n,\ \ j+k = n-1$:

$$\begin{aligned} H'_{n-1}(z') = & \sum_{0 \le k \le n-1} c_{n-1-k\,k} e^{i(n-1-2k)\theta} \bar{z}'^{n-1-k} z'^k \\ & -t \sum_{1 \le k \le n} c_{n-k\,k} k e^{i(n+1-2k)\theta} \bar{z}'^{n-k} z'^{k-1} \\ & -\bar{t} \sum_{0 \le k \le n-1} c_{n-k\,k}(n-k) e^{i(n-1-2k)\theta} \bar{z}'^{n-1-k} z'^k \end{aligned}$$

These coefficients are linear functions of the translation component $t$:

$$c'_{n-1-k\,k} = e^{i(n-1-2k)\theta}(c_{n-1-k\,k} - (k+1)c_{n-1-k\,k+1}t - (n-k)c_{n-k\,k}\bar{t}) \tag{7}$$

The first interesting property of (7) is that the term depending on the angle is a multiplicative factor in this set of equations. This means that, given any polynomial, the translation

$t_{linearcenter} = t$ which minimizes the linear least squares problem:

$$\sum_{0 \le k \le n-1} |c_{n-1-k\,k} - (k+1)c_{n-1-k\,k+1}t - (n-k)c_{n-k\,k}\bar{t}|^2 \tag{8}$$

does not depend on the rotation applied to the polynomial. Therefore, the algebraic curve can be translated by $t_{linearcenter}$ to center it at the origin of the coordinate system. This centering is invariant to any Euclidean transformation the algebraic curve may have originally undergone. This center is not different than the Euclidean center of a polynomial derived with the real representation of IPs in [1].

Even if $t_{linearcenter}$ is computed by solving a linear system as above, it is not using all the information available about the curve location, in particular, coefficients $c_{jk}, j < n-1$, are not involved. To use these, we proceed as follows. Compute $t_{linearcenter}$. Translate the polynomial, having coefficient vector $C$, by $t_{linearcenter}$. The resulting polynomial coefficient vector $\tilde{C}$ will be independent of any previous translations the original data set may have undergone (in practice *approximately* independent due to measurement noise and other deviations from the ideal). Now recenter the $\tilde{C}$ polynomial to obtain $(\tilde{C})^*$ by computing and using translation $t_{center} = \tilde{t}$ for which $\| \ (\tilde{C}) \ \|^2$ is minimum. Note, $\tilde{C}$ is an $nth$ degree polynomial in $\tilde{t}$, so we compute $\tilde{t}$ iteratively by using a first order Taylor series approximation and thus only the linear $\tilde{t}$ monomials in $(\tilde{C})$. All the $\tilde{c}_{jk}$ are used in this computation, which is why $t_{center}$ has smaller variance than does $t_{linearcenter}$. Generally, the optimum $\| \ \tilde{t} \ \|$ is small and only 2 or 3 iterations are needed to converge to the $\tilde{t}$ minimizing $\| \ \tilde{C} \ \|^2$. This defines the center $t_{center}$ which we use. Note, this $t_{center}$, determined solely by curve coefficients $C$, is of sub-optimal accuracy because we do not weight the components of $C$ in an optimal way to achieve maximum likelihood or minimum mean square error estimates.

The $t_{center}$ of a high degree polynomial has the same property as the conic center, namely, it is covariant with the Euclidean transformation applied to the data. Consequently, it can be used in the same way for comparing polynomial coefficients: each polynomial $C$ and $C'$ is centered by computing $t_{center}$ and $t'_{center}$, before computing the relative orientation using all the transformed coefficients.

For illustration, consider the simple case of a conic. For a conic $f_2(z) = c_{00} + 2Re(c_{10}\bar{z}) + 2Re(c_{20}\bar{z}^2) + c_{11}|z|^2$, the set of equations (7) is reduced to $c'_{10} = e^{i\theta}(c_{10} - c_{11}t - 2c_{20}\bar{t})$ and its conjugate. The well known center of a conic is defined as the point for which the linear terms vanish, i.e, its position satisfies $c_{10} - c_{11}t_{linearcenter} - 2c_{20}\bar{t}_{linearcenter} = 0$. This $t_{linearcenter}$ is different from the more stable $t_{center}$ introduced before which is the one minimizing $2 \mid c_{10} - c_{11}t_{center} - 2c_{20}\bar{t}_{center} \mid^2 + \mid c_{00} - 2Re(c_{10}\bar{t}_{center}) + c_{11}t_{center}\bar{t} - 2Re(c_{20}\bar{t}^2_{center}) \mid^2$. Notice that this criterion involve all the IP coefficients of the conic.

The second advantage of the complex representation is that we can derive not only one center but several, a different one for each summand in (8), with the same covariance to Euclidean Transformations. The property used here is that $c'_{n-1-k\,k}$ depends only on $c_{n-1-k\,k}$, $c_{n-k\,k}$, and $c_{n-k-1\,k+1}$, i.e, the complex representation decouples the rotation and the translation which is not the case with the coefficients $a_{jk}$. For every equation in (7), we are able to define a new Euclidean center for the IP as $t_k$ for which the right side is 0, as done in the conic case. Consequently, an IP of degree $n$ has $[\frac{n}{2}]$ extra centers $t_k$ ($[u]$ denotes the greatest integer not exceeding $u$), defined by:

$$c_{n-1-k\,k} - (k+1)c_{n-1-k\,k+1}t_k - (n-k)c_{n-k\,k}\bar{t}_k = 0 \tag{9}$$

A nice property of these centers for two curves is that they match one to another without a matching search problem since we know the degree $k$ associated with the center $t_k$. Hence, given two polynomials $C$ and $C'$ of degree $n$, after the computation of the $[\frac{n}{2}]$ centers for each polynomial, approximative but simple pose estimation is determined by $[\frac{n}{2}]$ matched points. Pose estimation of $4^{th}$ degree algebraic curves can be solved by doing the pose estimation between two centers, pose estimation of $6^{th}$ degree polynomials by doing the pose estimation between two triangles, and so on. Of course, these centers do not use all the pose information contained in the polynomial coefficients, so this pose estimation can be used either for fast computation or for a first approximation to maximum accuracy Euclidean pose estimation.

## 3.3 Rotation

Experimentally, it turns out that the center is very stable in the presence of noise and small perturbations (see Fig. 3). The computation of the rotation is in practice less accurate. Therefore, it is important to take advantage of all the information available in the polynomial to obtain the most accurate orientation estimation possible. Assume that the two polynomials are each centered by using Euclidean center $t_{center}$ defined in the previous section.

For a cubic, under rotation $R(\theta)$, $C$ transforms to the vector $C' = [c_{00}, e^{i\theta}c_{10}, e^{i2\theta}c_{20}, c_{11}, e^{i3\theta}c_{30}, e^{i\theta}c_{21}]^T$. In this equation, as seen in Sec. 2.2, complex coefficients $c_{10}$ and $c_{21}$ are rotated by angle $\theta$, $c_{20}$ by angle $2\theta$, and $c_{30}$ by angle $3\theta$. We use all of this information to estimate $\theta$.

In the general case, from (6), $C'$ as a function of $C$ under a rotation $\theta$ is given by $c'_{jk} = c_{jk}e^{i(j-k)\theta}$ where $0 \leq j, k, \;\; j+k \leq n$. Given $C$ and $C'$, we simply used least squares to estimate

$\theta$:

$$\min_{\theta} \sum_{0 \le j,k,\ j+k \le n} |c'_{jk} - c_{jk} e^{i(j-k)\theta}|^2 \tag{10}$$

which leads to maximization of $\sum |c_{jk}||c'_{jk}|cos((j-k)\theta + arg(c_{jk}) - arg(c'_{jk}) - 2\pi l_{jk})$. $l_{jk}$ is an unknown integer when $j-k \neq 1$, and $\frac{2\pi l_{jk}}{j-k}$ is the unknown phase. Integer $l_{jk}$ is between 0 and $j-k-1$. It is inserted to make the argument of the cosine close to 0, thus permitting the cosine to be well approximated by its second order Taylor expansion. Then an explicit approximated solution is derived:

$$\theta = \frac{1}{\sum w_{jk}} \sum w_{jk} \frac{arg(c'_{jk}) - arg(c_{jk}) + 2\pi l_{jk}}{j-k} \tag{11}$$

where weights $w_{jk}$ are $(j-k)^2|c_{jk}||c'_{jk}|$ (These weights can be easily used to test if an IP has $\pi$, $\frac{2\pi}{3}$, $\frac{\pi}{2}$, or any other kind of symmetries. Consequently, we are not limited to non-symmetric shapes in computing the pose estimate). The obtained solution is a good approximate solution, and a few iterations may be used to obtain the closest sub-optimal solution. An optimal estimate can be obtained using a Bayesian formulation and iterative globally optimizing techniques. In this case, the summands in (10) would be weighted by an appropriate inverse covariance matrix.

Since the computational cost is very small, the best estimated angle can be obtained by computing the estimate for all possible integers $l_{jk}$ and choosing the one which minimizes the $w_{jk}$ weighted standard deviation of $\frac{arg(c'_{jk}) - arg(c_{jk}) + 2\pi l_{jk}}{j-k} - \theta$. An even faster alternative is to get a good first estimate of $\theta$ by combining $arg(c'_{jk}) - arg(c_{jk})$ when $j-k=1$ or by using the centers defined with (9).

## 3.4 Estimation of Euclidean transformations

To estimate the Euclidean transformation between two shapes:

| | noise 0.1 | noise 0.2 | 10% missing data | 20% missing data |
|---|---|---|---|---|
| $\theta$ | 5.7% | 72.1% | 2.9% | 37.8% |
| translation | 5.9% | 13.4% | 3.0% | 6.0% |

Table 1: Standard deviation in percentage of the average of the angle and the norm of one translation component, with various perturbations for object in Fig. 2. Added colored Gaussian data noise has standard deviations 0.1 and 0.2 (12.5 and 25 pixels respectively). Occlusions are 10% and 20% of the curve at random starting points. Statistics are for 200 different random perturbations of each kind on the original shape data. As in Fig. 3, true rotation is 1 radian, true translation is 1. (Data lies in box having side-length 375 pixels).

- First each polynomial is centered by computing its center $t_{center}$ using information in all the coefficients of $f_n$ as discussed in Sec. 3.2.
- Then the rotation alignment is performed by using information in all the coefficients of $f_n$ using (11) and the discussion in Sec. 3.3.
- A first estimate of the translation and rotation are the displacement from one center to the other, and the rotation alignment.
- To remove remaining small translation and rotation estimation errors, the translation and the orientation alignment are iterated one or two times by minimizing the sum of squared errors between the two sides of (7) and for all the other $c_{jk}$, most of which involve higher degree monomials in $t$ and $\bar{t}$. This estimation of translation and rotation jointly results in maximum accuracy.

The proposed pose estimation is numerically stable to noise and a moderate percentage of missing data as illustrated in Fig. 3. The pose estimation error due to missing data increases nicely in the range from 0 to 15% (19 pixels). Similar results are obtained in the range $[0, 0.12]$ for the standard deviation of the noise. The added noise is a colored noise [15], i.e, a Gaussian noise in the direction normal to the shape curve at every point and then averaged along 10 consecutive points. We want to emphasis the fact that even if a noise standard deviation of 0.1 (equivalently, 12.5 pixels), is only 3% of the size of shape of the butterfly, this value of the noise represent a very large perturbation of the shape as shown on Fig. 2(a). For greater amounts of noise or missing data, we have the well know threshold effect [17] in estimation problems as shown in Table 1. It arises in our problem because of the nonlinear computations in the angle estimation.

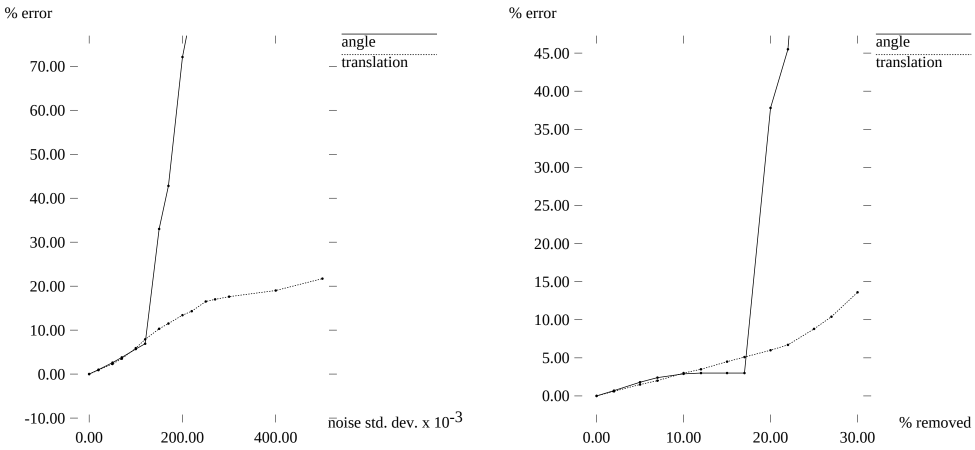



Figure 3: Left, variation of the standard deviation of the angle and the $x$ component of the translation as a function of increasing colored noise standard deviation. Right, variation as a function of increasing percentage of missing data. Plotted values are std. deviation of the error as a percentage of the average values of the pose components. Measurements based on 200 realizations. True rotation is 1 radian, translation is 1.

# 4 Recognition Using Invariants

In this section we solve the pose-independent shape recognition problem based on invariants arising from the complex representation.

## 4.1 Stable Euclidean Invariants

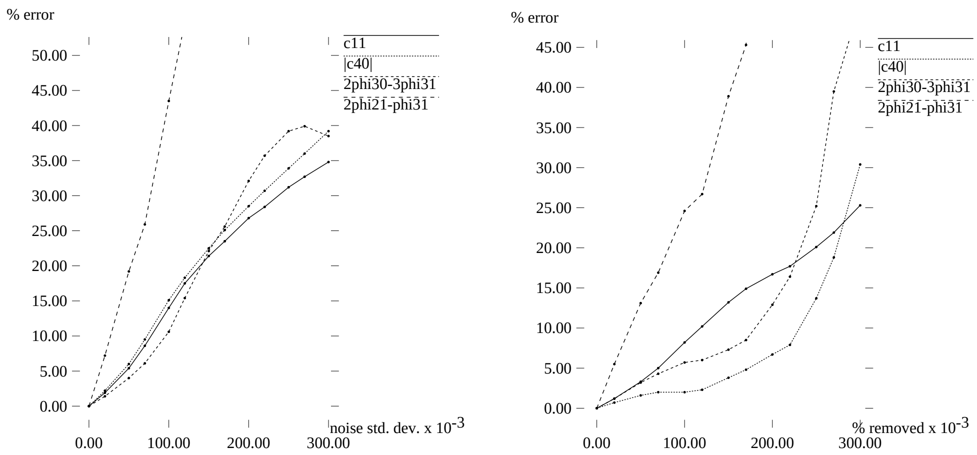



Figure 4: Left, variation of the standard deviations of invariants $|c_{40}|$, $c_{11}$, $|2\phi_{30} - 3\phi_{31}|$, and $|2\phi_{21} - \phi_{31}|$ as a function of increasing colored Gaussian noise std. deviation ($\phi_{jk}$ denotes $arg(c_{jk})$). Right, variation of the standard deviation of the same invariants as a function of an increasing percentage of missing data at random starting points. Values are std. dev. of the error as a percentage of the average value of the invariant. 200 realizations of the shape data were used.

When the IP is centered with the computation of the Euclidean center as described previously in Sec. 3.2, we have canceled the dependence of the polynomial on translation, and the only remaining unknown transformation is the rotation.

| | noise 0.1 | noise 0.2 | 10% missing data | 20% missing data |
|---|---|---|---|---|
| $\lvert c_{40}\rvert$ | 15.1% | 28.5% | 2.0% | 6.7% |
| $c_{11}$ | 14.0% | 26.8% | 8.2% | 16.7% |
| $\lvert 2\phi_{30} - 3\phi_{41}\rvert$ | 10.6% | 32.1% | 5.7% | 12.9% |
| $\lvert 2\phi_{21} - \phi_{31}\rvert$ | 43.5% | 81.4% | 24.6% | 90.0% |

Table 2: Standard deviations as a percentage of the means of a few invariants in response to various data perturbations. Gaussian colored noise has standard deviations 0.1 or 0.2. Occlusions are 10% or 20% of the curve at random starting points. Statistics for each case are computed from 200 different random realizations.

Since the number of coefficients of $f_n$ is $\frac{1}{2}(n+1)(n+2)$ and the number of degrees of freedom of a rotation is 1, the counting argument indicates that the number of independent geometric invariants [18] is $\frac{1}{2}(n+1)(n+2)-1$. We directly have $[\frac{n}{2}]+1$ linear invariants which are $c_{jj}$. From (6), we deduce that all other $|c_{jk}|^2$ are invariants under rotations. The number of these independent quadratic invariants ($2^{nd}$ degree functions of the $c_{jk}$) is the number of complex $c_{jk}, j \neq k$. This number is $o = ([\frac{n}{2}]+1)[\frac{n}{2}]$ for even degrees and $o = ([\frac{n}{2}]+1)[\frac{n}{2}]+[\frac{n+1}{2}]$ for odd degrees. Invariants $|c_{jk}|$ are geometric distances, but there are angles which also are Euclidean invariants for an IP. Indeed, the $\frac{o(o+1)}{2}$ relative angles $(l-m)arg(c_{jk})-(j-k)arg(c_{lm})$ are preserved under rotations. We can choose a maximal independent subset of these relative angles and these along with the preceding linear and quadratic invariants provides a complete set of independent rotation invariants for an IP of degree $n$, as for the cubic case in Sec. 2.2. We want to emphasis the fact that the obtained invariants are linear, quadratic, or arctan functions of ratios of linear combinations of coefficients, even for high degree polynomials. This leads to invariants less sensitive to noise than are others such as algebraic invariants which are rational functions perhaps of high degrees, of the polynomial coefficients. Moreover, these are the first complete set of Euclidean invariants for high degree IP curves appearing in the computer vision literature.

As shown in Fig. 4 and Table 2, invariants are individually less stable than pose parameters. In particular, angle invariants are more sensitive to curve-data perturbations. Nevertheless, these angular invariants are useful for discriminating between shapes as illustrated in Fig. 5. Moreover, as shown in Fig. 3, the better stability of the translation estimation in comparison to the angle estimation allows rotation invariants to be computed out of the range of stability of the angle estimation (0.2 noise std. dev. and 20% missing data). We observed that a few angular invariants have a standard deviation several times larger than the others. It turns out

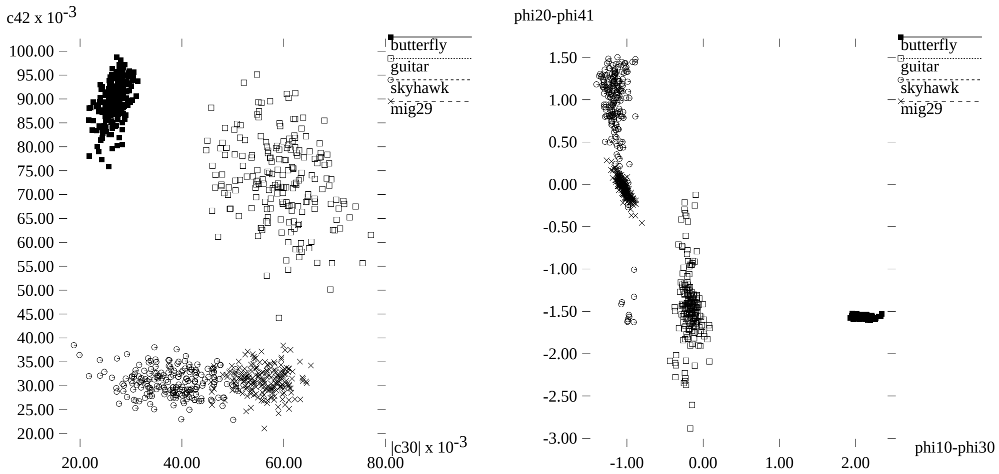



Figure 5: Left, scatter of invariants vector $(|c_{30}|, c_{22})$ for 200 perturbed data sets (colored noise with 0.05 standard deviation) of the 4 IPs of degree 4 in Fig. 1. Right, scatter, in radians, of invariants vector $(3\phi_{10} - \phi_{30}, \phi_{20} - \phi_{31})$.

that, for particular shapes, a few angular invariants become bimodal up to a particular amount of noise such as $\phi_{20} - \phi_{31}$ as shown in Fig. 5 for the sky-hawk.

## 4.2 Invariant Recognition

| | guitar | butterfly | sky-hawk | mig |
|---|---|---|---|---|
| guitar | 100% | 0% | 0% | 0% |
| butterfly | 0% | 100% | 0% | 0% |
| sky-hawk | 0% | 0% | 100% | 0% |
| mig | 0% | 0% | 0% | 100% |
| guitar | 95% | 1.5% | 3.5% | 0% |
| butterfly | 27.5% | 72.5% | 0% | 0% |
| sky-hawk | 2.5% | 0% | 97.5% | 0% |
| mig | 9.5% | 0% | 52% | 38.5% |
| guitar | 100% | 0% | 0% | 0% |
| butterfly | 0% | 100% | 0% | 0.0% |
| sky-hawk | 0.5% | 0% | 96% | 3.5% |
| mig | 4% | 0% | 0% | 96% |

Table 3: Percentage recognition on 3 sets of 200 perturbed shapes for colored noise of standard deviations 0.05 and 0.1, and 10% missing data, respectively.

Fig. 5 shows scatter plots vectors of pairs of invariants for the 4 shapes of degree 4 of Fig. 1. Though the scatter of individual components of invariant vectors are not always well separated, the use of the complete set of invariants appears to yield highly accurate recognition. The recognizer used is Bayesian recognition based on a multivariate colored Gaussian distribution for each object and having a diagonal covariance matrix estimated from 200 noisy perturbed shapes for each object with noise perturbation standard deviations 0.05 in the normal direction. This model is used to do recognition on two other noisy sets having standard deviation 0.05 and 0.1 (the latter is 12.5 pixels and is at the limit of the stability for pose) and on one set with 10% missing data. Results are quite good (see Table 3). For large noise perturbations (0.1 std. dev. colored noise), the sky-hawk becomes difficult to recognize from the other airplane, since details are lost in noise, but is still different from the guitar or the butterfly shapes. Better accuracy would be achieved by using $6^{th}$ degree IP curves [15].

### 4.3 Indexing in a silhouette database

In the previous section, recognition is applied to datasets deformed by synthetic perturbations: colored noise and missing data. To test the proposed recognition algorithm on real data, we used a database of 1100 boundary contours of fish images obtained from a web site [11]. The number of data points in each silhouette in this database varies from 400 to 1600. This database contains not only fishes but more generally sea animals, i.e, the diversity of shapes is large. So as not to use size for easy discrimination between shapes, all shapes are normalized to the same size. To prepare the database, every shape is fit by a $4^{th}$ degree polynomial and then polynomial curves are centered by setting $t_{center}$ at the origin. The last step is to compute the rotation invariants as in the previous section. We first run queries by example to test the rotation invariants. Two examples are shown in Fig. 6 where the rotation invariance is clear.

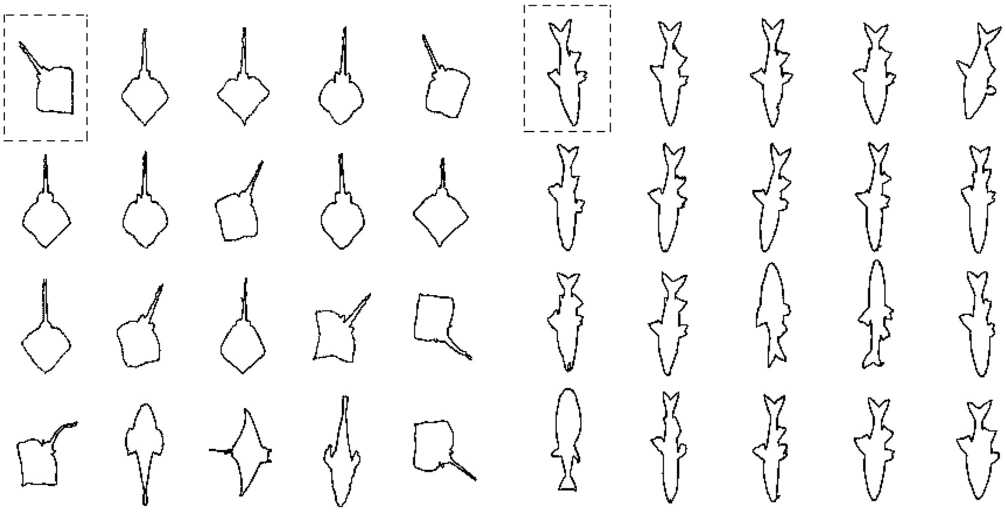


Figure 6: Queries by example invariant to Euclidean Transformations and reflections.

Then, we test the stability to small perturbations such as removing data on a few shapes and running queries with these modified shapes. In Fig. 7, small parts are removed in the query. The capability of the IPs to handle missing data (especially if small patches are removed at many locations throughout a silhouette) and to handle open (non closed) curves is one of the main advantage of this description in comparison to descriptions using arc length parameterization such as Fourier descriptors or B-splines.

With our approach query by sketch is also possible. For every query, a Bayesian recognizer is used since the variability of each invariant can have very different standard deviations. This variability is estimated from a training set. This training set is synthetically generated from the given sketch by adding perturbations: sample functions of a colored noise. Obtained standard deviations are used to weight each invariant during the comparison between shapes in the database, i.e, the Mahalanobis distance is used. (The optimal weighting is to use a full inverse

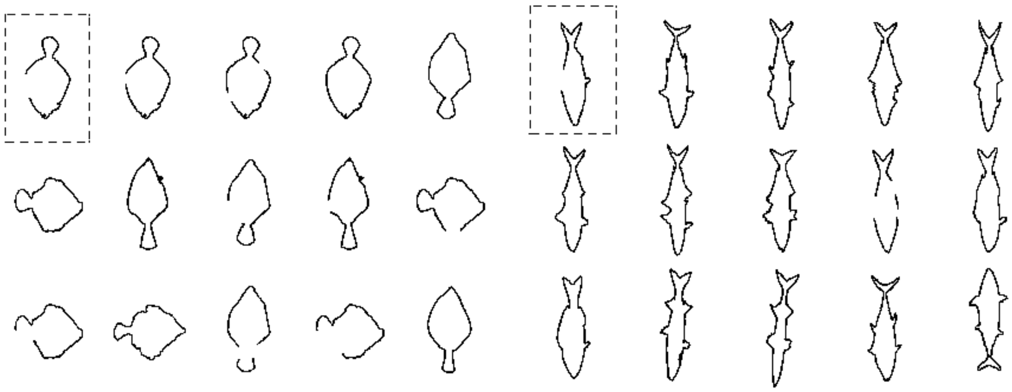


Figure 7: Queries by example where relative small parts as fans are removed.

covariance matrix in the quadratic-form recognizer, rather than the diagonal covariance matrix approximation used in these experiments.) It turns out that depending on the talent of the drawer and on the number of occurrences of and variability in the target shape in the database, the user may want to control the similarity measure used between the query and the searched-for database shapes. To handle this, the standard deviation used in generating training sets is decreased or increased depending on whether more or less similarity is needed. Therefore, in addition to the query, the user has to specify what degree of similarity he/she wants to use: very similar (std. dev. is 0.02), similar (std. dev. is 0.1), weakly similar (std. dev. is 0.2). Fig 8 illustrate the database shapes found for the same query but using three different similarity criteria.

Obviously, a $4^{th}$ degree polynomial is not able to discriminate shapes with only small scale dissimilarities. Better discrimination power can be achieved by using $6^{th}$ degree polynomials, see [15].

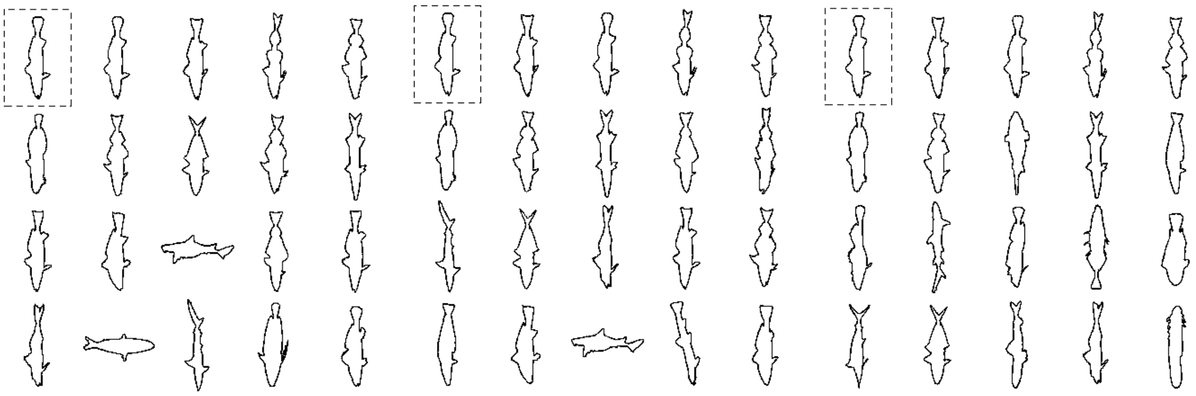


Figure 8: Queries using the same query example but with a recognizer trained on different standard deviations of the colored noise (0.02, 0.1, and 0.2 respectively). The first three closest shapes are always retrieved, but increasing variability can be observed in the other retrieved shapes.

# 5 Conclusions

Though the shape-representing IP's that we use may be of high degree, we have introduced fast accurate pose estimation, and fast accurate pose-independent shape recognition based on geometric invariants. Approximate initial single-computation estimates are computed, and these are iterated 2 or 3 times to achieve the closest local minimum of the performance functionals used. The pose estimation uses all the IP coefficients, but is not optimal because it does not use optimal weightings. The pose-independent recognition uses estimated centering based on all of the IP coefficients followed by rotation invariant recognition based on a complete set of geometric rotation invariants. However, optimal weightings were not used here either. Nevertheless, the effectiveness of the pose-invariant recognition is illustrated by the indexing application in a database of 1,100 silhouettes. Though some of the invariants may not effective discriminators, the complete set is. If put into a Bayesian or Maximum Likelihood framework, we can achieve fully optimal pose estimation and pose-independent shape recognition.

Extensions to 3D based on tensors are undergoing further development [19], as well as is handling local deformations. Of great importance is to extend the pose estimation and transformation-invariant shape recognition to handle two situations. First is that for which *considerable* portions of a silhouette are missing, perhaps due to partial occlusion or where the silhouette is much more complicated. Then, pose estimation and recognition can be based on "invariant patches". These invariant patches are discussed in [10], and the ideas in the present paper should be applicable. The second extension is to handle affine rather than just Euclidean transformations of shapes. The intermediate transformation, scaled Euclidean, should be easy to handle, since an isotropic scaling of the data set by $\lambda$ simply multiplies every monomial of degree $d$ in the polynomial by the factor $\lambda^d$. The full affine transformation is more challenging, and we are studying it. One approach is to convert the Affine Transformation Problem to a Euclidean Transformation Problem through a normalization based on the coefficients of the polynomials fit to the data [1]. The challenge here is to develop a normalization that uses much of the information contained in the polynomial coefficients [18] and which is highly stable. Another subject of interest is to consider small locally affine deformations along a silhouette.

# 6 Appendix: From Complex to Real Representation

What is the transformation relating the coefficients in the real and complex polynomial representations? Computing the coefficients of the complex representation given a real IP appears to be complicated. The transformation from the complex $C$ to the real $A$ vector representation, i.e, the reverse way, is easier to compute. Vector $C$ duplicates information since $c_{kj} = \bar{c}_{jk}$. Therefore, it is in practice more efficient to use the vector representation $B$ with components

$\alpha_{jk} = Re(c_{jk})2^{j+k}$, $\beta_{jk} = Im(c_{jk})2^{j+k}$, and $\gamma_{jj} = c_{jj}2^{2j-1}$, as introduced in Sec. 2.2. Indeed, $B$ is a minimal description of $C$.

Transformation matrix $T$ between $B$ and $A$ ($A = TB$) is block diagonal since the coefficients for a form transform independently of the coefficients for each other form. Thus,

$$T = \begin{bmatrix} T_0 & 0 & 0 & \dots & 0 \\ 0 & T_1 & 0 & \dots & 0 \\ 0 & 0 & \frac{1}{2}T_2 & \dots & 0 \\ \vdots & \vdots & \vdots & . & \vdots \\ 0 & 0 & 0 & \dots & \frac{1}{2^{n-1}}T_n \end{bmatrix} \tag{12}$$

where $T_l$ is the transformation matrix of homogeneous polynomial $H_l$ of degree $l$.

The goal of this section is to find a recursive way to compute $T_l$. Consider the following family of formal real homogeneous polynomials in complex representation:

$$D_l(z) = \frac{1}{2} \sum_{0 \leq j,k \leq l, j+k=l} d_{j-k} \bar{z}^j z^k$$

with $d_j = \bar{d}_{l-j}$. We deduce the following second order recursive formula for $D_l(z)$:

$$\begin{aligned} D_l(x,y) \quad & = Re(d_l \bar{z}^l) + z\bar{z}D_{l-2}(z) \\ & = Re(d_l) \textstyle\sum_{0 \leq 2k \leq l} \begin{pmatrix} l \\ 2k \end{pmatrix} (-1)^k x^{l-2k} y^{2k} \\ & + Im(d_l) \textstyle\sum_{0 \leq 2k+1 \leq l} \begin{pmatrix} l \\ 2k+1 \end{pmatrix} (-1)^k x^{l-(2k+1)} y^{2k+1} \\ & + (x^2 + y^2) D_{l-2}(x,y) \end{aligned}$$

where the second and third lines are the expansion of $Re(d_l \bar{z}^l)$. From the last equation, we deduce a recursive computation for $T_l$:

$$T_0 = N_0^{-1} = [1] \qquad T_1 = N_1^{-1} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$T_l = \begin{bmatrix} 1 & 0 & \\ 0 & \binom{l}{1} & \\ -\binom{l}{2} & 0 & 0_{(l-1)\times(l+1)} \\ 0 & -\binom{l}{3} & \\ \vdots & \vdots & \end{bmatrix} + \begin{bmatrix} 0_{2\times(l-1)} & T_{l-2} \\ 0_{2\times 2} & 0_{(l-1)\times 2} \end{bmatrix} + \begin{bmatrix} 0_{2\times 2} & 0_{2\times(l-1)} \\ 0_{(l-1)\times 2} & T_{l-2} \end{bmatrix}$$

Note, $\binom{l}{k}$ denotes the binomial coefficient $\frac{l!}{(l-k)!k!}$. As an illustration, the first three iterations give:

$$T_2 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 2 & 0 \\ -1 & 0 & 1 \end{bmatrix} \qquad T_3 = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 3 & 0 & 1 \\ -3 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \end{bmatrix}$$

$$T_4 = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 \\ 0 & 4 & 0 & 2 & 0 \\ -6 & 0 & 0 & 0 & 2 \\ 0 & -4 & 0 & 2 & 0 \\ 1 & 0 & -1 & 0 & 1 \end{bmatrix}$$

and one can check that from Sec. 2.2, $N_2 = (\frac{1}{2}T_2)^{-1}$ and $N_3 = (\frac{1}{2^2}T_3)^{-1}$.

These matrices $T_l$ specify $T$ (see (12)) which describes how a 2D polynomial is transformed from its complex $C$ to real $A$ representation. In practice, to obtain the real and imaginary parts of the complex polynomial coefficients that represent a data set, we first fit a real polynomial to the data set to estimate the coefficient vector $A$, and then obtain $B$ by $B = T^{-1}A$.

## Acknowledgments

This work was supported in part by a postdoctoral grant from INRIA, Domaine de Voluceau, Rocquencourt, France.

## References

[1] G. Taubin. Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation. *IEEE Trans. on Pattern Anal. and Machine Intell.*, 13(11):1115–1138, Nov. 1991.

[2] Z.H. Huang and F.S. Cohen. Affine-invariant b-spline moments for curve matching. *Image Processing*, 5(10):1473–1480, October 1996.

[3] F. Solina and R. Bajcsy. Recovery of Parametric Models from Range Images: The Case for Superquadrics with Global Deformations. *IEEE Trans. on Pattern Anal. and Machine Intell.*, 12(2):131–147, Feb. 1990.

[4] S. De Ma. Conics-based stereo, motion estimation and pose determination. *IJCV*, 10(1), 1993.

[5] T.H. Reiss. *Recognizing Planar Object Using Invariant Image Features.* Lecture Notes in Computer Science, Springer-Verlag, 1993.

[6] J.L. Mundy and A. Zisserman, editors. *Geometric Invariance in Computer Vision*. MIT Press, 1992.

[7] I. Weiss. Noise resistant invariants of curves. *IEEE Trans. Pattern Anal. and Machine Intell.*, 15(9):943–948, Sept. 1993.

[8] E. Calabi, P.J. Olver, C. Shakiban, A. Tannenbaum, and S. Haker. Differential and numerically invariant signature curves applied to object recognition. *IJCV*, 26(2):107–135, Feb. 1998.

[9] A.N. Bruckstein, A.M. Netravali. Differential invariants of planar curves and recognizing partially occluded shapes. In *Visual Form: Anal. and Recog.*, pages 89–98, 1991.

[10] Z. Lei, T. Tasdizen, and D.B. Cooper. Pims and invariant parts for shape recognition. In *Proceedings of Sixth ICCV*, pages 827–832, Mumbai, India, 1998. also as LEMS Tech. Report 163, Brown University.

[11] F. Mokhtarian, S. Abbasi, and J. Kittler. Robust and efficient shape indexing through curvature scale space. In *Proceedings of the sixth British Machine Vision Conference, BMVC'96*, pages 53–62, Edinburgh, 1996.

[12] J. Ponce, A. Hoogs, and D.J. Kriegman. On using CAD models to compute the pose of curved 3D objects. *CVGIP*, 55(2):184–197, March 1992.

[13] J.-P. Tarel and D. Cooper. A new complex basis for implicit polynomial curves and its simple exploitation for pose estimation and invariant recognition. In *Proceedings IEEE CVPR*, pages 111–117, Santa Barbara, California, USA, June 1998.

[14] M. Unel and W. Wolovich. Complex representations of algebraic curves. In *Proc. IEE Intl. Conf.Image Process.*, Chicago, Oct. 1998.

[15] T. Tasdizen, J.-P. Tarel, and D.B. Cooper. Improving the stability of algebraic curves for applications. *IEEE Trans. on Image Processing*, 9(3):405–416, March 2000.

[16] Z. Lei, M. M. Blane, and D. B. Cooper. 3L fitting of higher degree implicit polynomials. In *Proc. Third IEEE Workshop on Applics of Comp. Vision*, Sarasota, Florida, USA, Dec. 1996.

[17] M. Schwartz. *Information Transmission, Modulation, And Noise*. McGraw-Hill, 1990. 4th edition.

[18] J.-P. Tarel, W. A. Wolovich, and D. B. Cooper. Covariant conics decomposition of quartics for 2D object recognition and affine alignment. In *Proceedings of Intl. Conf.Image Processing*, pages 818–822, Chicago, Oct. 1998.

[19] J.-P. Tarel, H. Civi, and D. B. Cooper. Pose estimation of free-form 3D objects without point matching using algebraic surface models. In *Proceedings of IEEE Workshop on Model-Based 3D Image Analysis*, pages 13–21, Mumbai, India, 1998.

- Jean-Philippe Tarel graduated from the Ecole Nationale des Ponts et Chaussées, Paris, France in 1991. He received the Ph. D. degree in applied mathematics from Paris IX-Dauphine University in 1996. He was with the Institut National de Recherche en Informatique et Automatique (INRIA), France, from 1991 to 1996. From 1997 to 1998, he was a Research Associate at Brown University, Providence, RI. Now he is a Researcher with the Laboratoire des Ponts et Chaussées (LCPC), Paris, France. His research interests include computer vision, pattern recognition, algebraic geometry and computer graphics.

- David B. Cooper (S '53 - M '64 - SM '90 -LS'99-F'00) received the B.Sc. and Sc.M. degrees in electrical engineering from the Massachusetts Institute of Technology, Cambridge, under the industrial cooperative program in 1957, and the Ph.D. degree in applied mathematics from Columbia University, New York, NY, in June 1966. From 1957 to 1966, he worked first for Sylvania Electric Products, Inc., Mountain View, CA, and then for Raytheon Company, Waltham, MA, on communications and radar systems analysis. S-ince 1966, he has been a Professor of engineering at Brown University, Providence, RI, where he does research in pattern recognition, image understanding, and algebraic geometric stochastic approaches to computer vision. At Brown University, he was Associate Directory of the Laboratory for Engineering Man/Machine Systems (LEMS) from 1982 through 1997, and Academic Head of Electrical Engineering for 1996 to 1998.