A New Complex Basis for Implicit Polynomial Curves and its Simple Exploitation for Pose Estimation and Invariant Recognition

Jean-Philippe Tarel[†⊥]

[⊥]*INRIA*
Domaine de Voluceau, Rocquencourt
B.P. 105, 78153 Le Chesnay, France
jpt@lems.brown.edu

David B. Cooper[†]

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

Abstract

New representations are developed for 2D IP (implicit polynomial) curves of arbitrary degree. These representations permit shape recognition and pose estimation with essentially single, rather than iterative, computation, and extract and use all the information in the polynomial coefficients. This is accomplished 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. These rotations in the two dimensional coefficient subspaces are related in simple ways to each other and to rotation in the x, y data space. By recasting this approach in terms of complex polynomials, i.e, $z = x + iy$ and complex coefficients, further simplification occurs for rotations and some simplification occurs for translation.

1. Introduction

Object recognition based on invariants and object pose estimation are classic central problems in computer vision. In this paper, we deal with object recognition and pose estimation for Euclidean transformations of 2D Implicit polynomial (IP) curves. The relationship of this paper to the existing state of the art is as follows. There is a sizeable literature on alignment and geometric invariants based on moments, B-splines, superquadrics, conics, combinations of straight lines and conics, bitangents, and differential invariants. For lack of space, we simply reference [2] which contains a sampling of some of this literature. For IP, 2D curves and 3D surfaces, the most basic approach to comparison of two shapes is iterative estimation of the transformation of one from the other followed by recognition based on comparison of their IP coefficients or based on comparing the data set for one with the IP representation for the other [3, 7].

A major jump was the introduction of intrinsic coordinate systems for pose estimation and Euclidean and affine algebraic invariants for IP 2D curves and 3D surfaces [7, 2]. These are highly effective but do not use all the information in the IP coefficients. Recently, a geometric interpretation of a 2D IP was provided for quartics which led to pose estimation and invariant recognition under affine and Euclidean transformations [6].

The present paper focuses not on the geometry of the IP but rather on the geometry of the transformation of the IP coefficients and is built on the fact that when the (x, y) data set is rotated, the resulting IP coefficient vector undergoes an orthogonal transformation [7]. The significance of our development is that going to an appropriate linear basis for IP coefficient space, we can compute a complete set of rotational invariants and compute Euclidean pose estimation based on all the information in the IP coefficients and on simple functions of relatively stable linear combinations of IP coefficients.

2. Implicit Polynomial Model

2.1. Definition

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

$$f_n(x,y) = \sum_{0\le j+k\le n} a_{jk}x^j y^k = \underbrace{a_{00}}_{H_0} + \underbrace{a_{10}x + a_{01}y}_{H_1} + \ldots + \underbrace{a_{n0}x^n + a_{n-1\,1}x^{n-1}y + \ldots + a_{0n}y^n}_{H_n} = 0 \quad (1)$$

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.

Implicit polynomial f_n is often represented by coefficient vector A having components (a_{jk}), $0 \leq j,k, j+k \leq n$ (number of coefficients is $\frac{1}{2}(n+1)(n+2)$): `

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

where

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

Superscript T denotes matrix transpose.

In general, the vector representation is convenient for IP fitting since our fitting methods are set within a linear framework, while the tensor representation [5] provides a useful framework for pose estimation for any dimension.

2.2. Conics and Cubics under rotation

We first consider conics and cubics under rotation in order to exhibit properties we want to exploit.

A cubic is defined by 9 coefficients:

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

When a cubic is rotated through angle θ, the 9 coefficients (a_{ij}), $0 \leq i+j \leq 3$ are transformed as a messy function of θ. 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 defines the transformed coordinate system. The original cubic coefficients are written as a 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 relation between the two vectors $A' = LA$, where L is a function of the rotation angle only. This 9×9 matrix can be decomposed in blocks in the following way:

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

where the diagonal 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 $L_1 = R(\theta)$ and

$$L_2 = \begin{bmatrix} c^2 & -cs & s^2 \\ 2cs & c^2 - s^2 & -2cs \\ s^2 & cs & c^2 \end{bmatrix}$$

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

The elements of these blocks are non-linear functions of $c = \cos\theta$ and $s = \sin\theta$. It is well known that the trace $a_{20} + a_{02}$ of the quadratic form is not modified by a rotation, and it is convenient to introduce the angle 2θ to easily solve pose estimation of conics. Equivalent to the trace property is that the sum of the first and third lines of L_2 is independent of the angle θ, which is easily verified. To take advantage of this relation, we define a new parameterization, α_{20}, β_{20}, α_{21}, of the coefficients a_{20}, a_{11}, a_{02} of the polynomial by applying the following matrix N_2:

$$\begin{bmatrix} \alpha_{20} \\ \beta_{20} \\ \alpha_{21} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{20} \\ a_{11} \\ a_{02} \end{bmatrix}$$

These new parameters α_{2i} and β_{20} are linear functions of the original polynomial coefficients. With this new representation, the matrix L_2 is mapped into a matrix where 2θ appears:

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

The reason for this α, β notation is that α_{jk} and β_{jk} are the real and imaginary parts of the complex coefficient c_{jk} introduced in the next section.

For L_3, it turns out that a similar simplification is possible with the transformation N_3:

$$\begin{bmatrix} \alpha_{30} \\ \beta_{30} \\ \alpha_{31} \\ \beta_{31} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 3 & 0 & 1 & 0 \\ 0 & 1 & 0 & 3 \end{bmatrix} \begin{bmatrix} a_{30} \\ a_{21} \\ a_{12} \\ a_{03} \end{bmatrix}$$

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 specified by the square matrices (N_j), $1 \leq j \leq 3$ where L is mapped into diagonal 2×2 and 1×1 block form:

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

The coefficient vector of the cubic in the new basis is C, and C' after rotation $R(\theta)$. It is clear that in this new basis, the coefficient space is decomposed into one or two dimensional subspaces invariant to rotations and the vector

$C = [\alpha_{00}, \alpha_{10}, \beta_{10}, \alpha_{20}, \alpha_{30}, \beta_{30}, \alpha_{31}, \beta_{31}]$ is decomposed into 2D vectors $[\alpha_{jk}, \beta_{jk}]$ which rotate with angles θ, 2θ, or 3θ with respect to the applied rotation. This leads directly to a simple and robust way to compute the relative orientation between cubics. Moreover, it is very easy to compute a complete set of invariants under rotation for a cubic (in the sense that all other invariants are determined by these):

- 2 linear invariants: coefficients α_{21} and α_{00},
- 4 quadratic invariants: radiuses $\alpha_{10}^2+\beta_{10}^2$, $\alpha_{20}^2+\beta_{20}^2$, $\alpha_{30}^2+\beta_{30}^2$, and $\alpha_{31}^2+\beta_{31}^2$,
- and 3 relative angles: the angle between $\frac{1}{3}[\alpha_{30}, \beta_{30}]$ and $[\alpha_{31}, \beta_{31}]$, between $[\alpha_{10}, \beta_{10}]$ and $[\alpha_{31}, \beta_{31}]$, and between $\frac{1}{2}[\alpha_{20}, \beta_{20}]$ and $[\alpha_{31}, \beta_{31}]$, for instance.

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 Implicit Polynomials

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 invariant object recognition. To simplify notation, we focus on the form $H_j(x,y) = \sum_{0\le k\le j} a_k x^{j-k} y^k$. The main idea is to rewrite $H_j(x,y)$ as a real polynomial of complex variables $z = x + iy$ and $\bar{z} = x - iy$:

$$H_j(x,y) = H_j(z) = \sum_{0\le k\le j} \frac{a_k}{2^j i^k} (z+\bar{z})^{j-k} (z-\bar{z})^k$$

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

$$H_j(z) = \sum_{0\le k\le j} \frac{b_k}{2} z^{j-k} \bar{z}^k$$

Notice that coefficients (b_k), $0 \le k \le j$ are linear complex combinations of the (a_k), $0 \le k \le j$, and that $\bar{b}_{j-k} = b_k$ since the polynomial is real. Thus depending on odd or even degree, the previous expression can be rewritten as:

$$\begin{array}{rcl} H_{2l}(z) & = & Re(\sum_{0\le k<l} b_k z^{2l-k} \bar{z}^k) + \frac{b_l}{2} z^l \bar{z}^l \\ H_{2l+1}(z) & = & Re(\sum_{0\le k\le l} b_k z^{2l+1-k} \bar{z}^k) \end{array} \quad (4)$$

where $Re(.)$ and $Im(.)$ are the real and imaginary parts of the expression within parenthesis. Not to have to discuss odd and even cases, we introduce coefficient c_k and rewrite the previous equations as:

$$H_j(z) = Re(\sum_{0\le k\le [\frac{j}{2}]} \bar{c_k} z^{j-k} \bar{z}^k) \quad (5)$$

where $[u]$ denotes the greatest integer not exceeding u. We call the vector $C = (c_{jk})$ the *complex vector representation* of a algebraic curve which is defined by a real polynomial in z and $\bar{z}$:

$$f_n(z) = Re(\sum_{0\le j\le n, 0\le k\le [\frac{j}{2}]} \bar{c_{jk}} z^{j-k} \bar{z}^k) = Re(\bar{Z}^t C) = 0 \quad (6)$$

where $Z^T = [1, z, z^2, z\bar{z}, z^3, z^2\bar{z}, z^4, \ldots, z^n, \ldots, z^{n-[n/2]}\bar{z}^{[n/2]}]$ is the vector of complex monomials. Thus, we have $f_n(z) = Re(\bar{Z}^T C) = Y^T A$. We denoted by c_{jk} the conjugate of the $k^{th}+1$ coefficient of the j^{th} homogeneous polynomial H_j in z and $\bar{z}$.

For example, the complex representation of a conic is $f_2(z) = Re(\bar{c}_{00} + \bar{c}_{10} z + \bar{c}_{20} z^2 + \bar{c}_{21} z\bar{z})$ where c_{00} and c_{21} are real numbers. Thus, we rewrite it as $f_2(z) = c_{00} + Re(\bar{c}_{10} z) + Re(\bar{c}_{20} z^2) + c_{21}|z|^2$. Remembering notation of the previous section, it is easy to show that $\alpha_{00} = c_{00}$, $\alpha_{10} = Re(c_{10})$, $\beta_{10} = Im(c_{10})$, $\alpha_{20} = Re(c_{20})$, $\beta_{20} = Im(c_{20})$, and $\alpha_{21} = c_{21}$. The complex representation of a cubic is $f_3(z) = c_{00} + Re(\bar{c}_{10} z) + Re(\bar{c}_{20} z^2) + c_{21}|z|^2 + Re(\bar{c}_{30} z^3) + Re(\bar{c}_{31} z)|z|^2$, 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. We see that if the IP shape is rotated through angle θ (see (3)), z transforms as $z' = e^{i\theta} z$, so that $z = e^{-i\theta} z'$, and by substituting in (5):

$$H_j(z) = Re(\sum_{0\le k\le [\frac{j}{2}]} e^{-i(j-2k)\theta} \bar{c}_k z'^{j-k} \bar{z'}^k) = H'_j(z')$$

Hence, the coefficients of the transformed polynomial are

$$c'_k = e^{i(j-2k)\theta} c_k \quad (7)$$

Moreover, there is a recursive and thus fast way to compute the (N_j) matrix permitting to transform a given polynomial coefficient vector A to the new basis C for any degrees.

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 polynomials C and C' (see (7)). It turns out that complex representation is not only useful for rotation but also has nice properties under translation allowing us to do pose estimation under Euclidean transformation in a very fast way and by using all the information in the polynomial coefficients.

3.1. Implicit Polynomial Fitting

Since object pose estimation and recognition are realized in terms of coefficients of shape-representing IP's, the process begins by fitting an IP to a data set representing the 2D curve of interest. For this purpose we use 3L fitting [1], which is a least squares linear fitting of an explicit polynomial to the Distance Transform of the data set. The IP curve is the zero set of this explicit polynomial. 3L fitting [1, 5]

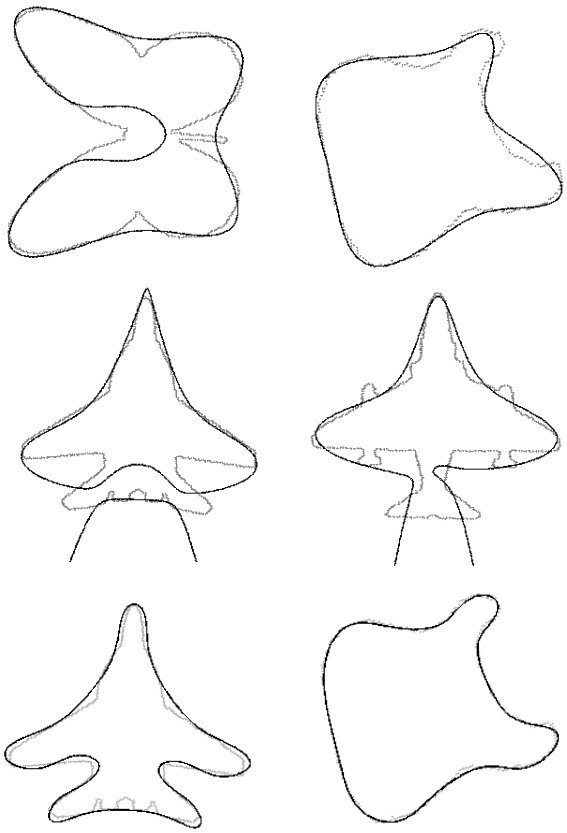

Figure 1. 3L fits of 4^{th} degree polynomials to a butterfly, a guitar body, a mig 29 and a sky-hawk airplane. In the bottom are 6^{th} degree polynomials fits.

is of lower computational cost and has better polynomial estimated-coefficient repeatability than all previously existing IP fitting methods. Fig. 1 shows measured curve data and the fit obtained with the 3L fitting. This fitting is numerically stable and repeatable, with respect to Euclidean transformations of the data set, and robust to noise and a moderate percentage of missing data as shown in Fig. 2. 4^{th} degree fits allows us to capture the global shape, and higher degree polynomials provide more accurate fits as shown in Fig. 1 with 6^{th} degree polynomials. We are working to maintain good fitted coefficient stability even for high degrees.

3.2. Translation

It is well known that none degenerate conics have a center. Given two conics, the two centers are very useful to estimate the relative pose since each conic can be centered before computing the relative orientation. The goal of this section is to compute a center with similar properties for any degrees in the complex representation.

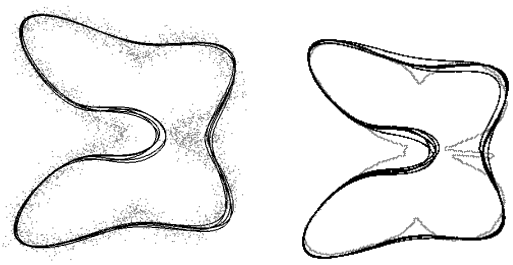

Figure 2. Left, 5 fits are superimposed with associated noisy data sets. The original data set is perturbed with a Gaussian noise along the normal with a standard deviation of 0.1 for a shape size of 3 (or, shape size is 375 pixels and the noise is with a 12.5 pixels standard deviation). Right, 10% of the curve is chopped at 5 random starting points.

From (6), if z is transformed as $z' = e^{i\theta}z + t$ (i.e rotated with an angle θ and translated by t), we have:

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

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

$$H'_n(z') = Re\Big(\sum_{0\le k\le[\frac{n}{2}]} \bar{c}_{nk} e^{-i(n-2k)\theta} z'^{n-k} \bar{z'}^{k}\Big)$$

Consequently, complex coefficients $(c'_{nk})_{0\le k\le[\frac{n}{2}]}$ of the transformed leading form H'_n are unaffected by a pure translation. Continuing expansions, we obtain the transformed next highest degree form $H'_{n-1}(z')$ having the coefficients $(c'_{n-1\,k})$, $0 \le k \le [\frac{n-1}{2}]$:

$$\begin{aligned} H'_{n-1}(z') = Re\Big(&\textstyle\sum_{0\le k\le[\frac{n-1}{2}]} \bar{c}_{n-1\,k} e^{-i(n-1-2k)\theta} z'^{n-1-k} \bar{z'}^{k} \\ &- t \textstyle\sum_{0\le k\le[\frac{n}{2}]} \bar{c}_{nk}(n-k) e^{-i(n-1-2k)\theta} z'^{n-1-k} \bar{z'}^{k} \\ &- \bar{t} \textstyle\sum_{1\le k\le[\frac{n}{2}]} \bar{c}_{nk} k e^{-i(n+1-2k)\theta} z'^{n-k} \bar{z'}^{k-1}\Big) \end{aligned}$$

Therefore, these coefficients are transformed in a linear way with respect to the translation component t:

$$c'_{n-1\,k} = e^{i(n-1-2k)\theta}(c_{n-1\,k} - (n-k)c_{nk}\bar{t} - (k+1)c_{n\,k+1}t) \quad (8)$$

Notice, that if n is even, a careful derivation yields a special equation for (8) when $k = [\frac{n-1}{2}]$: $c'_{n-1\,k} = e^{i(n-1-2k)\theta}(c_{n-1\,k} - (n-k)c_{nk}\bar{t} - 2(k+1)c_{n\,k+1}t)$. This is due to the symetric term in z and $\bar{z}$ in (4) for even degrees.

The first interesting property of (8) 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_{center} which minimizes the least squares problem:

$$\sum_k |c_{n-1\,k} - (n-k)c_{nk}\bar{t}_{center} - (k+1)c_{n\,k+1}t_{center}|^2 \quad (9)$$

does not depend on the rotation applied on the polynomial. Note, this t_{center} is determined solely by curve having coefficients C and is a center for this curve. The t_{center} of a high degree polynomial has the same property as the conic center. Consequently, it can be used in the same way: each polynomial C and C' are centered by computing t_{center} and t'_{center} before computing the relative orientation using all the transformed coefficients. This center is not different than the Euclidean center of a polynomial derived with a completely different approach in [7].

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 (9), with the same invariance to rotation. The property used here is that $c'_{(n-1)k}$ depends only on $c_{(n-1)k}$, c_{nk}, and $c_{n(k+1)}$, i.e, the complex representation uncorrelates the rotation and the translation. But the robust way to define a unique Euclidean center for IP is the point t_{center}. Indeed, this center makes use of the redundancy of information in all the coefficients of H_n and H_{n-1} to obtain robust estimate.

3.3. Rotation

In the previous section, only coefficients of leading form H_n and next highest degree form H_{n-1} are used from all the coefficients in C and C'. Experimentally, it turns out that it is the coefficients of H_n and H_{n-1} which are the most robust to noise and small perturbations. Nevertheless, it is important to take advantage of all the information available in the polynomial to obtain the most accurate pose estimation possible. At first, we assume that the two polynomials are centered by using the Euclidean center defined in the previous section.

For a cubic, under rotation $R(\theta)$, C transforms to $C' = [c_{00}, e^{i\theta}c_{10}, e^{2i\theta}c_{20}, c_{21}, e^{3i\theta}c_{30}, e^{i\theta}c_{31}]^T$. In this equation, as seen in section 2.2, complex coefficients c_{10} and c_{31} are rotated by angle θ, c_{20} by angle 2θ, and c_{30} by angle 3θ. These coefficients all have information about the shape orientation, but with different periods. If c_{10} and c_{31} give directly the relative orientation between C and C', c_{20} gives it up to π, and c_{30} up to $\frac{2\pi}{3}$.

In the general case, from (7), C' as a function of C under a rotation θ is given by $c'_{jk} = c_{jk}e^{i(j-2k)\theta}$ where $0 \leq j \leq n$ and $0 \leq k \leq [\frac{j-1}{2}]$. Given C and C', we simply used least squares to estimate θ:

$$\min_\theta \sum_{0\leq j\leq n, 0\leq k\leq [\frac{j-1}{2}]} |c'_{jk} - c_{jk}e^{i(j-2k)\theta}|^2 \quad (10)$$

which leads to maximization of $\sum |c_{jk}||c'_{jk}| cos((j-2k)\theta + arg(c_{jk}) - arg(c'_{jk}))$. Assuming that the noise and the error is small, the previous equation is approximated by its second order Taylor expansion. Then the solution is derived, and this maximization is solved as:

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

where weights w_{jk} are $(j-2k)^2|c_{jk}||c'_{jk}|$.

As just pointed out in the cubic example, l_{jk} is an unknown integer when $j - 2k \neq 1$, and $\frac{2\pi l_{jk}}{j-2k}$ is the period. Integer l_{jk} is between 0 and $j - 2k - 1$. Since the computational cost is very small, the best estimated angle can be obtained by computing the estimate for all possible integers and chose the one which minimizes the w_{jk} weighted standard deviation of $\frac{arg(c'_{jk}) - arg(c_{jk}) + 2\pi l_{jk}}{j-2k}$. (Note, the accuracy of this least squares solution can be improved by dividing these differences by their respective standard deviations.) An alternative is to search for the roots of the derivative of (10) with respect to θ, which is a polynomial in $e^{i\theta}$.

3.4. Euclidean transformations

At this point, if only shape rotation has occurred, we have indicated how to make optimal (or very good) use of all coefficients in order to estimate the rotation. The computation of the translation does not use all the information available. To achieve an optimal algorithm with respect both rotation and translation, refinement can be applied, by assuming that t is small in magnitude.

If z is translated as $z' = z + t$ and $|t|$ is small, $f_n(z' - t)$ should be well approximated by the constant and linear term in the Taylor series expansion of $f_n(z' - t)$ with respect to t, and then the linear equation in t for H_{n-1}, (8), applies to all lower degree forms H_j, $j \leq n-1$. This property allows us to compute the refined translation by using all coefficients of the polynomial. After this step, the orientation can be refined by computing the residual relative angle as previously. The refinement of the translation and of the orientation iterates a few times since the convergence is in practice very fast.

In summary, for pose estimation:

- First the polynomial is centered by computing the Euclidean center from the coefficients of H_n and H_{n-1} with (9) as discussed in section 3.2.
- Then the rotation is computed by using information in all the coefficients of f_n using (11) and the discussion in section 3.3,
- The translation and the orientation are iteratively refined one or two times with (9) extended to all forms and (11) applied on all the coefficients of f_n.

The advantage of this algorithm is its robustness and simplicity since most of the computations are linear.

	noise 0.1	n. 0.2	occl. 10%	o. 20%
angle	1.7%	59.3%	1.1%	42.7%
trans.	1.4%	9.2%	1.4%	3.3%

Table 1. Standard deviation in percentage of the average mean of the angle and the norm of one translation component with various perturbations. Added 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.

The proposed pose estimation is numerically stable, repeatable, and robust to noise and a moderate percentage of missing data as illustrated in Table 1. The shape is the butterfly, having size 3 (or 375 pixels) shown in Fig. 1, and the degree of the fit is 4. The pose estimation error due to occlusion increases nicely in the range from 0 to 15% (19 pixels), as shown in Fig. 3. Similar results are obtained in the range $[0, 0.17]$ for the standard deviation of the noise. A noise standard deviation 0.17 is 6% of the size of shape of the butterfly (or 21 pixels), which represent a relatively large perturbation (see Fig. 2). For higher values of noise, we have the well know threshold effect [4] in estimation problems. It arises in our problem because large additive noise masks the shape bumps useful for rotation estimation.

4. Recognition Using Invariants

In the previous section, complex representation is used to solve the problem of pose estimation for IPs of any degree. But the complex representation is also useful for deriving rotation invariants. The use of these invariants to compare two IPs is a very fast way to do shape recognition in 2D images.

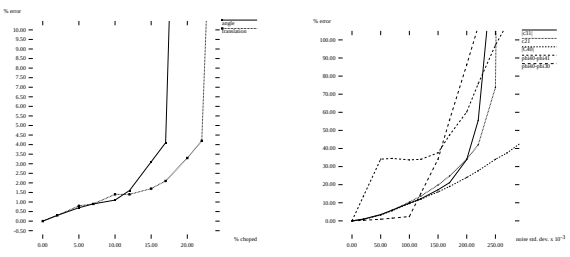

Figure 3. Left, variation of the standard deviation of the angle and the x component of the translation for an increasing percentage of occlusion at 200 random starting points. Right, variation of the standard deviation on invariants $|c_{31}|$, c_{21}, $|c_{40}|$, $|\phi_{40} - \phi_{41}|$, and $|\phi_{40} - \phi_{30}|$ (with 200 realizations) for an increasing amount of Gaussian noise at each point. Value are percentage of the average value of the corresponding pose component or invariant. Rotation is 1 radian, translation is 1.

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

	noise 0.1	n. 0.2	occl. 10%	o. 20%
$\|c_{31}\|$	9.8%	33%	6.7%	31.6%
c_{21}	10.4%	34%	8.1%	18.9%
$\|c_{40}\|$	9.6%	24%	2.1%	6.1%

Table 2. Standard deviations as a percentage of the means of a few invariants in response to various data perturbations. Gaussian noise has standard deviation 0.1 and 0.2. Occlusions are 10% and 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 geometric invariants [6] is $\frac{1}{2}(n+1)(n+2)-1$. We directly have $[\frac{n}{2}]+1$ linear invariants which are $c_{j[\frac{j}{2}]}$ when j is odd. From (7), we deduce that all other $|c_{jk}|$ are invariants under rotations. This gives us $([\frac{n}{2}]+1)[\frac{n}{2}]$ quadratic invariants. Invariants $|c_{jk}|$ are geometric distances, but there are angles which are also Euclidean invariants for an IP. Indeed, relative angles between the $\phi_{jk} = \frac{arg(c_{jk})}{j-2k}$ are preserved under rotations. This yields a complete set of rotation invariants for an IP of degree n as for the cubic case in section 2.2. We want to emphasis the fact that most of the obtained invariants are

linear or quadratic even for high degree polynomials. This leads to invariants less sensitive to noise than are those derived from other approaches such as symbolic methods.

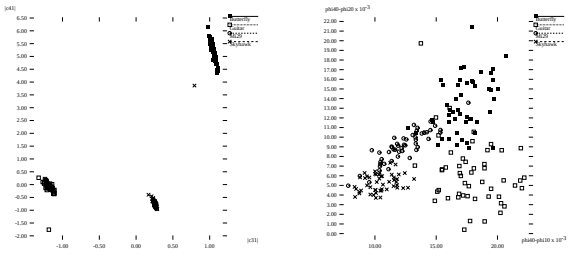

Figure 4. Left, scatter of invariants vector $(|c_{31}|, c_{21})$ for 50 perturbations (noise with 0.1 standard deviation) of each shape in Fig. 1. Right, scatter of invariants vector $(|c_{40}|, \phi_{40} - \phi_{41})$.

As shown in Table 2, invariants are less robust than pose parameters since there are computed independently. But as shown in Fig. 3, the better robustness of the translation estimation in comparison to the angle estimation allows rotation invariants to be computed out of the range of robustness of the angle estimation. For particular shapes, a few angular invariants become bimodal up to a particular amount of noise, such as $|\phi_{40} - \phi_{30}|$ for the butterfly as shown in Fig. 3.

	mig	butterfly	skyhawk	guitar
mig	100%	0%	0%	0%
butterfly	15%	91.5%	0%	7%
skyhawk	55%	0%	45%	0%
guitar	8%	0%	0%	92%

Table 3. Percentage recognition on a set of 200 perturbed shapes for noise of standard deviations 0.2.

Fig. 4 shows scatter plots vectors of pairs of invariants for the 4 shapes of Fig. 1. Though the scatter of 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 Gaussian distribution for each object and having a diagonal covariance matrix estimated from 200 noisy shapes for each object with standard deviations 0.1 in the normal direction. This model is used to do recognition on another noisy set with standard deviation 0.2 (25 pixels, i.e, at the limit of the robustness for pose). Results are quite good (see Table 3). For large noise perturbations, the sky-hawk becomes difficult to recognize from the other airplane, since details are lost in noise.

5. Conclusions

Though the shape-representing IP's that we use may be of high degree, we have introduced pose estimation, and recognition based on geometric invariants, which are realized by linear operations followed by reasonably stable nonlinear operations on the IP coefficients. The goal is to take advantage of the intrinsic robustness and accuracy of linear and near linear estimation. If put into a Bayesian or Maximum Likelihood framework, we can achieve pose estimation or object recognition which uses all the information contained in the IP coefficients. In this paper we presented approximately optimal pose estimation and hybrid recognition. The approximately optimal pose estimation used all the information available in the IP coefficients, but not with the optimal weightings. The hybrid recognition used translation estimation based on some of the IP coefficients followed by rotation invariant recognition based on a complete set of geometric rotation invariants. The translation estimation is quite accurate, and the object recognition based on a complete set of rotation invariants produces a single computation recognizer that appears to be highly accurate because, though some of the invariants are not effective discriminators, the complete set is! This is the first complete set of geometric invariants for an IP that we are aware of! This approach needs to be explored further, and extensions to 3D are under study.

References

[1] Z. Lei, M. M. Blane, and D. B. Cooper. 3L fitting of higher degree implicit polynomials. In *Proceedings of Third IEEE Workshop on Applications of Computer Vision*, Sarasota, Florida, USA, December 1996.

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

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

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

[5] 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.

[6] J.-P. Tarel, W. A. Wolovich, and D. B. Cooper. Aligned conic decompositions of quartics for 2D object recognition and pose estimation. Lems tech. report, Division of Engineering, Brown University, 1997.

[7] 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(11):1115–1138, November 1991.