IMPLICIT ALGEBRAIC CURVES AND SURFACES FOR SHAPE MODELING AND RECOGNITION

by

Hakan Çivi

M.S. in Computer Engineering, Boğaziçi University, 1992

B.S. in Computer Engineering, Boğaziçi University, 1989

B.S. in Mathematics, Boğaziçi University, 1989

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for
the degree of Doctor of Philosophy
in Industrial Engineering

Boğaziçi University

October 1997

Abstract

Implicit algebraic 2D curves and 3D surfaces are among the most powerful shape representations. With this approach, objects in 2D images are described by their *silhouettes* (em surfaces in 3D) and then represented by 2D implicit polynomial curves; objects in 3D data are represented by the implicit polynomial surfaces. Implicit polynomial degrees of $2nd$ and greater are used, typically up to 18th for 2D curves and 12th for 3D surfaces. In the thesis four contributions are presented whose common theme is using implicit polynomials for 2D and 3D shape modeling and object recognition purposes. Through the proposed concepts and algorithms, we argue that implicit polynomials provide a fast, and low computational cost model for low-error indexing into shape databases in 2D and 3D.

The first contribution finds algebraic invariants of implicit polynomials under projective, affine, and Euclidean transformations by using and extending the symbolic method of classical invariant theory and shows how the invariants can be used for object recognition in invariant spaces. The second contribution is a novel model-based explicit algebraic expression for 2D-2D coordinate transformation estimation for Euclidean, similarity, and non-uniform scale transformations. The proposed explicit algebraic pose estimation expressions involve only the implicit polynomial coefficients for an object in two positions, and are drawn from an observation of how the implicit polynomial object model transforms under coordinate transformations. The third contribution is a 3D version of the 3L algorithm and an investigation of the specific problems in applying the technique in 3D as opposed to 2D. Finally, the fourth contribution explores the problem of getting *affine invariant fits* based on the 3L fitting as the 3L algorithm is inherently Euclidean invariant, but not affine invariant.

Özet

Örtük cebirsel 2B (2 boyutlu) eğriler ve 3B (3 boyutlu) yüzeyler en güçlü şekil temsil yöntemleri arasındadır. Bu yaklaşımla 2 boyutlu imgelerdeki nesneler silüetleriyle tanımlanıp 2B örtük polinom eğrileriyle, 3B veri kümelerindeki nesneler de 3B örtük polinom yüzeylerle temsil edilirler. 2 ya da daha yüksek dereceli örtük polinomlar kullanılır; tipik olarak 2B eğrilerde 18inci dereceye, 3B yüzeylerde de 12inci dereceye kadar çıkılılır. Bu tezde, ortak teması örtük polinomları 2 ve 3 boyutta şekil temsil etme ve tanımada kullanmak olan dört katkı sunulmaktadır. Önerilen kavramlar ve algoritmalar yoluyla örtük polinomların 2B ve 3B şekil veri tabanları için düşük hata oranlı indeksler olarak kullanılabilecek hızlı ve düşük hesaplama maliyetli bir model oldukları gösterilemeye çalışılacaktır.

İlk katkı yoluyla klasik değişmezler kuramının sembolik yöntemi izdüşüm, ilgin (affine) ve Öklid dönüşümleri altında örtük polinomların cebirsel değişmezlerini elde etmek için kullanmakta ve genişletmektedir. Bunun yanı sıra, elde edilen cebirsel değişmezlerin değişmezler uzayında nesne tanımada nasıl kullanılabileceği açıklanmaktadır. İkinci katkı, 2 boyuttan 2 boyuta Öklid, benzerlik ve birbiçimsiz ölçekleme (non-uniform scaling) dönüşümlerinin tahminini sağlayacak yeni bir açık cebirsel ifadedir. Sadece nesnenin iki durumdaki örtük polinom katsayılarına dayanan önerilen ifade, nesnenin örtük polinom modelinin koordinat dönüşümleri altında nasıl değiştiğine ilişkin bir gözlemden yola çıkılarak elde edilmiştir. Üçüncü katkı 3L (3 Düzey) eğri uydurma algoritmasını 3 boyuta genellemekte ve bu tekniğin 3 boyuttaki kullanımınına özgü problemleri incelemektedir. Son olarak dördüncü katkıyla, 3L algoritması kullanılarak ilgin değişmez eğri uydurması yapılması problemi ele alınmaktadır. 3L algoritması tanımı gereği Euclidean değişmez olduğu halde affine değişmez değildir.

Acknowledgments

My greatest debt is to Nurdan Çivi whose support, love and patience have made this thesis possible. I also owe much to my family for their unconditional support throughout my life.

I sincerely thank my thesis advisors Prof. Aytul Erçil at Boğaziçi University and Prof. David Cooper at Brown University, not only for their steadfast guidance, encouragement and many insights that built my understanding of the scientific endeavor and computer vision, but also for their invaluable friendships I am most fond of. I would like to thank Prof. David Cooper separately for his meticulous review and correction of this manuscript.

I am especially grateful to Dr. Colin Christopher of Plymouth University for introducing me to algebraic geometry which later developed into the theoretical backbone of this thesis and coauthoring articles. I appreciate Prof. Bill Wolovich at Brown University for his thorough readings of the articles forming parts of this thesis and his insightful comments.

I would like to thank my thesis committee members: Prof. Işıl Bozma, Prof. Muhittin Gökmen and Prof. Süleyman Özekici for accompanying me at my last step through this journey.

I would also like to thank all my professors at Boğaziçi University, particularly Kuban Altınel and Yaman Barlas, for their guidance and friendship over the past years.

I sincerely thank my collageaues Michael Blane, Zhibin Lei, and Jean-Philippe Tarel in the Brown University LEMS Laboratory for their friendship and the opportunity to have coauthored articles. I am also grateful to Michael Escuti, Thomas Sebastian, Tolga Taşdizen, Hüseyin Tek and Mustafa Ünel in the LEMS Laboratory, and Ayşe Aksoğan, Necati Aras, Ayşegül Arsoy, Nijaz N. Bajgoric, Aslı Erdem, Aslı Özgökmen, Umur Özkul, Özlem Verün, and Tülin Yazgaç at Boğaziçi University for their companionship, cooperation and many fruitful interactions.

I am grateful to Kevin Leary and Alex Zatsman of Analog Devices, and Muzaffer Hıraoğlu and Selçuk Taral of TUBITAK Marmara Research Center for their friendship, support and being understanding throughout my professional engineering life at both institutions during my graduate studies.

This research was partially supported by TUBITAK NATO-A2 Fellowship.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As the volume and circulation of images increase at an enormous rate due to new multi–media tools for manipulating them and the internet for distributing them, the need for automated handling of image content is stimulating new technologies. These technologies such as pictorial and video databases, 3D reconstruction for virtual reality are defining problems, some of which have solutions which require efficient shape representations. For instance, in 2D using shape information for indexing into pictorial databases, or in 3D shape recovering for indexing video content and automatic querying, are still mostly unsolved shape-based tasks. Together with its well–established status in relatively older fields such as robot vision and industrial inspection, shape understanding has become a pivotal problem.

Some researchers believe that implicit algebraic 2D curves and 3D surfaces are among the most powerful shape representations presently known. With this approach, objects in 2D images are described by their *silhouettes* and then represented by 2D implicit polynomial curves; objects in 3D data are represented by the implicit polynomial surfaces. Implicit polynomial degrees of $2nd$ and greater are used, typically up to $18th$ for 2D curves and $12th$ for 3D surfaces. Implicit curves and surfaces of degree 2 (*conics*, i.e., ellipses, hyperbolas, pair of lines) and 4 (*quartics*) have been the most extensively studied ones. Among the main strengths of this representation are its interpolation property for handling missing data, smoothing property against noise and perturbations, Bayesian recognizers, closed-form pose estimators, and algebraic invariants of polynomial curves or surfaces under various groups of coordinate changes [1, 2, 3, 4, 5, 6]. These advantages and its mathematical attractiveness, perhaps deeply rooted in its simplicity and availability of the strong arsenal of the concepts

and algorithms of algebraic geometry, make implicit polynomial representation a viable approach to solving many shape related computer vision problems including the ones defined by the new technologies.

This potential has led to research efforts not only in computer vision but also in a host of other areas ranging from computer graphics to industrial inspection. Although these works secured a respectable standing for implicit polynomials in the computer vision community, widespread adoption did not follow. This trend reveals itself best in the eventual waning of interest in the mid 90's in exploring implicit polynomials for computer vision purposes.

The primary source of discouragement was the lack of a satisfactory solution to a seemingly easy problem: fitting an implicit polynomials to a given 2D or 3D data set. If algebraic invariants, pose estimators and other mathematical features of implicit polynomials are to be brought to bear, an indispensable requirement is to have a fast, robust, repeatable fitting algorithm. For existing fitting algorithms the solution was only a locally optimal one, and it was usually sensitive to noise or missing data, especially when the data set to be fit was intrinsically complex, i.e., too complex to be represented by a single implicit polynomial. Fortunately, most of these problems have been circumvented by a recent breakthrough in implicit polynomial fitting, namely the 3-level (3L) fitting algorithm [7, 8].

The initial goal of this thesis research was to apply the polynomial technology to the industrial inspection problem. Although we have done some work in this area [9], the richness of theoretical and practical problems resulting from the availability of an efficient fitting algorithm shifted our focus to the four relatively independent contributions of this thesis having as common theme the use of implicit polynomials for the purposes of 2D and 3D shape modeling and object recognition. Through the proposed concepts and algorithms, we will argue that implicit polynomials provide a fast and low computational cost model for low-error indexing into shape databases in 2D and 3D and facilitate the solution of some of the problems stemming from industrial inspection.

The first topic we tackle is finding algebraic invariants of implicit polynomials and showing how they can be used for object recognition in invariant spaces. To this end, the symbolic method of classical invariant theory is extended and applied for the calculation of algebraic invariants of implicit curves and surfaces under projective,

affine, and Euclidean transformations.

The second contribution is a novel model-based explicit algebraic expression for 2D-2D coordinate transformation estimation for Euclidean, similarity, and non-uniform scale transformations. The proposed explicit algebraic pose estimation expressions involve only the implicit polynomial coefficients for an object in two positions, and are drawn from an observation of how the implicit polynomial object model transforms under coordinate transformations. The explicit solution expression provides a fast, efficient pose estimation technique for many model based vision applications.

The third contribution is a 3D version of the 3L algorithm and an investigation of the specific problems in applying the technique in 3D as opposed to 2D. We report our results and outline the application areas the developed representation methodology ideally suits to.

Finally, the fourth contribution explores the possibility of computing *affine invariant implicit polynomial fits* as a variant of the 3L fitting, as the 3L algorithm is inherently Euclidean invariant, but not affine. That is, with the 3L, a Euclidean transformation of a data set leads to a transformation of the fitted polynomial coefficients by the same transformation and this is not the case for affine transformations.

As a final note, in this manuscript we have used portions of some of our previous writings-a number of articles, resulting from our thesis research, currently under review for different journals and conferences [7, 2, 4, 9, 10].

The organization of the thesis is as follows: Chapter 2 provides an overview of the literature pertinent to implicit polynomials and introduces some of the implicit polynomial concepts. Chapters 3 and 4 explain our work on computing algebraic invariants and our results on pose estimation, respectively. Chapter 5 discusses our work on extending the 3L algorithm to 3D, gives a brief glimpse of the characteristics of the resulting implicit surface fits through examples, and lays out implications for 3D object representation. Chapter 6 describes the preliminary results of our investigation on making the 3L fitting affine invariant. The final chapter draws conclusions and points at further research directions.

Shape Modeling for Industrial Automation

As briefly mentioned, shape representation is pivotal for solving many machine vision problems. In this thesis, we are interested in shape models for the purpose of manufac-

turing automation and industrial inspection. Also of importance is machine recognition of shapes and shape-based indexing into databases for 2D curve and 3D surface shapes.

Industrial inspection requires efficient solutions to three shape related problems: *object recognition, pose estimation*, and *automated inspection.* In manufacturing environments, raw shape information is acquired by using Coordinate Measuring Machines (CMM), calibrated camera rigs, laser range finders etc. or through a combination of them. Usually through a preliminary processing of the raw data a set of 2D points forming the object contour (object surface in 3D) is obtained (see Chapters 2, 5, 6). This data set is then represented by a single or multiple implicit polynomials. In such environments typically, different objects are carried along a conveyor belt or by robots and at different stages of the process classifying objects with respect to their identification is needed. Object recognition techniques developed in implicit polynomial paradigm form a good candidate for addressing this problem. Chapter 3 discusses this problem and explains our approach. Inspecting the geometry of a part or a finished product involves determination of whether a given item differs from its manufacturing specifications, usually specified by certain dimensions and tolerances. Implicit polynomials have nice properties for inspection such as determining whether a given feature point is inside or outside the object boundary. Also with the 3L fitting algorithm for small shape variations distance of a point from the boundary can be determined acurately. Chapters 2, 5 explain some of these features of implicit polynomials. Prior to inspection the object is usually in an arbitrary position and orientation. The object should be placed into a standard position with respect to which manufacturing specifications are given. This can be accomplished by estimating the pose of the object and this information can be used for moving the object into its standard position. In Chapter 4 we discuss our approach to this problem.

Chapter 2

Implicit Polynomials

2.1 Literature Overview

Implicit polynomials are one of the leading *shape representations* in computer vision [11, 6, 12, 13]. Although the underlying theory, i.e. *algebraic geometry* [14, 15] has long been around, major inroads into applications of implicit polynomials had to wait until the 1980's. Since then, independent research in a number of fields, including computer graphics, geometric modeling, and computer vision, has accumulated valuable insights into various properties of implicit polynomials important for solving practical problems. Research in geometric modeling brought to bear parametric to implicit conversion techniques which, in essence, is an elimination theory problem, and relative strenghts of parametric and implicit representations were understood [16, 17, 18, 19]. Computer graphics works proved that implicit polynomials can play an important role in visualization [1, 20, 21, 22, 23]. However, probably the most comprehensive contributions came from computer vision where the idea created an elaborate object representation and recognition paradigm enjoying several strengths of implicit polynomials, such as their interpolation property for handling missing data, smoothing property against noise and perturbations, Bayesian recognizers, and, most importantly, their algebraic invariants [7, 2, 24, 25, 4, 26, 5, 6].

An essential requirement for practicality of implicit polynomial related techniques is to have robust and consistent implicit polynomial fits to data sets. The problem has been formulated through different minimization criteria and constraints. A variety of iterative and non-iterative solution techniques, perturbation methods and stopping

rules have been proposed. A classical minimization criterion is $\sum f^2(x,y)$ with which in the unconstrained case the solution is the well-known least-sqaures technique [27, 28]. Cooper and Yalabık [29] gave a geometric interpretation of the criterion and remarked that it does not really correspond to a minimization of *the geometric distance* of a point to a curve. When *the geometric distance* from a point to the zero set of an implicit polynomial is minimized, an iterative process is needed because, except for line fitting in 2D and plane fitting in 3D, there is no explicit expression for this distance. This formulation requires nonlinear optimization. Several geometric distance approximations have been used, such as the first order approximation [6], which speed up computation considerably, but iterative nonlinear optimization is still required.

Keren focused on obtaining bounded fitting, particularly for the $4th$ degree implicit polynomials, by incorporating the symmetric positive defineteness of the leading form coefficient matrix as a constraint in a nonlinear optimization formulation [3, 30]. Another bounded fitting algorithm is the ellipse fitting of [31] through a generalized eigenvector solution, yet the result does not generalize to higher degree implicit representations. Linear programming fitting of Lei [32, 33] allows users to interactively choose significant points such that the fitted curve lies within a user specified ϵ using the linearized distance approximation; other linearized constraints can be imposed. Finally, the 3L fitting algorithm (see Section 2.4.1) provides a linear solution to the fitting problem and seems to overcome many drawbacks of the previous algorithms.

In computer vision, a typical application of implicit polynomials has been object recognition for large object databases. For this purpose, first a set of sensed data points of an object to be recognized, usually the object silhouette (contour), or the object surface in 3D, is fit by an implicit polynomial. Then a vector of algebraic invariants is computed from the polynomial coefficients to form an *invariant space representation* of the object at hand. This vector is compared with vectors stored in the database, to resolve the matching problem [2, 34, 35, 36, 37, 38, 39]. Alternatively, if the object is modeled by more than one implicit polynomials, joint invariants of pairs of polynomials are used along with algebraic invariants of individual polynomials [40, 41, 42]. A more detailed account of different applications of the concept of invariance in computer vision is presented in Chapter 3 where we also discuss our work on the invariant theory and its applications in computer vision. In [5, 42, 43] a Bayesian decision theoretic framework and its implications for coefficient and invariant space representations are discussed as a cure to coefficient variability arising from noise and occlusion of data

sets. The approach is to employ an asymptotic Bayesian approximation for defining probability distributions for polynomial coefficients (or invariants) and to use the resulting Mahalonobis distance as an effective recognizer.

Another popular application of implicit polynomials has been determining the orientation of an object from its implicit polynomial representation and using this information for alignment or recognition [4, 44, 45]. Along these lines Ponce and Kriegman proposed a 3D pose estimation technique [46], and Taubin formulated the canonical frame of reference [6, 36]. Taubin's work is based on the fact that algebraic curve and surface representation naturally gives rise to an intrinsic frame of reference consisting of an intrinsic coordinate center and intrinsic coordinate orientation. The idea is generalized to an affine canonical frame of reference with moments of data sets and for implicit polynomial curves [37]. A detailed discussion of the pose estimation problem in computer vision and the use of polynomials is given in Chapter 4 along with our contribution.

More recently, the conic-line decomposition of Wolovich and Unel [47, 48], provides an important simple geometric interpretation of the underlying geometry for a given implicit polynomial. This representation leads to pose estimation algorithm and allows the formulation of new invariants based on certain covariant characteristics such as conic centers because they map under an affine transformation. Another latest development is Polynomial Interpolated Measure (PIM) in [39]. Given two data sets and the corresponding implicit polynomial representations, PIM defines a similarity measure between the data sets as it satisfies all the properties of a distance metric and can be expressed as a Mahalanobis distance in the polynomial coefficients.

2.2 Geometry of the 2D Plane

In order to investigate the algebraic properties of a geometric shape (object) in a plane an arbitrary coordinate system $\{O, X, Y\}$ is chosen. Given another coordinate system $\{O', X', Y'\}$ the points in the two coordinate frames are related by a coordinate transformation. Note that because of the relativity of the shape motion and the motion (or change) of the underlying coordinate system, a shape position change can also be accounted for as a coordinate system change. Below we define some of the linear coordinate transformations from special to more general and list some of the geometric

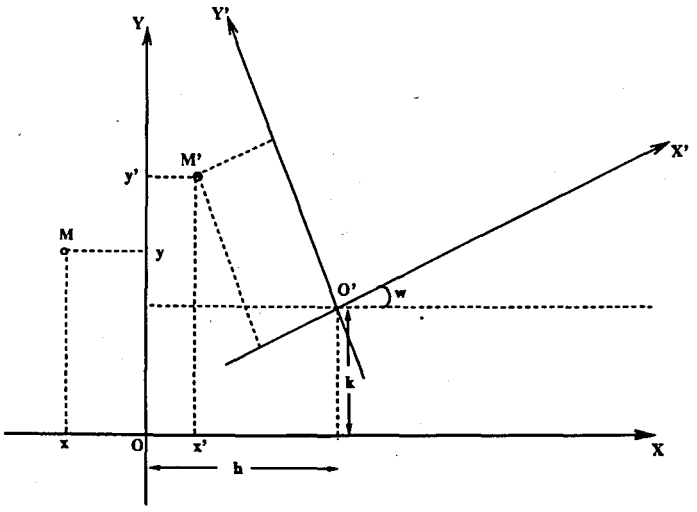


Figure 2.1: A Euclidean coordinate transformation.

properties preserved under these transformations.

A *Euclidean transformation* is a rigid motion of the coordinate frame, i.e., a rotation and a translation of it. A Euclidean transformation (see Figure 2.1) maps a point M with the coordinates (x, y) to a point M' with the coordinates (x', y') according to the following equations:

$$\begin{cases} x' = cos\omega\ x - sin\omega\ y + h \\ y' = sin\omega\ x + cos\omega\ y + k \end{cases} \tag{2.1}$$

Euclidean transformations form an algebraic group with 3 transformation parameters: an angle ω, and two translation parameters h and k along the x axis and y axis, respectively. Under Euclidean transformations, the distance of two points, and the angle between two lines are preserved.

A slight generalization of the Euclidean transformation is the *similarity transformation*. Similarity transformations include a uniform scale factor λ and form a 4 parameter algebraic group. The transformation rule for the similarity transformation is defined as follows:

$$\begin{cases} x' = \lambda cos\omega\ x - \lambda sin\omega\ y + h \\ y' = \lambda sin\omega\ x + \lambda cos\omega\ y + k \end{cases} \tag{2.2}$$

As a consequence of the scaling factor, a similarity transformation does not preserve the distance between two points; angles and the ratio of the lengths of the projections of a line on the two coordinate axes are preserved.

A *non-uniform scaling* involves a further scaling factor which causes different scalings along the two axes and it is a 5 parameter algebraic group. The non-uniform scaling is defined as

$$\begin{cases} x' = \lambda_1 cos\omega\ x - \lambda_1 sin\omega\ y + h \\ y' = \lambda_2 sin\omega\ x + \lambda_2 cos\omega\ y + k \end{cases} \tag{2.3}$$

A non-uniform scaling still preserves angles, but the ratio of the projections lengths are not preserved.

An *affine transformation* has 6 parameters and is defined by the equations

$$\begin{cases} x' = ax + by + h \\ y' = cx + dy + k \end{cases} \tag{2.4}$$

where it is assumed that the following matrix is nonsingular

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

An affine transformation does not preserve angles, therefore the coordinate axes are not necessarily perpendicular. Some of the properties invariant under an affine transformation are parallelism and the ratio of the length of a pair of line segments.

The most general linear transformation is the *projective transformation.* The projective plane is constructed from the Cartesian plane by introducing homogeneous coordinates X, Y, Z for the plane points

$$x = X/Z, \quad y = Y/Z, \quad Z \neq 0$$

where x, y are the Cartesian coordinates of a point. This construction is augmented by adding *points at infinity (improper points)* for which at least one of X, Y is non-zero and $Z = 0$. The plane so obtained is called the projective plane. In the projective plane, proportional points (X, Y, Z) and (kX, kY, kZ) correspond to the same point, and points which are not proportional are different points. A projective transformation of the projective plane is defined by the below 3 homogeneous equations in the 3 variables X, Y, Z

$$\begin{cases} X' = aX + bY + cZ \\ Y' = dX + eY + fZ \\ Z' = gX + hY + kZ \end{cases} \tag{2.5}$$

Because of the proportional point equivalance, projective transformations have 8 parameters instead of 9. For a detailed account of the projective geometry of the plane we refer the reader to [14, 49].

The coordinate transformations defined above have an important relevance to computer vision in modeling image plane correspondences. Different camera geometry assumptions [50, 51, 52] lead to modeling the image plane correspondence with different transformations. The linear transformations outlined in this section are commonly used for modeling image plane correspondence. Figures 2.3 and 3.2 give examples of Euclidean, similarity, and affine transformations of objects in 2D images. We will further discuss this relation in Chapter 6 as it pertains to our work.

2.3 Implicit Polynomial Model

In 2D an implicit curve of degree n is given by the following equation

$$f(x,y) = \sum_{0\le i,j;i+j\le n} a_{ij}x^i y^j = \underbrace{a_{00}}_{H_0} + \underbrace{a_{10}x + a_{01}y}_{H_1(x,y)} + \underbrace{a_{20}x^2 + a_{11}xy + a_{02}y^2}_{H_2(x,y)} + \ldots$$

$$+ \underbrace{a_{n0}x^n + a_{n-1,1}x^{n-1}y + \ldots + a_{0n}y^n}_{H_n(x,y)} = \sum_{r=0}^{n} H_r(x,y) = 0, \tag{2.6}$$

Here $H_r(x,y)$ is a *homogeneous binary (i.e., two variables) polynomial (form)* of degree r in x and y. Notice that in Equation 2.6 the graded lexicographic ordering on monomials induced by $x_3 \prec x_2 \prec x_1$ is applied. To facilitate implicit polynomial fitting, $f(x,y)$ is written in vector form as (T denotes matrix transpose)

$$f(x,y) = \mathbf{Y}^T\mathbf{A}, \tag{2.7}$$

$$\mathbf{A} = [\, a_{00} \quad a_{10} \quad a_{01} \quad a_{20} \quad a_{11} \quad a_{02} \quad \ldots \quad a_{0n} \,]^T \tag{2.8}$$

$$\mathbf{Y} = [\, 1 \quad x \quad y \quad x^2 \quad xy \quad y^2 \quad x^3 \quad \ldots \quad x^n \quad \ldots \quad xy^{n-1} \quad y^n \,]^T \tag{2.9}$$

Note that such curves of degree 2 are the commonly used *conics*–ellipses, circles, pairs of straight lines,hyperbolas, etc. *Quartics* (implicit polynomials of degree 4), too, have drawn much attention and their properties have been extensively studied. The above formulation (2.6) extends naturally to 3D. An *implicit polynomial surface* is a function in 3 variables x, y, z, and a *3D implicit curve* is represented by the intersection of two implicit surfaces.

In computer vision, objects in 2D images are described by their *silhouettes* and, in this thesis, then represented by implicit polynomial curves. More formally, an implicit polynomial surface is said to represent a shape (object) $\Gamma_0 = \{(x_i, y_i) | i = 1, ..., K\}$ if every point of the shape Γ_0 is in the zero set of the implicit polynomial

$$\mathcal{Z}(f) = \{(x, y) | f(x, y) = 0\}$$

If Γ_0 is a (usually noisy) set of sensed data points along a curve of finite length, and $K > n$ then $\mathcal{Z}(f)$, the zero set of an implicit polynomial fitted to the data will usually be close to the points of Γ_0 but cannot contain all of them. It is customary to use fitting procedures to obtain implicit polynomial representations for objects.

2.4 Implicit Polynomial Fitting

The implicit polynomial 2D curve fitting problem can be set up as follows (the 3D surface fitting problem is similar). Given a data set $\Gamma_0 = \{p_i = (x_i, y_i) | i = 1, .., K\}$, find the *nth* degree implicit polynomial $f(x, y)$ that minimizes the average squared distance from the data points to the zero set $\mathcal{Z}(f)$ of the polynomial [30, 8]. An iterative process is needed to solve for *the geometric distance* from a point to the zero set of an implicit polynomial because there is no explicit expression for this distance. A commonly used first order distance approximation is [6]:

$$d(p_i, \mathcal{Z}(f)) \approx \frac{| f(p_i) |}{\| \nabla f(p_i) \|} \tag{2.10}$$

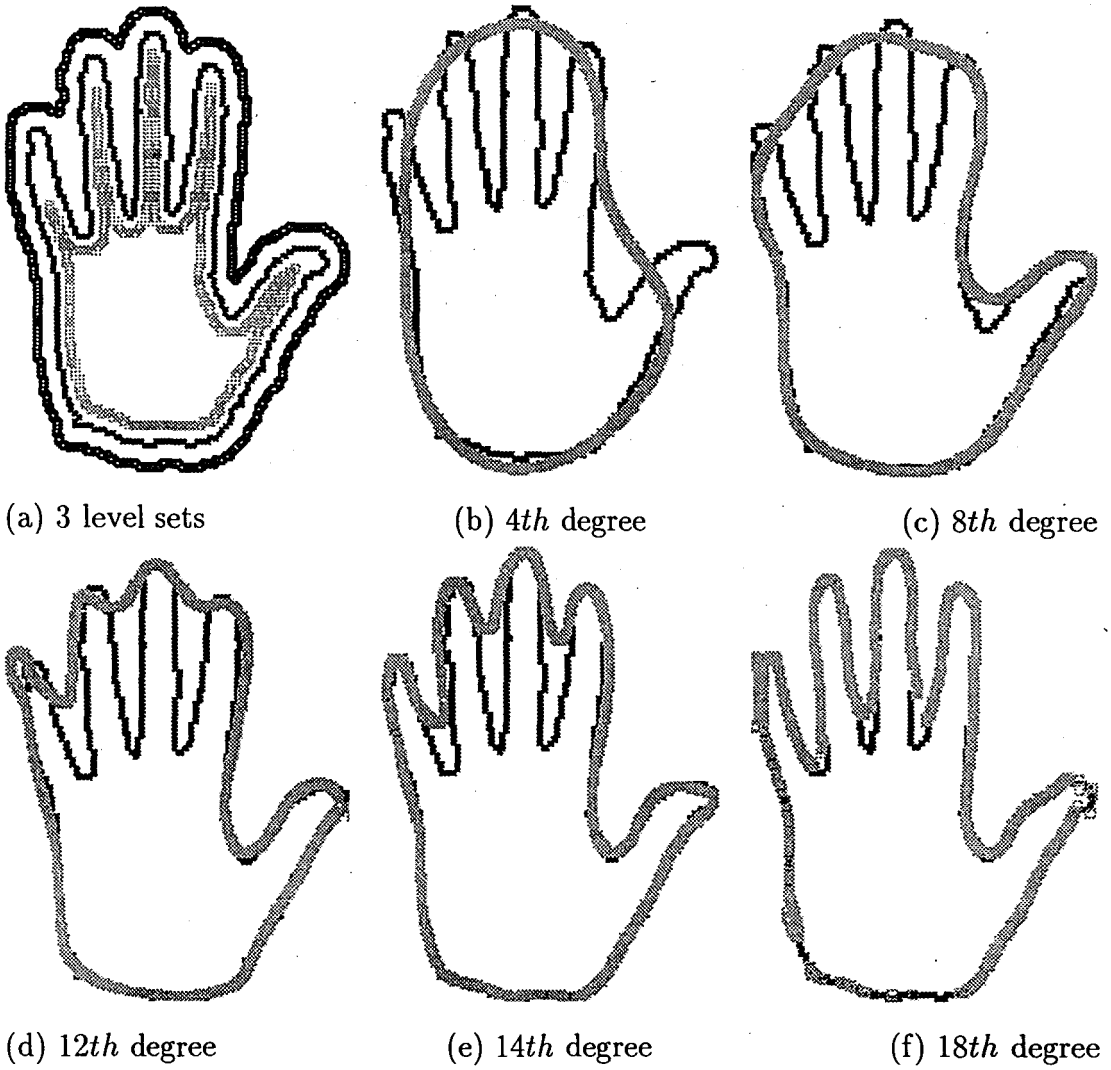
and so the average squared distance is:

$$\overline{d}^2 \approx \frac{1}{N} \sum_{i=1}^{N} d^2(p_i, \mathcal{Z}(f)) = \frac{1}{N} \sum_{i=1}^{N} \frac{| f(p_i) |^2}{\| \nabla f(p_i) \|^2} \tag{2.11}$$

This formulation gives rise to a nonlinear optimization problem to be solved.

2.4.1 3L Fitting

An important drawback with implicit polynomial fitting formulations had been their inability to better constrain the fitting in the vicinity of the data set. The polynomial $f(x, y)$ is an explicit function at all values of x, y, $-\infty < x, y < \infty$, and usually fitting formulations take into account only Γ_0. It is possible to get fast, stable, repeatable implicit polynomial fits by fitting the explicit polynomial $f(x, y)$ to a portion of the



(a) 3 level sets (b) $4th$ degree (c) $8th$ degree

(d) $12th$ degree (e) $14th$ degree (f) $18th$ degree

Figure 2.2: Level sets and different degree implicit polynomial fits for a hand shape.

distance transform, $d(x, y)$, of Γ_0. $d(x, y)$ is the function which, at (x, y), takes on the value of the *signed* distance from (x, y) to Γ_0. That is, $d(x, y)$ is the distance from point (x, y) to the closest point in Γ_0, and takes positive values on one side of the data set Γ_0 and negative values on the other side. 3L fitting, besides the original data set Γ_0, uses a pair of synthetically generated data sets Γ_{+c} and Γ_{-c}, where Γ_{+c} consists of points at a distance c to one side of Γ_0 and Γ_{-c} consists of points of distance c to the other side of Γ_0. Note, Γ_{+c} and Γ_{-c} are the level sets of $d(x, y)$ at levels $+c$ and $-c$, respectively. $d(x, y)$ can be generated by a distance transform computation algorithm from Γ_0. For each data point in Γ_0, the Euclidean distance transform determines a point in Γ_{+c} and another one in Γ_{-c} which are at a perpendicular distance c to each side of the original curve Γ_0 [53, 54]. Figure 2.2(a) (courtesy of Z. Lei) shows the 3 level sets for a hand shape.

Let $\Gamma_0 \cup \Gamma_{+c} \cup \Gamma_{-c} = \{(x_i\ y_i)^T : 1 \leq i \leq 3K\}$ and

$$M = [\mathbf{Y}_1\ \ \mathbf{Y}_2\ \ \ldots\ \ \mathbf{Y}_{3K}]^T$$

where $\mathbf{Y}_i$ is $\mathbf{Y}$ (in 2.9) evaluated at $p_i = (x_i, y_i)$. Also define $\mathbf{d}$ as a vector whose *ith* component is $d(x_i, y_i)$, the distance of the point p_i to Γ_0. As previously explained, the level sets we use for $d(x_i, y_i)$ are only -c, 0, and +c. Then estimating the vector of polynomial coefficients $\mathbf{A}$ (in Equation 2.7) is minimization of $\sum_{i=1}^{3K}(d(x_i, y_i) - \mathbf{Y}_i^T\mathbf{A})^2$, or $\|\mathbf{MA} - \mathbf{d}\|^2$. The least squares solution to this problem is:

$$\mathbf{A} = (\mathbf{M}^T\mathbf{M})^{-1}\mathbf{M}^T\mathbf{d} \tag{2.12}$$

Figure 2.2 gives different degree 3L fittings to a hand shape.

The purpose of introducing the two level sets as additional constraints is three-fold. First, it makes the fitting more stable and consistent with regard to transformations of data sets, and more robust to noisy or missing data. It accomplishes this by fitting a polynomial to more data than just Γ_0; it fits $f(x, y)$ to a ribbon of data rather than to a curve of data. The level sets bring in the additional benefit of forcing singularities away from the vicinity of the data set, as singularities occur at local extrema or saddle-points, and the use of Γ_{-c} and Γ_{+c} discourage the occurrence of singularities within the synthetic data ribbon. Second, the fitted polynomial $f(x, y)$ is an approximation to the distance transform $d(x, y)$. This then has the nice property that given a new data point $(\hat{x}, \hat{y})$, $|\ f(\hat{x}, \hat{y})\ |$ is an approximation to the Euclidean distance from $(\hat{x}, \hat{y})$ to Γ_0. Third, since the implicit curve representation has been turned into the problem of studying the properties of the *explicit* polynomial $f(x, y)$, the full arsenal of linear vector space theory and algorithms becomes applicable, and implicit polynomial curves can be studied and manipulated in completely new ways.

The pose estimator and the algebraic invariants discussed in the following sections particularly relies on consistency and robustness of fitting under transformations, noise, and occlusion. Figure 2.3 illustrates the effect of traditional least square 1L implicit polynomial fittings and the 3L implicit polynomial fittings for a duck shape under various transformations and occlusion. From the irregularities exhibited in the zero set of the least square 1L fittings, it can be seen that it is very sensitive to the transformations and missing data. The actual distance between coefficients of implicit polynomial models is very large, and hence it is not appropriate for the application of pose estimation or algebraic invariants.

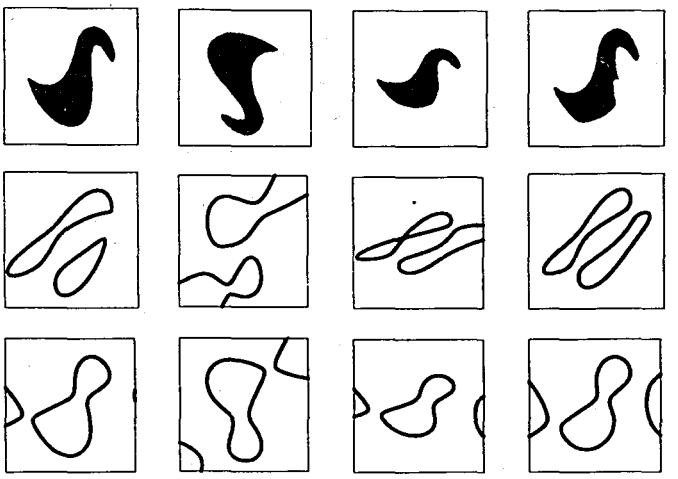


Figure 2.3: First row: object shape, with rotated, scaled and perturbed versions. Second row: zero sets of the least squares IP fittings. Third row: zero sets of the 3L IP fittings.

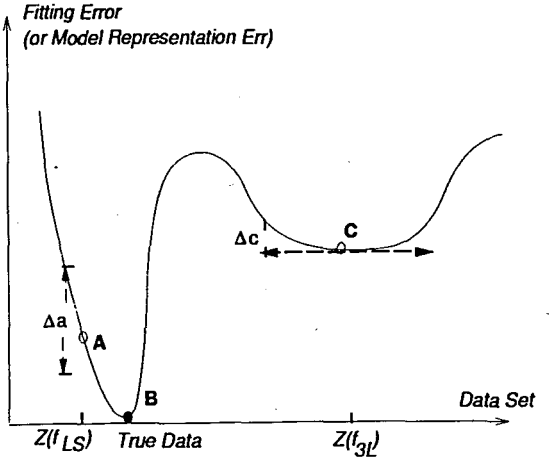


Figure 2.4: Comparison of least squares and the 3L fitting methods in obtaining the best implicit polynomial model for a given data set.

Conceptually, as shown in Figure 2.4, the 3L fitting does not try to solve for the global optimum, but instead, it always returns a solution within a somewhat stable region. In Figure 2.4, the horizontal axis represents the *Data Set* and vertical axis represents the *implicit polynomial Model Fitting Error*. Assume the true data set is at point B and the true implicit polynomial model for it has representation error 0. Nonlinear optimization and least square 1L methods try to search for this optimal solution and eventually stop at some point A with zero set $\mathcal{Z}(f_{LS})$, which could be close or far away from *True Data*. Because no stability or robustness mechanism is employed, A usually lies in a region very sensitive to the perturbation in the data (from noise, missing data or various transformations). A small amount of change in *True Data* can cause a big change $\triangle a$ in the implicit polynomial representation. The 3L fitting returns a solution c that is always within a stable region. Although its zero set $\mathcal{Z}(f_{3L})$ could be somewhat further away than those of nonlinear methods, it is very robust to changes in the *True Data*, i.e., $\triangle c$ is very small regardless of small changes in the data set. Hence, the implicit polynomial model obtained by the 3L fitting contains useful object shape information and can be used to accurately estimate the underlying coordinate system transformation.

Chapter 3

The Classical Theory of Invariants and Object Recognition Using Algebraic Curve and Surfaces

3.1 Concept of Invariance and Computer Vision

The idea of invariance is an inherent element in almost all object recognition works. One can trace at least two different usages of the concept: *photometric invariance* and *geometric invariance.* Of the photometric properties whose invariance is sought are certain intensities such as shading, reflectance, color, etc. These properties arise from a complex interaction of surface characteristics, illumination conditions and view changes which is difficult to control. The geometric invariance paradigm views geometry as the basis for computer vision. With this approach the recognition problem is formulated as extracting ultimate invariant descriptors, stemming from the geometry of the object e.g., edges, which are independent of certain kind of viewpoint changes but not others. The viewpoint changes are expressible in terms of the action of a transformation group such as Euclidean, affine or projective motions. Recently the idea has found a wide application in several works on object representation and recognition [2, 34, 35, 36, 37, 38]. Among approaches having this flavor are those exploiting local differential features such as curvature, those combining local and global information through semi-differential invariants, and those based on global geometry of an object such as implicit polynomial or parametric models [55, 56, 6, 57].

Algebraic invariants of polynomial curves or surfaces under various groups of coordinate changes [25, 36]. An *algebraic invariant* of a polynomial curve or surface in this case is just a function of polynomial coefficients. It is the generation of these functions which concerns us in this part.

In mathematics, the search for *invariants*—quantities which remain the same under a given group of mappings—is fundamental, and its roots go far back into history. Perhaps the most far reaching of these was the Erlangen Program of F.Klein in the 19th Century, where different geometries are classified by their invariants. According to this view, Euclidean geometry, for example, can be considered as the study of motions of the plane for which the distance between any two points is preserved. In computer vision, the main question is to choose the right class of invariants for indexing shapes to be recognized. This choice tightly depends on the camera model and calibration assumptions, as well as the types of the shapes of interest [52].

Another product of the 19th century search for invariants was the classical theory of invariants due to Cayley, Sylvester, Young and many other leading mathematicians of the day including, of course, Hilbert. As aluded to above, the idea was to seek polynomial functions of the coefficients of a curve or surface which are invariant under various transformations. The two main problems that confronted them were how to calculate such invariant polynomials and how to know when you have found them all. A beautiful and complete answer to the first question was found by means of the symbolic method which will be detailed later in this paper. The second question was resolved by Hilbert in two outstanding papers in 1890 and 1893 [58]. The first of these shows that there is a finite set of invariants which generate all the others. The second demonstrates how one can constructively determine when this set has been obtained. Ironically, these papers which heralded a new era for mathematics contributed to the downfall of invariant theory. Having solved the major questions, only tedious and highly complex task of classifying the invariants remained. However, the mood of the times changed and constructive mathematics received very little attention. To this day very few real systems of equations have been exhaustively studied.

The subsequent history of invariant theory is quite chequered. Already in 1971 Dieudonné could write: "Invariant theory has already been pronounced dead several times, and like the phoenix it has been again and again rising from its ashes". In the past decade or so, a new interest has again been taken in the classical theory of

invariants by combinatorics. In particular, the work of Kung and Rota sought to put the classical symbolic method onto more acceptable mathematical foundations. We shall describe this method in the next section.

Until now, the method of computing algebraic invariants for computer vision has been ad hoc. Attention has been restricted to arbitrary classes of invariants, or they have been computed by exhaustive brute force searches [38, 25]. In particular, the findings of the classical theory, especially the symbolic method, have often been ignored. The reason behind this has been a partially justifiable scepticism about the method's applicability.

The aim of this chapter is to demonstrate that a direct approach to the calculation of algebraic invariants via the symbolic method is available. We claim that using the symbolic method we are able to compute complete system of invariants up to a given degree for projective, affine and Euclidean groups of motions. The main advantages of the method are its generality and completeness, i.e., given a polynomial (2-D or 3-D) of any degree or pairs of such the method can compute the full system of invariants up to a given degree. Compared with other methods, there is no elimination needed to ensure that the polynomials produced are invariant, as the invariants are produced directly. The disadvantage at the moment is that the same invariant may be generated several times. However, use of a Gröbner basis package can easily eliminate redundancies. This computational step is necessary in the other methods as well if we are to avoid algebraically dependent invariants.

We will also present a list of some relative affine invariants generated by the proposed algorithm along with the corresponding symbolic representations, and illustrate how they can be used in object recognition by building an example shape database where the objects are represented by implicit polynomials and the indexing into the database is done through the computed invariants.

3.2 The Symbolic Method

The symbolic method provides an elegant procedure for expressing the invariants of a given homogeneous polynomial, or form. An n-th degree form ϕ in 3 variables, for instance, is $\phi(x, y, z) = \sum_{0 \leq i,j,k; i+j+k=n} a_{ijk} x^i y^j z^k$. There is no restriction in considering only homogeneous polynomials as there is a 1-1 correspondence between homogeneous

polynomials in m variables and non-homogeneous (general) polynomials in m–1 variables, via the transformation $f(x_1, x_2, \ldots, x_{m-1}) = x_m^n \phi(x_1, x_2, \ldots, x_m)$, where f is a general polynomial and ϕ is the corresponding homogeneous one. Furthermore, we shall see later that invariants of f can be obtained from joint invariants of ϕ and x_m.

The main idea behind the symbolic method can be summarized as follows: given a binary form

$$\phi(x, y) = \sum_{k=0}^{n} \binom{n}{k} a_k \, x^k \, y^{n-k} \tag{3.1}$$

we rewrite it as $\phi(x, y) = (\alpha_1 x + \alpha_2 y)^n$ where α_1 and α_2 are symbols which we agree to evaluate as $a_0 = \alpha_1^n, a_1 = \alpha_1^{n-1}\alpha_2^1, \ldots, a_n = \alpha_2^n$. Although this would seem to lead to great confusion, a moment of reflection will convince the reader that as long as the coefficients a_i appear linearly in a function no ambiguity can occur. However, if we consider quadratic terms in the a_i then one can show $a_0 a_2 = a_1^2$! To overcome this difficulty, a second set of symbols β_1 and β_2 are introduced. Hence in the example above $a_0 a_2 = \alpha_1^n \beta_1^{n-2} \beta_2^2$, and $a_1^2 = \alpha_1^{n-1} \alpha_2 \beta_1^{n-1} \beta_2$. A similar process is observed for higher degree terms in the a_i.

For any pair of symbols α, β a bracket is defined as $[\alpha\beta] = \alpha_1\beta_2 - \alpha_2\beta_1$. The fundamental theorem of invariant theory is that every invariant can be written as a product of brackets, and every product of brackets with correct number of symbols is an invariant.

There are at least three different interpretations or formal justifications of the symbolic method. The first approach, used by Grace and Young [59], suggest treating each of the symbols α_1, α_2 as operators $\partial/\partial x_\alpha$ and $\partial/\partial y_\alpha$ acting on their own copy of the polynomial $\phi(x_\alpha, y_\alpha)$. The second approach, given by Gurevich [49], relates the polynomial ϕ to a symmetric tensor $c_{ij\ldots k} x^i x^j \ldots x^k$. The symbols are then considered as place markers for various operations on the indices of c. This approach can also be seen, in a more abstract guise, in Dieudonné's book [60]. The third approach is that of Kung and Rota [61] who seek to justify the symbolic method directly via *umbral calculus.*

3.2.1 Umbral Notation

We now give a brief account of the work of Kung and Rota as it pertains to our own work. For simplicity of representation, we explain the notation for binary forms (that

is, homogeneous polynomials in two variables), the case for more than two variables follows through with trivial modifications. We refer the reader to the above article and the Sturmfels's book [62] for complete proofs of the theorems stated below.

Binary Forms and Invariants

Consider the binary form (3.1). After a linear transformation

$$\begin{aligned} x &= \xi_1 x' + \delta_1 y' \\ y &= \xi_2 x' + \delta_2 y' \end{aligned} \tag{3.2}$$

we obtain the expression

$$\phi'(x', y') = \sum_{k=0}^{n} \binom{n}{k} a_k (\xi_1 x' + \delta_1 y')^k (\xi_2 x' + \delta_2 y')^{n-k} = \sum_{k=0}^{n} \binom{n}{k} a_k x'_k y'_{n-k} \tag{3.3}$$

An invariant of weight w of a binary form of degree n is defined as a polynomial $I(x_0, \ldots, x_n)$ for which following identity holds:

$$I(a'_0, \ldots, a'_n) = (\xi_1 \delta_2 - \xi_2 \delta_1)^w I(a_0, \ldots, a_n) \tag{3.4}$$

An *absolute weight invariant* is an invariant of weight zero. In the same way, given a set of r binary forms $\phi_i(x, y)$ of degree $r(i)$

$$\phi_i(x, y) = \sum_{k=0}^{r(i)} \binom{n(i)}{k} a_{ik}\, x^k\, y^{n(i)-k} \quad i = 1, 2, \ldots, r \tag{3.5}$$

we define a *joint invariant of weight* w to be a polynomial $I(x_{10}, \ldots, x_{1n(1)}, \ldots, x_{rn(r)})$ for which

$$I(x'_{10}, \ldots, x'_{1n(1)}, \ldots, x'_{rn(r)}) = (\xi_1 \delta_2 - \xi_2 \delta_1)^w I(x_{10}, \ldots, x_{1n(1)}, \ldots, x_{rn(r)}) \tag{3.6}$$

where the primed variables are obtained in a manner similar to above. It can be shown that any invariant can be written as a sum of invariants homogeneous in each of the sets of coefficients. Here, we will consider only this sort of invariants.

The Umbral Calculus

Let Φ be an alphabet of an infinite set of letters. Each letter α is associated with two variables α_1, α_2. The *umbral space* U is the set of polynomials in these letters (viewed as variables).

The umbral operator $\Psi : U \to P$ is a linear operator from umbral space U to P, the ring of polynomials in the coefficients of ϕ; it obeys the following rules:

1. $\langle \Psi \mid \alpha_1^k \alpha_2^{n-k} \rangle = a_k$
2. $\langle \Psi \mid \alpha_1^i \alpha_2^j \rangle = 0 \ \ for \ \ i + j \neq 0$
3. $\langle \Psi \mid \alpha_1^i \alpha_2^j \beta_1^k \beta_2^l \ldots \gamma_1^m \gamma_2^n \rangle = \langle \Psi \mid \alpha_1^i \alpha_2^j \rangle \langle \Psi \mid \beta_1^k \beta_2^l \rangle \ldots \langle \Psi \mid \gamma_1^m \gamma_2^n \rangle$

Every polynomial $I(a_0, \ldots, a_n)$ has an *umbral representation* $Q \in U$ for which I is equal to $\langle \Psi \mid Q(a_1, \ldots, a_n) \rangle$, conversely I is called the *umbral evaluation* of Q.

Umbral notation for several forms is defined in a similar way. For each form $\phi_i(x, y)$, we take a distinct alphabet. The umbral operator is extended in the obvious fashion.

Bracket Polynomials and the Fundamental Theorem

Given symbols as above, we define their bracket $[\alpha\beta]$ to be $\alpha_1\beta_2 - \alpha_2\beta_1$. An element of U which can be expressed as a polynomial in these brackets will be termed a *bracket polynomial.* A *bracket monomial* M is a product of brackets $M = [\alpha\beta][\alpha\delta]\ldots[\omega\nu]$ where the total number of brackets is called the *weight* of the monomial. If a bracket polynomial consists entirely of monomials of a fixed weight w, then we say the polynomial is of weight w. Let B be the subspace of bracket polynomials in U, and B_w the space of bracket polynomials of weight w.

Theorem 1 (The fundamental theorem): Every invariant of weight w can be written as the umbral evaluation $\langle \Psi \mid P \rangle$ of a bracket polynomial P in B_w, and every such evaluation is an invariant of weight w.

Generating Invariants through the Standard Tableau

The fundamental theorem characterizes invariants in terms of bracket poynomials. However, if the theorem is to serve as a basis for the calculation of invariants, it becomes necessary to eliminate redundant expressions as much as possible. Such redundancy comes from two sources. First, an umbral expression may have several representations as bracket polynomials. The second problem is that two umbral expressions may evaluate to the same expression.

The first problem is resolved by introducing a lexicographic ordering to the letters of the alphabet(s). A bracket monomial $M = [\alpha\beta][\alpha\delta]\ldots[\omega\nu]$ can be viewed as a

tableau

$$\begin{pmatrix} \alpha & \beta \\ \alpha & \delta \\ \vdots & \\ \delta & \gamma \end{pmatrix} = [\alpha\beta][\alpha\delta]\ldots[\delta\gamma]$$

Such a tableau is called *standard* if letters are ordered increasingly from left to right, and nondecreasingly from top to bottom.

Theorem 2: The standard tableaux of weight w form a basis of B_w.

Thus, when generating invariants from tableaux, it is only necessary to consider those tableaux which are standard.

It is worth mentioning at this point that given an element in B_w, it is possible to rewrite the brackets in an algorithmic way into a linear combination of standard tableaux: the so-called *straightening algorithm* [62]. However, we will not need this in what follows.

As for the second problem, it can be shown [61] that an umbral expression evaluates to zero if and only if its symmetrisation with respect to equivalent letters is zero. Unfortunately, the process of symmetrisation and that of monomial ordering are mutually exclusive, thus we are led to seek different ways to ensure that duplicate invariants are discarded.

3.2.2 Joint, Affine and Euclidean Invariants

Joint invariants (also called mutual invariants) of several forms can be computed using the standard tableau as described in the previous subsection. Here, lexicographic ordering can be arbitrarily extended between different alphabets. Although there may be many latent applications of joint invariants in computer vision such as associated objects or objects represented through more than a single polynomial, our interest lies in how joint invariants allow one to obtain invariants for different groups of motion.

It can be shown (see Gurevich [49] for an excellent introduction) that the affine group of motions are exactly those which leave the line at infinity invariant (fixed). It follows that every affine invariant of a form can be written as a joint invariant of the form together with the line at infinity. In two dimensions, this amounts to finding the joint invariants of a ternary form with the linear form $\phi(x, y, z) = (\alpha_1 x + \alpha_2 y + \alpha_3 z)$ (in symbolic notation) where α_1 and α_2 evaluates to zero and α_3 evaluates to 1.

It is also possible to treat Euclidean invariants by the symbolic method. Here, we calculate joint invariants of the form, the line at infinity and the second order tensor

$$\eta^{ij} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

In order to do this we need a slight generalization of the symbolic method described so far. The details of this can be worked out directly from the framework given by Gurevich [49], which we hope to present elsewhere. It suffices for our purposes here to say that we need only to add a new type of bracket symbol $(\alpha\,\beta)$ which represents the product $\alpha_1\beta_1 + \alpha_2\beta_2$. All invariants can be expressed as sums of products of the two types of brackets [] and ( ) and vice versa.

It is possible to extend the tableaux mentioned above to contain these two types of brackets. Furthermore, if ζ, ξ are symbols representing the line at infinity, then it is easy to show that $[\alpha\,\beta\,\zeta][\gamma\,\delta\,\xi] = (\alpha\,\gamma)(\beta\,\delta) - (\alpha\,\delta)(\beta\,\gamma)$. Thus we need to consider at most one bracket of the form $[\alpha\,\beta\,\zeta]$.

3.2.3 Gröbner Bases and Hilbert's Basis Theorem

The main drawback to the symbolic method is that the same invariant may be generated many times. However, even if we exclude multiple copies of a given invariant we are still left with the problem that new invariants are just polynomials in the old invariants. But this problem is inherent in all other approaches used so far, as there is no analysis to remove such dependencies. Fortunately, there is a method to deal with this problem which takes us back to Hilbert's original solution to the existence of a finite number of invariants.

Recall that Hilbert's basis theorem (developed exactly for this purpose) says that given an infinite number of polynomials $\phi_1, \phi_2, \ldots$ in a fixed number of variables, there exists a number N such that for any $i > N$, ϕ_i can be written in the form

$$\phi_i = \sum_{k=1}^{N} \rho_k \phi_k \tag{3.7}$$

with the ρ_i polynomials in the same variables. Of course, this is not the same as saying that ϕ_i is a function of $\phi_1, \ldots, \phi_N$. However in the case where the ϕ_i are all invariants, it is possible to 'average' the coefficients ρ_i in such a way that they are also invariants.

Suppose that we have calculated all homogeneous invariants of order less than m. If a new invariant lies in the ideal generated by the ones already calculated, then we have a relation of the form (3.7) with the ρ_i invariants. It is easy to see that the degrees of the ρ_i are less than m and hence (3.7) expresses the fact that our new invariant is merely a polynomial in the invariants calculated so far.

To use this fact, we need to have some information about the ideal generated by our invariants. This can be obtained by the calculation of a Gröbner basis of the invariants. If a new invariant lies in the ideal of these invariants then its reduction modula this basis will be zero. If not, then the invariant provides new information and should be added to the list of invariants. A new Gröbner basis should then be calculated. Gröbner basis packages which allow these calculations are available in most computer algebra packages.

3.2.4 Implementation

Implementation of the algorithm was done by the REDUCE symbolic computation software [63]. The two inputs to the algorithm are the degree of the form (or degrees for joint invariants) and the maximum degree for the invariants sought. The algorithm performs three tasks: generation of standard tableaux, conversion of these into polynomials in the coefficients of the form, and prunning of redundant invariants.

The tableau generation phase exhaustively produces all possible tableaux up to a given height w, determined by the two inputs to the algorithm, iterating through the number of symbols, also determined by the inputs. Since each bracket consists of k symbols for k-variable forms (2 for binary, 3 for ternary), the total degree of symbols in a tableau of height w is kw. If i is the degree of the invariant (number of symbols) then $w = ni/k$. Similarly for the joint invariant of r different forms $w = (n_1 i_1 + \ldots + n_r i_r)/k$ where $n_1, \ldots, n_r$ are degrees of forms and $i_1, \ldots, i_r$ are the degrees of the invariant in the coefficients of each form. In the second phase, the standard bracket monomials are multiplied out to obtain polynomials in the umbral symbols. These in turn are converted to polynomials in the coefficients of the form, using the umbral evaluation rules implemented through the substitution/pattern matching mechanism of the REDUCE. The final step takes each new invariant and checks whether it can be expressed as a polynomial of previous invariants. This, as explained above is done by reducing the invariant modulo the current Gröbner basis. If the reduction is non-zero

then the invariant is added to the list, and a new Gröbner basis is calculated.

3.3 Experiments

In this section we will demonstrate how some of the invariants generated by our algorithm can be used for recognizing objects from 2D images. First, a set of some relative affine invariants of fourth degree implicit curves generated by the algorithm will be presented along with their umbral space representations, and then they will be weight and rank normalized for obtaining absolute invariants (see Section 3.3.2). Finally these absolute invariants will be tested for discriminating 10 arbitrary shapes, represented by implicit polynomials, in the corresponding invariant space by creating a number of random affine transformations.

3.3.1 Some Affine Invariants and their Umbral Representations

In our experiments we used the following relative affine invariants of fourth degree implicit polynomials generated by the explained algorithm:

$$\mathcal{I}_1 = 45a_{13}^2a_{20}^2 - 30a_{12}a_{13}a_{20}a_{21} + 3a_{12}^2a_{21}^2 + 6a_{11}a_{13}a_{21}^2 + 48a_{04}a_{20}a_{21}^2 - 12a_{03}a_{21}^3 + 20a_{12}^2a_{20}a_{22} - 30a_{11}a_{13}a_{20}a_{22} - 120a_{04}a_{20}^2a_{22} - 16a_{11}a_{12}a_{21}a_{22} + 54a_{10}a_{13}a_{21}a_{22} + 12a_{03}a_{20}a_{21}a_{22} + 20a_{02}a_{21}^2a_{22} + 17a_{11}^2a_{22}^2 - 36a_{10}a_{12}a_{22}^2 - 8a_{02}a_{20}a_{22}^2 - 36a_{01}a_{21}a_{22}^2 + 72a_{00}a_{22}^3 - 12a_{12}^3a_{30} + 54a_{11}a_{12}a_{13}a_{30} - 162a_{10}a_{13}^2a_{30} - 72a_{04}a_{12}a_{20}a_{30} + 54a_{03}a_{13}a_{20}a_{30} - 72a_{04}a_{11}a_{21}a_{30} + 54a_{03}a_{12}a_{21}a_{30} - 72a_{02}a_{13}a_{21}a_{30} + 432a_{04}a_{10}a_{22}a_{30} - 72a_{03}a_{11}a_{22}a_{30} + 12a_{02}a_{12}a_{22}a_{30} + 54a_{01}a_{13}a_{22}a_{30} - 81a_{03}^2a_{30}^2 + 216a_{02}a_{04}a_{30}^2 + 6a_{11}a_{12}^2a_{31} - 36a_{11}^2a_{13}a_{31} + 54a_{10}a_{12}a_{13}a_{31} + 180a_{04}a_{11}a_{20}a_{31} - 72a_{03}a_{12}a_{20}a_{31} + 54a_{02}a_{13}a_{20}a_{31} - 324a_{04}a_{10}a_{21}a_{31} + 54a_{03}a_{11}a_{21}a_{31} - 30a_{02}a_{12}a_{21}a_{31} + 54a_{01}a_{13}a_{21}a_{31} + 54a_{03}a_{10}a_{22}a_{31} - 30a_{02}a_{11}a_{22}a_{31} + 54a_{01}a_{12}a_{22}a_{31} - 324a_{00}a_{13}a_{22}a_{31} + 54a_{02}a_{03}a_{30}a_{31} - 324a_{01}a_{04}a_{30}a_{31} + 45a_{02}^2a_{31}^2 - 162a_{01}a_{03}a_{31}^2 + 972a_{00}a_{04}a_{31}^2 - 36a_{04}a_{11}^2a_{40} + 432a_{04}a_{10}a_{12}a_{40} - 72a_{03}a_{11}a_{12}a_{40} + 48a_{02}a_{12}^2a_{40} - 324a_{03}a_{10}a_{13}a_{40} + 180a_{02}a_{11}a_{13}a_{40} - 324a_{01}a_{12}a_{13}a_{40} + 972a_{00}a_{13}^2a_{40} + 216a_{03}^2a_{20}a_{40} - 576a_{02}a_{04}a_{20}a_{40} - 72a_{02}a_{03}a_{21}a_{40} + 432a_{01}a_{04}a_{21}a_{40} - 120a_{02}^2a_{22}a_{40} + 432a_{01}a_{03}a_{22}a_{40} - 2592a_{00}a_{04}a_{22}a_{40}$$

$$\mathcal{I}_2 = 6a_{22}a_{22}a_{22} - 27a_{13}a_{22}a_{31} + 81a_{04}a_{31}a_{31} + 81a_{13}a_{13}a_{40} - 216a_{04}a_{22}a_{40}$$

$$\mathcal{I}_3 = 144a_{40}a_{04}a_{00} - 36a_{40}a_{03}a_{01} + 12a_{40}a_{02}a_{02} - 36a_{31}a_{13}a_{00} + 9a_{31}a_{12}a_{01} - 6a_{31}a_{11}a_{02} + 9a_{31}a_{10}a_{03} + 9a_{30}a_{13}a_{01} - 6a_{30}a_{12}a_{02} + 9a_{30}a_{11}a_{03} - 36a_{30}a_{10}a_{04} + 12a_{22}a_{22}a_{00} - 6a_{22}a_{21}a_{01} + 4a_{22}a_{20}a_{02} - 6a_{22}a_{12}a_{10} + 2a_{22}a_{11}a_{11} + 2a_{21}a_{21}a_{02} - 6a_{21}a_{20}a_{03} + 9a_{21}a_{13}a_{10} - a_{21}a_{12}a_{11} + 12a_{20}a_{20}a_{04} - 6a_{20}a_{13}a_{11} + 2a_{20}a_{12}a_{12}$$

$$\mathcal{I}_4 = 120a_{40}a_{04} - 30a_{31}a_{13} + 10a_{22}a_{22}$$

$\mathcal{I}_3$ is actually a projective invariant of fourth degree ternary forms and its umbral representation is $\mathcal{I}_3 = [\alpha\,\beta\,\gamma]^4$. $\mathcal{I}_1$ is an affine invariant obtained by the joint invariant

Invariant	Weight	Rank
$\mathcal{I}_1$	6	4
$\mathcal{I}_2$	6	3
$\mathcal{I}_3$	4	3
$\mathcal{I}_4$	4	2

Table 3.1: Weight and ranks of the invariants

technique and its umbral representation is $\mathcal{I}_1 = [\alpha\beta\gamma]^2[\alpha\beta\delta]^2[\gamma\delta\ \alpha'][\gamma\delta\ \beta']$, where $\alpha\beta$, γ, δ are the symbols for the $4th$ degree form, and $\alpha'\ \beta'$ are used to represent the line at infinity. $\mathcal{I}_2$ and $\mathcal{I}_4$ are affine invariants of the fourth degree implicit polynomials which in fact are just projective invariants of the fourth order binary form which represents the highest order terms of the polynomial. Their umbral representations are $\mathcal{I}_2 = [\alpha\beta]^2[\alpha\gamma]^2[\beta\gamma]^2$, and $\mathcal{I}_4 = [\alpha\beta]^4$.

3.3.2 From Relative to Absolute Affine Invariants

If an invariant is to be used in object recognition under affine and projective transformation of the image plane, it should be an absolute weight invariant (see Section 3.2.1). Note that in the Euclidean case, every invariant is automatically an absolute weight invariant. This is not the only complication however. An implicit polynomial $f(x,y)$ multiplied by a nonzero constant k still represents the same zero set, i.e., $Z(kf(x,y)) = Z(f(x,y))$. Yet, the value of the invariant should be independent of whether the fitting algorithm used produces the coefficient vector $\mathbf{c}$ or any of its scalar multiples $\mathbf{c}' = k\mathbf{c}$. A homogeneous invariant I of degree d will satisfy $I(\mathbf{c}') = k^d I(\mathbf{c})$, and is called a *relative rank invariant* of *rank d*. When $d = 0$, I is called an *absolute rank invariant*. Table 3.1 shows the weight and ranks of $\mathcal{I}_1, \ldots, \mathcal{I}_4$.

Finally, the below two absolute invariants, i.e., invariant which are absolute weight and rank invariants, are obtained from $\mathcal{I}_1, .., \mathcal{I}_4$ through weight and rank normalization, i.e. forming a function of the relative invariants where the weight and rank factors are cancelled out.

$$I_1 = \frac{\mathcal{I}_1 \mathcal{I}_4}{\mathcal{I}_2 \mathcal{I}_3} \tag{3.8}$$

$$I_2 = \frac{\mathcal{I}_1^2}{\mathcal{I}_3^2 \mathcal{I}_4} \tag{3.9}$$

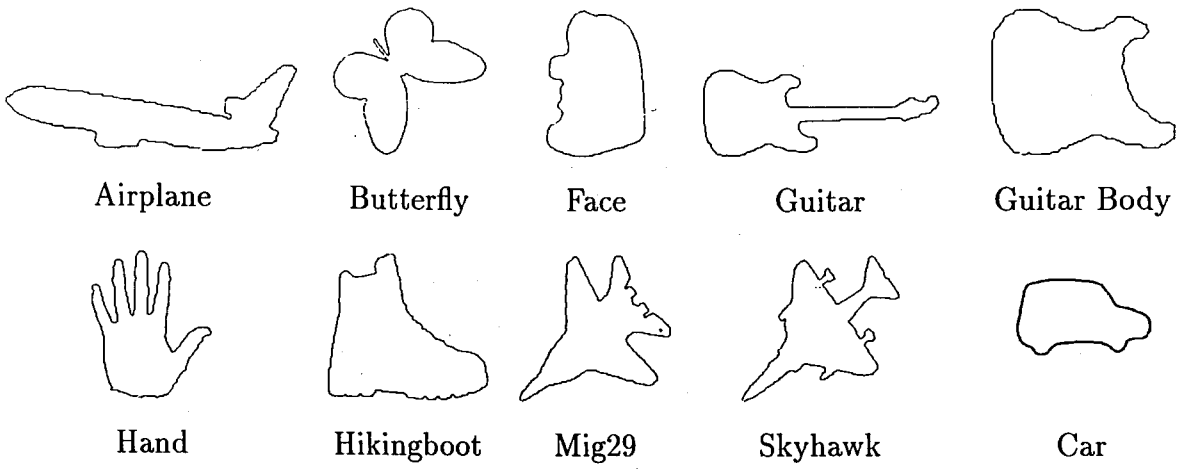


Figure 3.1: Objects used in the experiments.

3.3.3 Experiments: Building a Sample Shape Database

In the experiments the 10 arbitrary shapes shown in Figure 3.1 are used. The scenario is to built a sample shape database where the objects are represented by $4th$ degree implicit polynomials and the indexing into the database is done through the absolute invariants I_1 and I_2 of the previous section. To this end, for each shape (shape data set) random affine transformations are generated and using the 3L fitting algorithm, $4th$ degree implicit polynomials are fitted to the transformed data sets. The fitting algorithm is not fully affine invariant and for the transformed data set the resulting polynomial may be far off from being an affine transformation of the implicit polynomial for the original shape. To discard such anomalous cases, we compare the two polynomials, after transforming them into the same reference frame (as the affine transformation between them is known), as coefficient space vectors and weed out those transformations of the data set for which the alignment is way off. The alignment criterion is to compute the angle between the two coefficient vectors and discard those greater than a chosen threshold angle, 90 degrees in our case. Finally, to form the invariant space representations –indices into the database– I_1 and I_2 are computed from the $4th$ degree implicit polynomials fitted to the eligible transformations of the data sets. For each shape 350 such transformations and the resulting invariant space representations are included.

Figure 3.2 shows an affine transformation of the car and the hikingboot, and the implicit polynomial fittings to the original and transformed data sets. For generating

Figure 3.2: Implicit polynomial fittings to car and hikingboot, and one of their affine transformations. Small dots are the data sets, darker lines are the zero set of the best fit 4th degree IP models.

random affine transformations the equivalent singular value decomposition is used.

$$\begin{bmatrix} x_t \\ y_t \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \begin{bmatrix} cos\varphi & -sin\varphi \\ sin\varphi & cos\varphi \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix}$$

Here, θ, φ are random variables in $[0, 2\pi]$, λ_1, λ_2 are in $[0.5, 2]$, and t_x, t_y are in $[-10, 10]$. In reference to realistic image plane transformations, these ranges constitute a physically meaningful subset of all affine transformations. For instance λ_1, λ_2 values constrain that the objects centered at the origin, which is the case with our object data sets, can be shrunk or enlarged by a factor of at most 2.

Figure 3.3 shows the manifolds generated by some of the shapes in the invariant space spanned by I_1 and I_2. Here, each point corresponds to one of the 350 affine transformations. Note that ideally, each shape should correspond to a single point in the invariant space. However, as alluded to above the 3L fitting is not fully affine invariant, and in the invariant space the resulting error maps the shape representation into clouds rather than to a unique point.

Figure 3.4, at two different scales (resolutions), shows how the shapes are located in the same space with respect to each other. The results indicate a clear seperation of the shape boundaries in the invariant space. It should be reminded that the invariant space used in this experiment is actually a subspace (or a 2D projection) of all possible invariants. Therefore, adding more invariants would increase the dimension of the classifying space, and with it the separation of these objects.

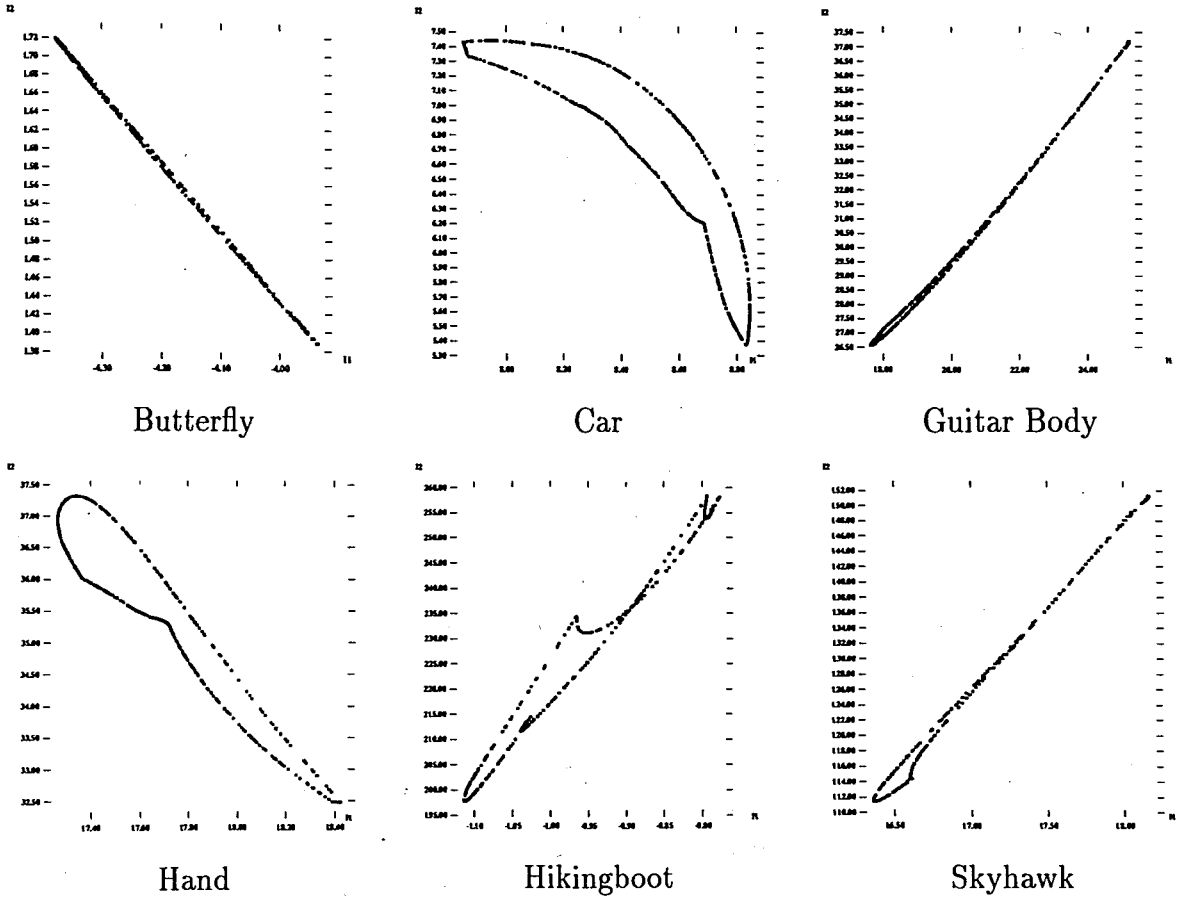
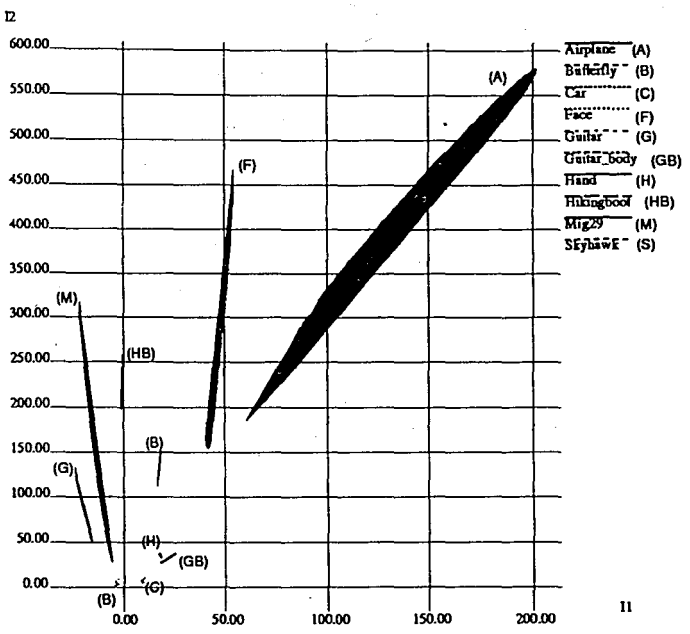
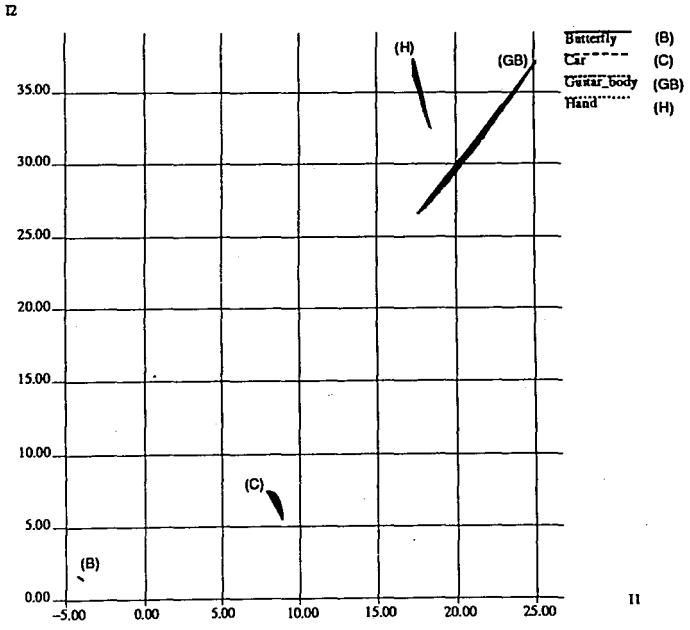


Figure 3.3: Individual invariant space representations for some of the shapes.



(a)



(b)

Figure 3.4: (a) Invariant space representations for all the objects. (b) Invariant space representation for buttefly, car, guitar body and hand

3.4 Assessment of Results

We have described how it is possible to use the symbolic method of classical invariant theory to calculate projective, affine, Euclidean and joint invariants for polynomial curves and surfaces. The main advantages of the method were shown to be its generality and completeness, (i.e., given polynomials (2-D or 3-D) of any degree, the method computes all invariants up to any given degree), and the fact that the invariants are produced directly, without the need for elimination of auxiliary variables. We have also demonstrated how these results can be applied in computer vision for object recognition.

Our experiments have shown that for low degree polynomials (2 to 4) the algorithm is very efficient. However, for higher degrees the performance decreases dramatically, due to the height of the tableaux required and its conversion into algebraic form. As with all computations of this nature, expressions often exceed the limits of the hardware available, and there is a need for a more detailed study before the relative merits for this method can be ascertained properly. However since the invariant calculations are carried out off-line, the time is not a binding constraint for real time object recognition. We indicate some necessary areas of improvement below.

On the computer algebra side, two problems need to be addressed. First the optimization of the tableaux generation, which is very rudimentary at present. Secondly, to code an efficient algorithm for conversion of the symbolic representation. At the moment we have depended on native pattern matching algorithms in REDUCE.

From the computer vision standpoint, efficient use of invariant space representations of objects for recognition is an area of active research. In the past, a Bayesian framework has been proposed. This technique, is based on a model for the probability distributions of the polynomial coefficients, and its extension to invariants has certain drawbacks [5]. An alternative approach is to view the invariant space representation of each object as arbitrary 'clouds' of sample points and applying vector quantization or other standard pattern classification techniques. However, since most invariant space representations seem to have an ellipsoidal signature, one can simply take the minimal ellipsoid which covers each objects 'cloud'. An object can then be identified on substitution of the invariants into the appropriate ellipsoidal equations.

Chapter 4

Closed-form Object Pose Estimation using Algebraic Shape Representations

4.1 Pose Estimation and Implicit Polynomials

Aligning objects in two images is a central fundamental problem in several disciplines ranging from computer vision, robotics, industrial inspection to photogrammetry. Among computer vision problems requiring efficient and robust alignment, sometimes referred to as 2D-2D pose estimation, are recognition of objects from two images, recovering camera motion between video frames for 3D reconstruction, video mosaicing and visual inspection for industry [36, 64, 65, 9]. In abstract terms, the object alignment involves finding the appropriate coordinate transformation between the images of an object. The coordinate transformation may be, in the order of generality, a rigid motion, a similarity transformation, an affine, or a projective transformation. Each forms an algebraic group and includes the less general ones as subgroups. Some transformations not having the group property, such as non-uniform scaling or nonlinear ones such as bilinear, pseudoperspective, are also of interest especially as approximations to more complex groups.

Scatter matrix alignment is a well-known object alignment technique where eigenvectors of the scatter matrix computed from the data points are used for finding the transformation [66]. Another popular point-based approach derives closed-form formu-

lae for data set alignment based on point correspondences over the images and least squares methods [67, 45]. In much of the recent work, projective geometry has been used for structure and motion estimation [51]. In this framework image correspondence is expressed through the fundamental matrix computed from a set of identifiable point feature correspondences- which accounts for the epipolar geometry of the two images [64, 68]. The main drawback of the point-based methods is that they are sensitive to even amounts of missing data. In the case of scatter matrices, the empirical covariance matrices may differ widely, as may also happen with the center of mass which is used for estimating the translational component. The closed-form solutions and fundamental matrix approach however presuppose a reliable point correspondence. A common problem is that almost none of the methods involves any smoothing, which makes them further susceptible to noise related problems. Another group of frame alignment techniques aims essentially at finding the coordinate transformation between video frames from the same footage for motion recovery and is inspired by the idea of optical flow; under this category falls the comprehensive treatment of Mann and Picard [65].

The methods more closely related to our pose estimation technique are those directed towards positioning of 2D or 3D objects, modeled through algebraic surfaces and curves, from monocular image contours. Along these lines, Ponce and et al. [46] proposed a 3D pose estimation technique, and Taubin formulated the canonical frame of reference [36]. Taubin's work is based on the fact that algebraic curve and surface representation naturally gives rise to an object-based canonical frame of reference, or intrinsic frame of reference, consisting of an intrinsic coordinate center and intrinsic coordinate orientation. The coordinate system thus defined is a covariant function of the coefficients of the implicit polynomial representation for the object. The idea is a generalization of the case for conics where the center of symmetry constitutes the intrinsic center and the eigenvectors of the matrix associated with the second degree terms can be taken to be the intrinsic orientation. A formulation of canonical frame of reference for affine case exists for moments of data sets and for implicit polynomial curves [37].

This chapter presents a novel model-based 2D-2D closed-form algebraic solution for coordinate transformation estimation for Euclidean, similarity, and non-uniform scale transformations. The proposed closed-form algebraic pose estimation expressions, involving only the implicit polynomial coefficients for the model and the input objects, are drawn from an observation of how the implicit polynomial object model transforms

under different coordinate transformations. The approach is generalizable to pose estimation with higher order polynomials and to 3D-3D pose estimation.

The closed-form solution provides a fast, efficient pose estimation technique, and enjoys reliability particularly made possible, as is explained in the 2nd chapter, by the recent advances in implicit polynomial fitting to objects as we have more reliable and robust fittings. The emphasis on the algebraic and geometric structure allows us to compare objects with varying surface patterns or colors under different lightenings- a condition which largely bars the application of optical flow methods as these methods rely on the assumption that pixel intensities of the two 2D projections of a 3D point are approximately the same. Besides these, the method has all the other advantages of a model-based formulation, such as not requiring an accurate point correspondence, resistance to modest missing data because of the interpolation power of the algebraic object representation and relative insensitivity to noise and perturbations because of the smoothing involved in fitting such models. Various experiments are run for objects with missing data and small perturbations -the two common unsolved problems in almost every computer vision and image processing application- to test the strength of the method in dealing with them. Finally, an analytical stability analysis method to determine the robustness of the solution around the neighborhood of the model coefficients, and point out its connection with the classical statistical pattern recognition approach are proposed.

4.2 Closed-form Coordinate Transformation Estimation

4.2.1 Why model-based geometric shape alignment?

Given two images of the same 3D object taken from different directions, we want to know if there is a single-valued and invertible transformation T that maps one image to the other; and if this mapping exists, what can be said about T. This is not a well-defined problem in general. For instance, one image may contain other objects or different parts of the same object. Different lighting conditions may create another problem. Here, we assume the images are from the same planar shape structure (or approximately planar silhouettes of the 3D object surface), and segmentation has been

applied to generate the (x, y) edge map. We further assume that each data set is represented by a geometric model (here, the IP model) and there exists a transformation T that transforms one model to the other.

The advent of geometric models has brought about two advantages. First, point-wise correspondence is no longer necessary; point-wise correspondence determination is especially difficult when there is a scale change or two images have different number of points. Second, it makes the object alignment relatively less sensitive to outliers of the data set due to noise in the process of image data formation. It can be also used to handle inexact matching situations when there is missing data due to object occlusion or lighting condition changes. Besides these, as we will show, there is a natural correspondence between the IP models of the two data sets and the transformation relating them. This makes it an appealing tool for 2D-2D pose estimation.

However, as with any other model based approach, the IP model faces two potential problems. One is the robustness of the representation. The matching is unreliable if the underlying model is noise sensitive. The other is the so-called over-fit problem, i.e., the model contains more information than the original data set. Hence, a perfect match of the models might not correspond to a meaningful match of the data sets. The first problem has been solved by the new IP model fitting procedures as they are able to generate robust, repeatable, and physically meaningful IP models. As for the over-fit problem associated with the model-based approach, we propose a stability analysis technique which measures how well the models can be used to generate a good matching result. The technique aims to detect possible multiple alignments in over-fitting and to set the criterion for proper model modification.

4.2.2 General Approach

Having chosen IP as the model for the object alignment problem, and discussed the pros and cons of this approach, we can begin to explore the relation between IP models and coordinate system transformations, For an IP model of given degree n, there are $1 + 2 + 3 + ... + (n + 1) = \frac{(n+1)(n+2)}{2}$ coefficients as the model parameters. Assume that a general affine transformation relating two coordinate systems $\{O_1, X_1, Y_1\}$ and $\{O_2, X_2, Y_2\}$:

$$\begin{cases} x' = ax + by + h \\ y' = cx + dy + k \end{cases} \tag{4.1}$$

and $f_1 = \sum_{0 \le i,j;i+j \le n} a_{ij} x_1^i y_1^j$ and $f_2 = \sum_{0 \le i,j;i+j \le n} b_{ij} x_2^i y_2^j$ are the IP models for an object in coordinate systems 1 and 2 respectively, are given. The coefficient vectors $\{a_{ij}\}$ and $\{b_{ij}\}$ of the IP models contain the source information for aligning the object in these coordinate systems. A careful inspection of the model would show that leading term coefficients, coefficients of the homogeneous *nth* degree part, of the IP model are sufficient for the alignment. One of the reasons for using only the leading term coefficients is their invariance to translation which immediately reduces the number of transformation parameters involved from six to four. Moreover, there are only $n+1$ leading term coefficients instead of $\frac{(n+1)(n+2)}{2}$ for the whole coefficient vector. Because of the homogeneity of the leading form, the leading terms can be normalized without changing the object shape (zero set of the IP model), for instance simply by setting the norm of the leading form coefficients vector equal to one. Finally, the leading term coefficients are in general more robust than the lower order terms. Because of the higher order monomials associated with them, a small amount of change in the leading term coefficients tends to generate more dramatic change in the zero set. Hence, leading term coefficients are less susceptible to the noise in the data formation and the fitting processes.

Determination of translational part

The general affine transformation of (4.1) has six parameters: the linear part ($a, b, c,$ and d) and the translational part (h and k). In pose estimation, the translational part can be treated independently from the linear part. Once the linear part is determined using the leading coefficients, the translation can be estimated using the rest of the coefficients. For example, denote the linear part of the transform as T and assume that coefficients a_{ij} of f_1 are transformed into $c_{ij} = p(T, a_{ij})$ (a polynomial function of T and a_{ij}) under T. We can choose h and k such that the following error is minimum:

$$\sum_{0 \le i,j;i+j \le n-1} \delta(i+j)(b_{ij} - c_{ij})^2 = \sum_{0 \le i,j;i+j \le n-1} \delta(i+j)(b_{ij} - p(T, a_{ij}))^2$$

where δ is a weight factor corresponding to the degree of a monomial. It is used to compensate the fact that different degree monomials have different levels of sensitivity

to noise. The zero*th* monomial (constant term) is the least reliable one since all the other monomials (degree higher than 0) will expand and generate a real term under the translation. These real terms eventually add up into the zero*th* monomial. Hence, $\delta(0)$ should be set very small.

Another way of determining the translation from the polynomial representation is to use the intrinsic object (coordinate) centers. The intrinsic center is a covariant vector, and a coordinate transformation maps the intrinsic centers. Thus the difference of the two intrinsic object center vectors would give the translation. For the general formal definition of the intrinsic coordinate center the reader is referred to [36]. In our experiments we used the conic center of symmetry -from which the idea of intrinsic center was originally derived and generalized- given below:

$$\begin{cases} x_0 = \frac{g_2 g_4 - 2 g_1 g_5}{4 g_3 g_5 - g_4 g_4} \\ y_0 = \frac{g_1 g_4 - 2 g_2 g_3}{4 g_3 g_5 - g_4 g_4} \end{cases} \tag{4.2}$$

here, (x_0, y_0) is the conic center for conic $g_0 + g_1 x + g_2 y + g_3 x^2 + g_4 xy + g_5 y^2 = 0$. In experiment section 4.4, we compare the method of conic center for translation determination with the traditional mass center method, i.e., $x_0 = \frac{\sum_i x_i}{N}, y_0 = \frac{\sum_i y_i}{N}$.

We now will formulate some closed-form transformation estimations for different coordinate transformation groups in the order of increasing generality.

Euclidean transformations

Assume that $f_1 = \sum_{0 \le i,j; i+j \le n} a_{ij} x_1^i y_1^j$ and $f_2 = \sum_{0 \le i,j; i+j \le n} b_{ij} x_2^i y_2^j$ are the nth degree IP models for an object in coordinate systems 1 and 2, respectively, with a Euclidean transformation between them. As explained before, translational terms can be ignored if we use leading term coefficients. The rotation angle between two coordinate systems is the only unknown here. Let $T = T(\theta)$ be a rotation of the coordinate system resulting in $(x_1, y_1) = T(x_2, y_2)$. Then $f_1 o T$ and f_2 are the IP representations of the same object in the coordinate system 2, they should be the same IP representation up to a scale factor, i.e., $f_1 o T = k\ f_2$ for some nonzero constant k. Denote the coefficients of $f_1 o T$ as a'_{ij}, which are function of $\mathbf{A}$ and θ, where $\mathbf{A}$ is a vector having the coefficients a_{ij} as its components. We then have a family of equations:

$$\frac{a'_{ij}}{b_{ij}} = \frac{a'_{kl}}{b_{kl}} \tag{4.3}$$

where $i+j=k+l=n$ and $(i,j)\neq(k,l)$. The simultaneous solution of the equations gives the estimated rotation angle θ. We illustrate the idea by an example from conics.

Example: If the original data set is represented by

$$f_1(x_1,y_1)=a_{20}x_1^2+a_{11}x_1y_1+a_{02}y_1^2+a_{10}x_1+a_{01}y_1+a_{00}$$

and

$$f_2(x_2,y_2)=b_{20}x_2^2+b_{11}x_2y_2+b_{02}y_2^2+b_{10}x_2+b_{01}y_2+b_{00}$$

is the IP representation after the Euclidean transformation

$$\begin{bmatrix}x_1\\y_1\end{bmatrix}=\begin{bmatrix}cos\theta & -sin\theta\\ sin\theta & cos\theta\end{bmatrix}\begin{bmatrix}x_2\\y_2\end{bmatrix}+\begin{bmatrix}t_x\\t_y\end{bmatrix}=cos\theta\begin{bmatrix}1 & -k\\ k & 1\end{bmatrix}\begin{bmatrix}x_2\\y_2\end{bmatrix}+\begin{bmatrix}t_x\\t_y\end{bmatrix}$$

where $k=\frac{sin\theta}{cos\theta}$. Making the substitution into f_1 we obtain

$$f_1(T(x_2,y_2))=A_{20}x_2^2+A_{11}x_2y_2+A_{02}y_2^2+A_{10}x_2+A_{01}y_2+A_{00}$$

where A_{ij} is a function of a_{ij} and the transformation parameters. In particular,

$$\begin{cases} A_{20} &= a_{20}cos^2\theta+a_{11}cos\theta sin\theta+a_{02}sin^2\theta=cos^2\theta(a_{20}+a_{11}k+a_{02}k^2)\\ A_{11} &= -2a_{20}cos\theta sin\theta-a_{11}sin^2\theta+2a_{02}cos\theta sin\theta+a_{11}cos^2\theta\\ &= cos^2\theta(-2a_{20}k-a_{11}k^2+2a_{02}k+a_{11})\\ A_{02} &= a_{20}sin^2\theta-a_{11}cos\theta sin\theta+a_{02}cos^2\theta=cos^2\theta(a_{20}k^2-a_{11}k+a_{02})\end{cases}$$

From $\frac{A_{20}}{b_{20}}=\frac{A_{11}}{b_{11}}$ and $\frac{A_{20}}{b_{20}}=\frac{A_{02}}{b_{02}}$ we obtain the following equations in k:

$$\begin{cases}(b_{20}a_{11}+b_{11}a_{02})k^2+(b_{11}a_{11}-2b_{20}a_{02}+2b_{20}a_{20})k+(b_{11}a_{20}-b_{20}a_{11})=0\\ (-b_{20}a_{20}+b_{02}a_{02})k^2+(b_{20}a_{11}+b_{02}a_{11})k+(b_{02}a_{20}-b_{20}a_{02})=0\end{cases}\tag{4.4}$$

The simultaneous solution of (4.4) gives the estimated value k:

$$k=\frac{-(a_{20}+a_{02})(b_{11}a_{02}+b_{20}a_{11}-b_{02}a_{11}-b_{11}a_{20})}{2b_{20}a_{20}{}^2-2a_{02}b_{20}a_{20}+b_{20}a_{11}{}^2-2b_{02}a_{02}a_{20}+a_{20}b_{11}a_{11}+2b_{02}a_{02}{}^2+b_{11}a_{02}a_{11}+a_{11}{}^2b_{02}}$$

and $\theta=arctan(k)$. However, $k=\frac{sin\theta}{cos\theta}$ substitution is not defined around 0 degrees. To handle this problem a second solution is computed in a similar way by using $k=\frac{cos\theta}{sin\theta}$ substitution, which is undefined around 90 degrees- and $\theta=arccot(k)$. The true rotation angle is determined by computing the two solutions, reorienting the data set with respect to the estimates and taking the best match.

Pose estimators for higher order polynomials are derived exactly the same way. In our experiments we used conics and fourth degree polynomial based pose estimators. The result for fourth degree Euclidean pose estimation is listed in Appendix A.

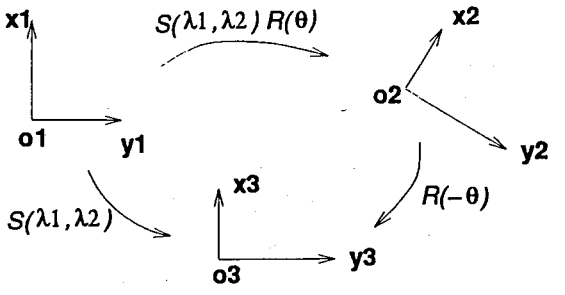


Figure 4.1: Separation method: a composed non-uniform transformation can be decomposed into two separate transformations, rotation and stretching.

Similarity transformations

The expression for the similarity transformation is: $(x_1, y_1) = \lambda R(\theta)(x_2, y_2)$ or

$$x_1 = \lambda cos\theta x_2 - \lambda sin\theta y_2 + h$$

$$y_1 = \lambda sin\theta x_2 + \lambda cos\theta y_2 + k$$

Since only the leading term coefficients are used, after substituting (x_1, y_1) in f_1, all the coefficients will have a common factor λ^n, where n is the degree of the IP model. This common factor does not change the simultaneous equation system (4.3). Hence, closed-form solution for Euclidean transformations can be used for similarity transformations to estimate the rotation angle. The unknown scale factor can be obtained by rotating back the transformed data set and then comparing the size of two objects. Alternatively, one can obtain the scale factor from the polynomial coefficients by minimizing the following error metric:

$$\sum_{0 \le i,j;i+j \le n} \delta(i+j)(b_{ij} - c_{ij}\lambda^{i+j})^2 = \sum_{0 \le i,j;i+j \le n} \delta(i+j)(b_{ij} - p(T, a_{ij})\lambda^{i+j})^2$$

here, again δ is a weight factor and $T = T(\theta)$ is rotation.

Non-uniform scale transformations.

Non-uniform scale transformation can be written as:

$$(x_2, y_2) = S(\lambda_1, \lambda_2)R(\theta)(x_1, y_1) \quad \text{where } S(\lambda_1, \lambda_2) \quad = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}$$

Direct substitution leads to a complex simultaneous equation system. Instead we use a separation method which dissociates the effect of rotation and stretching (Figure 4.1). For this an intermediate coordinate system (x_3, y_3) is defined such that

$$(x_3, y_3) = S(\lambda_1, \lambda_2)(x_1, y_1)$$

$$(x_3, y_3) = R(-\theta)(x_2, y_2)$$

and setting up the simultaneous equations in the coordinate system of (x_3, y_3). This separates the two set of parameters, i.e., rotation and stretching, and the equations are simplified. Using the Gaussian elimination method for the equations a $6th$ degree polynomial equation is obtained. The roots are equal to $tan(\theta)$. Therefore there could be up to six real roots. Each real root should be tested for the true rotation angle. Because of the space limitation, we do not list all the final expression here. However, those who are interested can obtain the complete list from the authors.

After the rotation is found, this information can be used to rotate the IP model back and compare the coefficients for the scaling factors λ_1 and λ_2, i.e., by minimizing the following error metric:

$$\sum_{0 \le i,j;i+j \le n} \delta(i+j)(b_{ij} - c_{ij}{\lambda_1}^i{\lambda_2}^j)^2 = \sum_{0 \le i,j;i+j \le n} \delta(i+j)(b_{ij} - p(T, a_{ij}){\lambda_1}^i{\lambda_2}^j)^2$$

here δ and $T = T(\theta)$ are the same as before.

In some cases, non-uniform transformation is a good approximation for affine transformation. Certainly non-uniform scale transformation is enough for the conics as an affine transformation would not change the type of the conic, i.e., an ellipse remains an ellipse etc.

4.3 Stability Analysis

The closed-form solutions are dependent on the two coefficient sets of the IP models. Although 3L fitting generates an IP model that is stable and robust with respect to transformation and perturbation of data set, some small variations will inevitably exist in the coefficients of fitted IP models due to digitization effects and other errors in image acquisition. In general, the model S that stored in the database is just one copy of the object shape from a class of geometric shapes that are close to S (Figure 4.2) in Hausdorff metric, which is defined as:

$$dist(S_1, S_2) = max_{z_1 \in S_1}(min_{z_2 \in S_2} \parallel z_1 - z_2 \parallel) + max_{z_2 \in S_2}(min_{z_1 \in S_1} \parallel z_1 - z_2 \parallel)$$

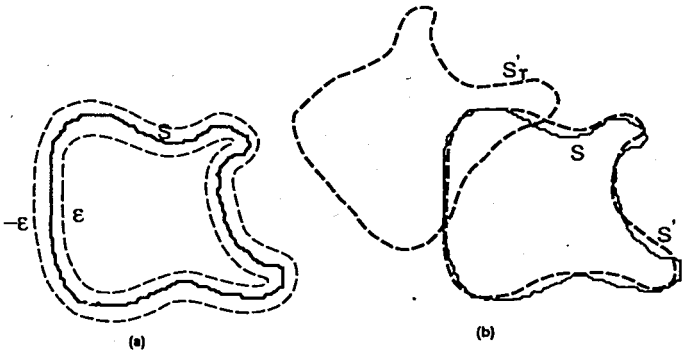


Figure 4.2: (a) A class of shapes that are within $-\epsilon$ and ϵ of model shape S are indistinguishable in Hausdorff metric. (b) S' is one shape close to model shape S and produces the transformed shape S'_T. The pose estimation is actually applied to shapes S and S'_T instead of S' and S'_T.

here $\| \, . \, \|$ is the Euclidean norm.

As shown in Figure 4.2, shapes that are within ϵ (some small positive number) of the model shape S are indistinguishable from one another. The actual data set S'_T is never an exact transformed copy of the model shape S. It is a transformed version of one shape S' in the Hausdorff shape class of S. The small perturbation between shapes S and S' will cause small variations in the coefficients of the corresponding IP models. Hence, we need to know whether or not a small amount of change in coefficients can result in a dramatic change in the pose estimation result, i.e., whether or not the closed-form solution is stable within the neighborhood of the given coefficient sets.

Stability analysis has two important implications. For the data set to be processed (coefficients b_{ij}), it can tell us whether or not a relative pose estimation result of data set with regard to one of the database model is reliable. For the database model (coefficients a_{ij}), it can be used to test if the model is good enough for the possible data sets or if a few or a class of models is needed to provide sufficient database information.

In the Euclidean case, the rotation angle is the function of coefficients a_{ij} and b_{ij} of f_1 and f_2: $k = tan(\theta) = g(a_{ij}, b_{ij})$. We then can calculate all the first order derivatives of g wrt the coefficients, i.e.,

$$\frac{\partial \theta}{\partial b_{ij}} = \frac{1}{1+k^2} \frac{\partial g}{\partial b_{ij}} \tag{4.5}$$

There are altogether 10 such derivatives for 4th degree IPs. They measure the rate of

change in the estimation angle with respect to change in the coefficients. Some of the derivatives being higher than a given threshold would mean that the solution is not very reliable and that a better IP model is needed. Figure 4.3 shows two cases for the butterfly shape using conics in the pose estimation (section 4.4). The first row and the second row are based on the patch 15 and the patch 48 in Figure 4.6, respectively. One is a reliable rotation angle estimation (first row) and another is a poor estimation result (second row). In both cases, database IP model coefficients a_{ij} are fixed along with one of the leading term coefficients b_{ij} of the best fit IP for the data. k is plotted as the surface of two coefficients from (b_{20}, b_{11}, b_{02}), which vary within the neighborhood of b_{ij}. The derivatives (4.5) are very small for the reliable pose estimation, whereas for the poor estimation result, they are relatively big.

From the statistical pattern recognition point of view, the stability analysis method has an interesting interpretation in terms of the accuracy of the MAP (maximum a posteriori probability) estimation [27]. For Euclidean transformations, denote $P(\theta|data, model)$ as the probability of rotation θ given the raw data set and database model. In order to have an accurate estimation, P has to be a narrow function of θ, i.e., sharply peaked around the MAP estimator $\hat{\theta}$ or close to a Dirac delta function. If we assume the prior probability density function for θ is uniform over $[0°, 360°]$ (i.e., no prior knowledge), then equivalently, we would want $P(data|\theta, model)$ to be a narrow function of θ. In [5, 42], a Bayesian recognition framework is set up under the assumption that data points are actually generated according to the zero set of the best fit IP model f plus white noise. Under this assumption, we can write down $P(data|\theta, model)$ as:

$$P(data|\theta, model) = \frac{1}{Z} e^{-\sum_i dist^2(z_i, model(\theta))} = \frac{1}{Z} e^{-dist^2(data, model(\theta))} \tag{4.6}$$

here, $\frac{1}{Z}$ is a normalization factor and $model(\theta)$ is the IP representation for the database model after θ degree rotation. Hence, in order for $P(data|\theta, model)$ to be a narrow function of θ, $\frac{\partial dist(data, model(\theta))}{\partial \theta}$ should be big around the MAP estimator $\hat{\theta}$. By the chain rule,

$$\frac{\partial dist(data, model(\theta))}{\partial \theta} = \frac{\partial dist(data, model(\theta))}{\partial b_{ij}} \frac{\partial b_{ij}}{\partial \theta}$$

we want $\frac{\partial b_{ij}}{\partial \theta}$ to be big, or equivalently, $\frac{\partial \theta}{\partial b_{ij}}$ (as in (4.5)) to be small. This is exactly the condition we require for reliable pose estimation.

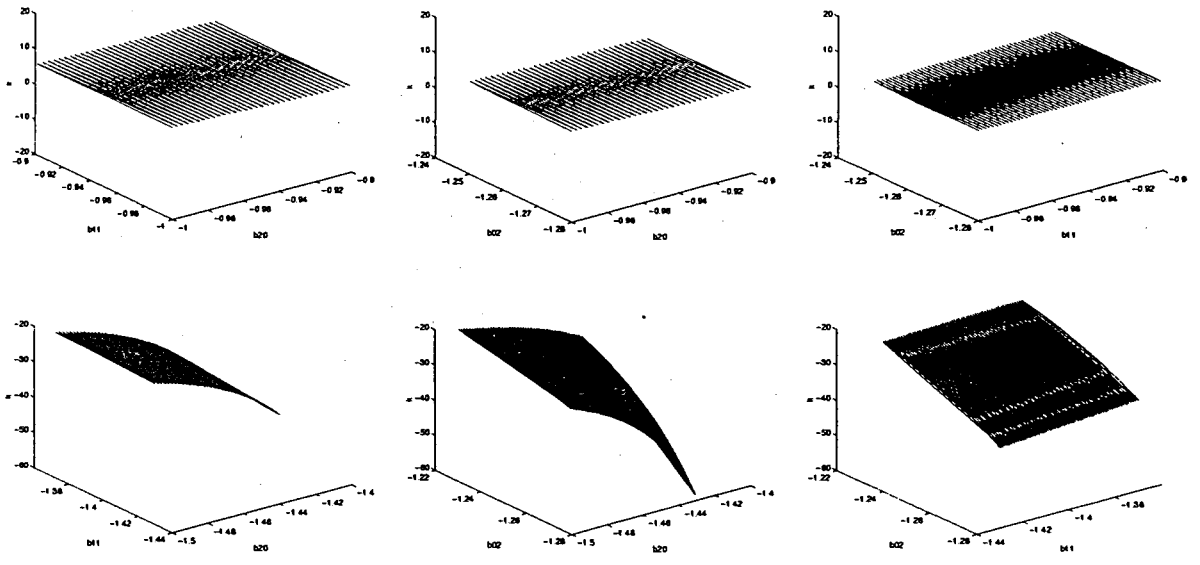


Figure 4.3: Stability analysis for the rotation estimation using conics for butterfly shape under 10% missing data. k is plotted as the surface of two coefficients from leading term coefficients (b_{20}, b_{11}, b_{02}). The first row and the second row are based on the patch 15 and the patch 48 in Figure 4.6, respectively.

4.4 Experiments

In this section, we demonstrate the method in the Euclidean case and give a comparative performance analysis of different degree IP pose estimators and the scatter matrix. Since, the underlying representation allows a separate treatment of the rotation angle and the translation, experiments for each will be shown independently. It should also be noted that the rotation and translation estimation for similarity transformations are essentially the same as for the Euclidean case.

Euclidean rotation under missing data

The Euclidean rotation experiments were run on 10 arbitrary shapes shown in Figure 4.4. The performance of the closed form pose estimations for the conic and fourth degree polynomial cases were compared with the scatter matrix method under 5% and 10% missing data of the object boundaries. The objects were rotated at a given angle and occluded, then the algorithms were run with the original objects, and their rotated and occluded versions to determine the rotation angle. For a systematic missing data analysis, the object boundaries were divided into 50 equal pieces and the boundary was

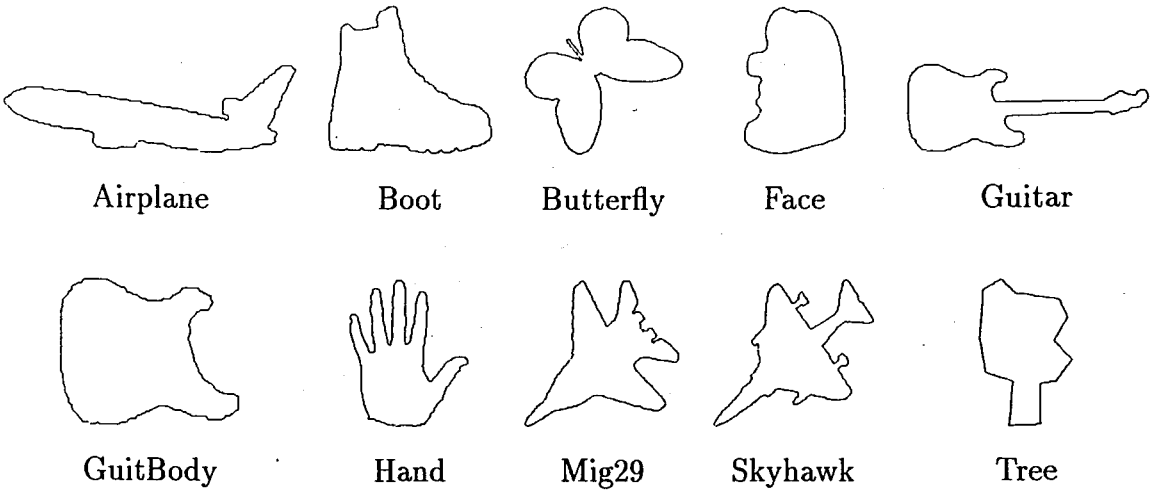


Figure 4.4: Object shapes used in the pose estimation experiments.

Figure 4.5: 5% and 10% occluded versions of rotated airplane and butterfly shapes.

traced clockwise each time starting from the beginning of the next piece and chopping off the required percentage of the whole data. Figure 4.5 show examples of 5% and 10% missing data.

Figures 4.6 show how the algorithms perform at each of the 50 points for the airplane and the butterfly with 5% and 10% missing data under an 80 degree rotation angle. The occasional spikes in the polynomial estimates indicate high variability in the coefficient space. These cases can be detected through the stability analysis discussed in the previous section. In fact, the example for the unstable pose estimation given in that section has the same IP representations as the highest peak of the above butterfly with 10% missing data.

Table 4.1 summarizes the 5% and 10% missing data performances of the algorithms for all objects. The true rotation angle is 80 degrees and the performance measure is the average of the absolute errors at the 50 starting points along the object boundary.

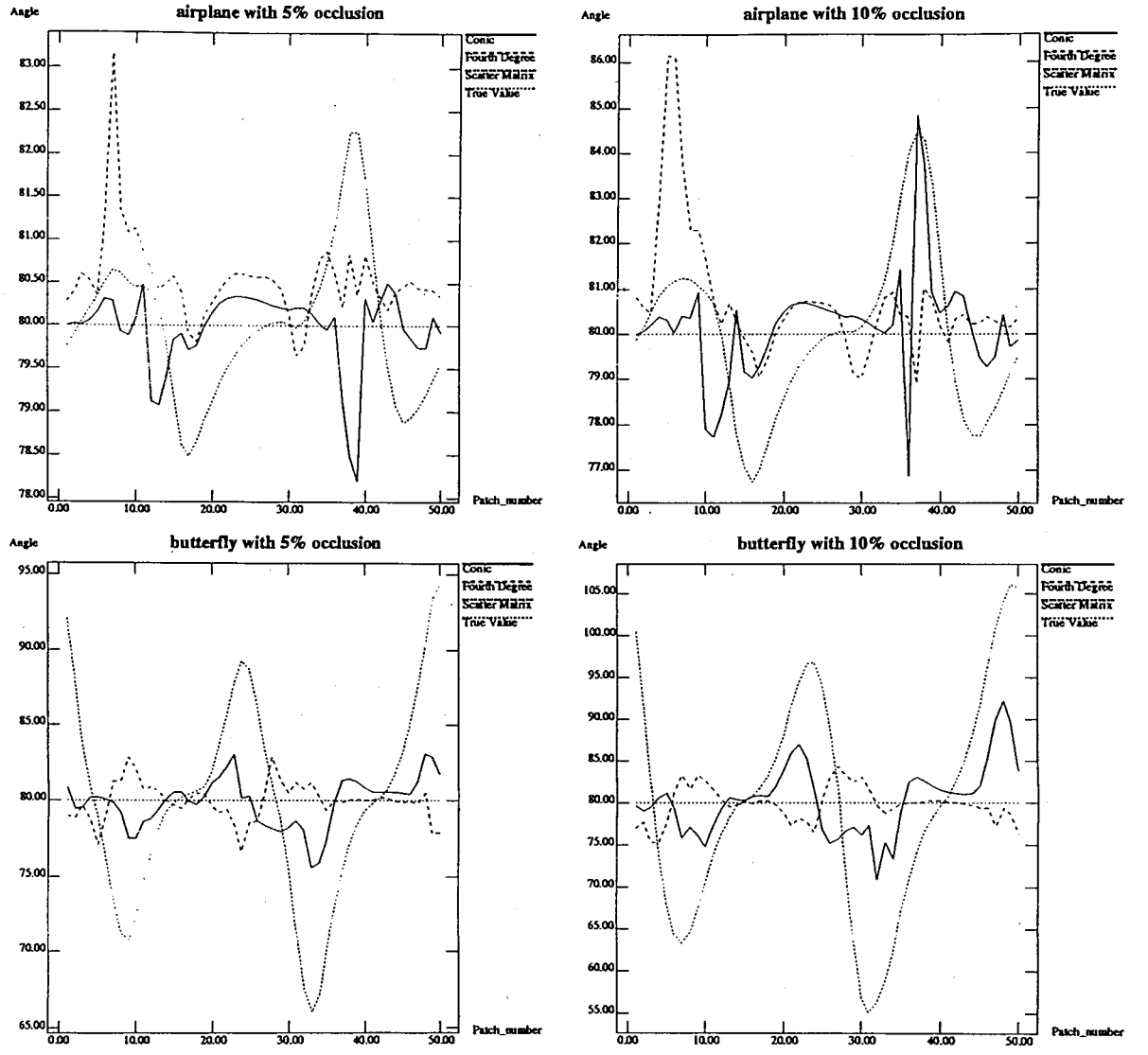


Figure 4.6: Pose estimation results for airplane and the butterfly with 5% and 10% missing data under 80 degree rotations.

Object	Conic	Fourth	Scatter
Airplane	0.301	0.574	0.639
Boot	1.800	1.062	2.921
Butterfly	1.217	0.872	5.177
Face	0.865	2.257	4.427
Guitar	0.830	2.223	0.706
Guitbody	1.251	0.937	15.866
Hand	2.212	2.845	20.122
Mig29	5.861	1.564	3.519
Skyhawk	3.118	1.346	1.353
Tree	0.954	0.951	19.966

(a)

Object	Conic	Fourth	Scatter
Airplane	0.787	0.939	1.380
Boot	4.440	2.510	6.453
Butterfly	3.129	1.492	10.933
Face	2.142	4.417	9.996
Guitar	1.645	6.779	1.499
Guitbody	3.157	2.159	13.181
Hand	4.024	6.957	20.725
Mig29	7.129	3.783	12.870
Skyhawk	5.743	2.382	2.823
Tree	1.891	2.824	19.895

(b)

Table 4.1: Average estimation errors (a) with 5% missing (b) with 10% missing

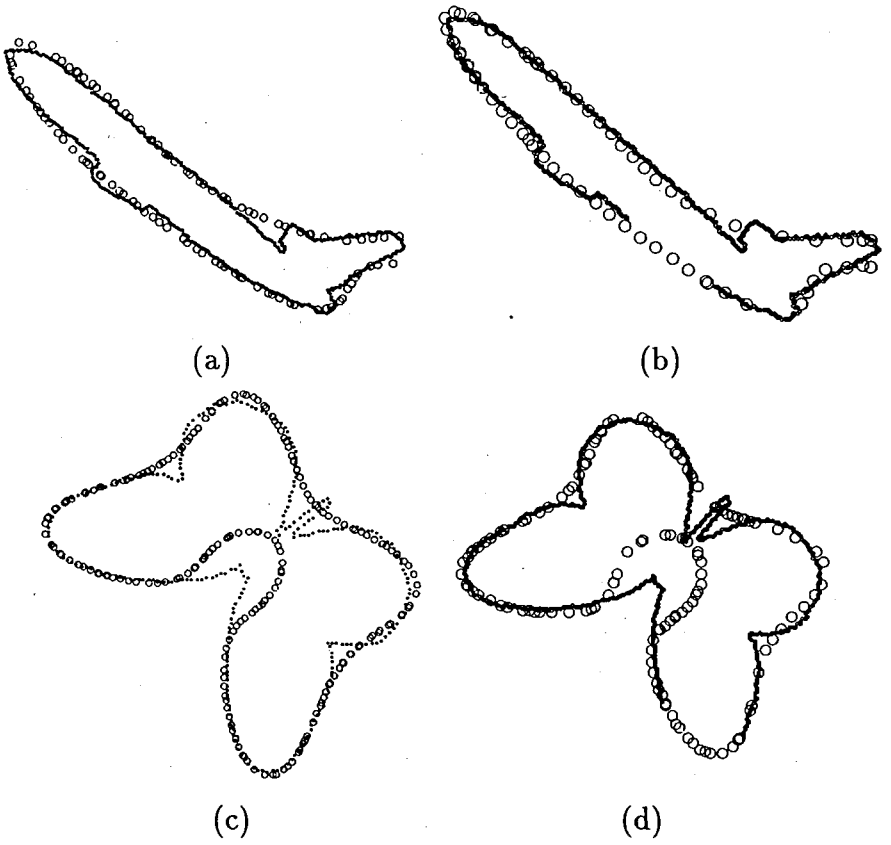


Figure 4.7: Interpolation property of IP representation. Small dots are airplane and butterfly data set with 10% missing data. Circles are the zero set of the best fit 4th degree IP models. (a) Poly Fit of the whole airplane shape; (b) Poly Fit of airplane with 10% missing data; (c) Poly Fit of the whole butterfly shape; (d) Poly Fit of butterfly with 10% missing data.

The experiments clearly show that the polynomial estimations outperform the scatter matrix estimation by compensating the lost data through interpolation. In general, the fourth degree polynomial based estimator performs better than the conic estimator due to better interpolation and representation power of fourth degree polynomials. The scatter matrix approach gives reasonable estimates for elongated objects- i.e., objects having two apparent eigenvector directions corresponding to minor and major axes - such as the airplane, and the guitar. In the other cases, i.e., blobby objects or objects with several principal directions, it significantly diverges from the true value. In fact, this anomaly is better understood by comparing the results for the guitar with and without the handle, the guitar and the guitbody respectively. Figure 4.7 shows some data sets and the fittings.

One characteristic of the polynomial method is that its sensitivity to missing data

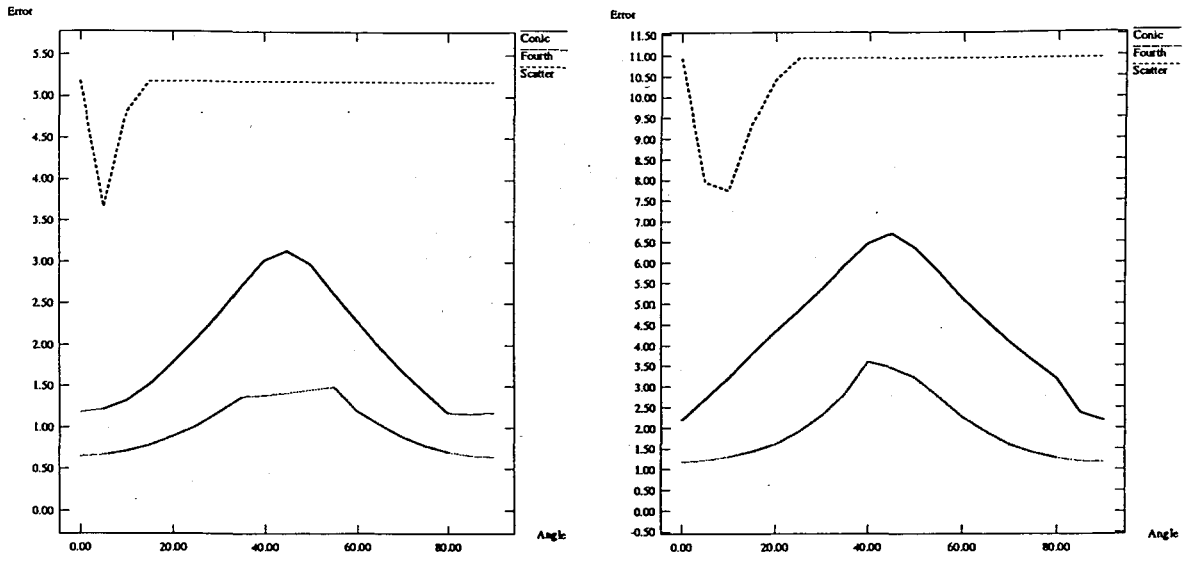


Figure 4.8: Average estimation errors for butterfly for all angles (a) 5% missing (b) 10% missing.

is not uniform with respect to the rotation angle. Our estimators give the best results for the values around 0 degree and 90 degree. Their worst performances are around 45 degrees. They repeat this pattern in every 90 degrees. Still their average performance is better than the scatter matrix. Figure 4.8 demonstrates the average absolute estimation errors for the butterfly for varying rotations.

Translation estimation under missing data

Figure 4.9 shows the comparison of translation estimation results for a hand shape using center of mass and center of conic methods (4.2). For the experiments, the objects were occluded, with the same technique as above, and no actual translation was added. The result is much better using the conic center under 5% missing data. The average error and standard deviation are 1.856 and 0.719 for mass center method and 0.999 and 0.725 for conic center method. It is very similar for 10% missing data except for one location where the fitted conic shape for the shape with the missing data can not match well with that of the original shape. This suggests that a higher degree IP model is more appropriate for the interpretation of missing data.

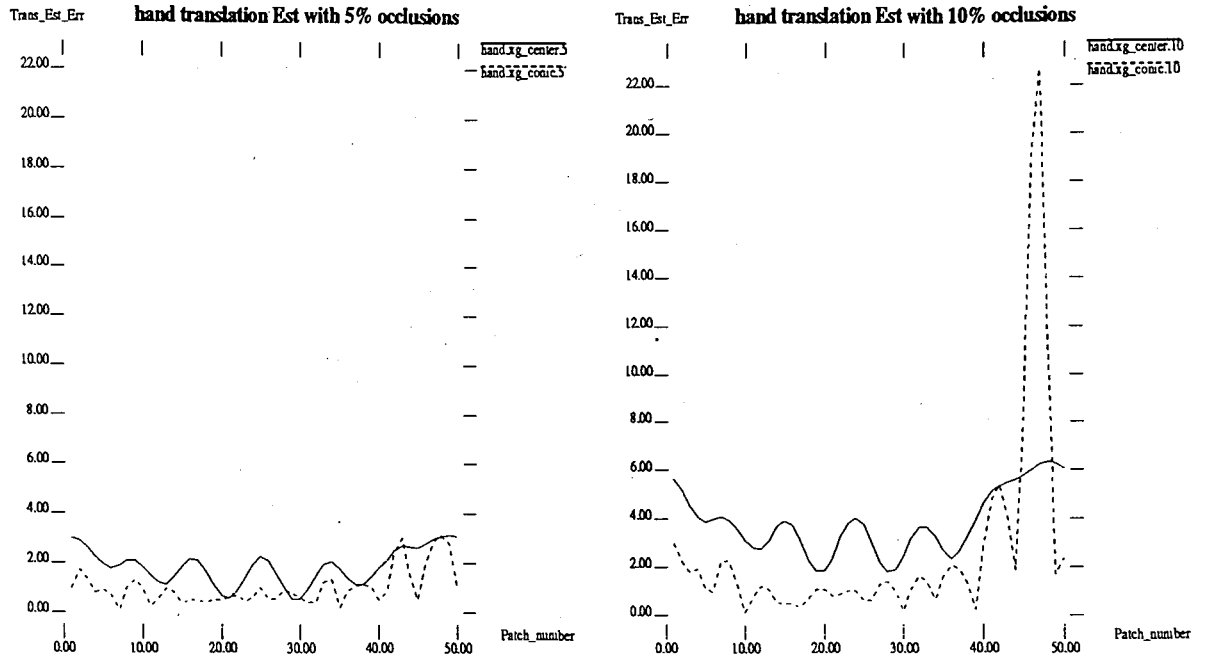


Figure 4.9: Translation Estimation for hand shape under 5% and 10% missing data.

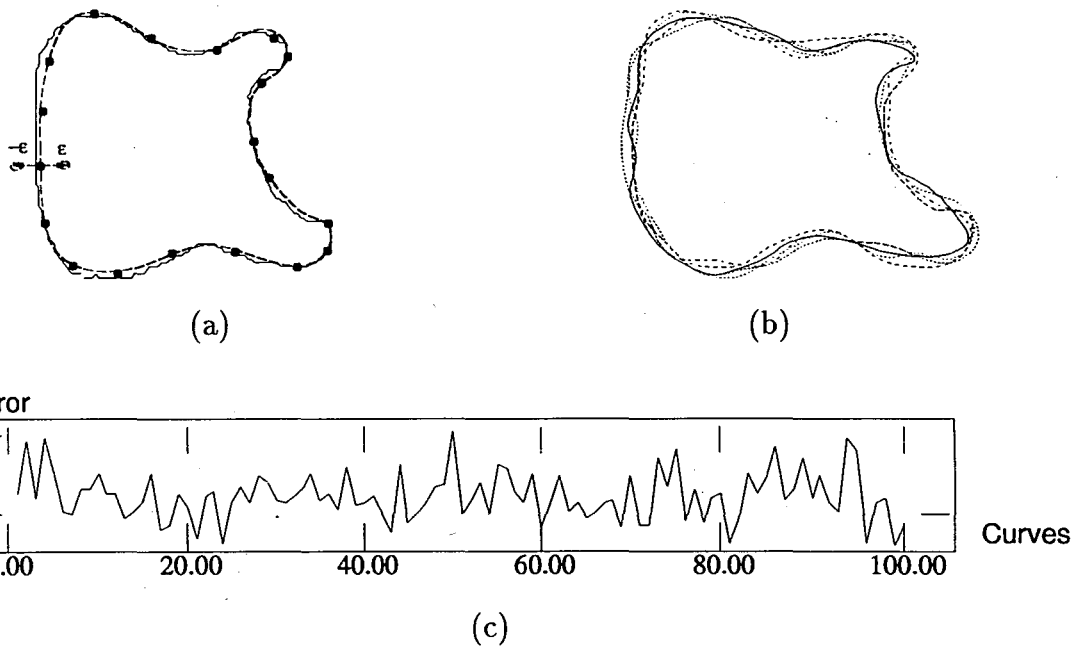


Figure 4.10: (a) Equally spaced control points (black dots) along the object shape (solid line) are used to generate B-spline curve (dotted line) for the experiment. (b) A few perturbed B-spline curves. (c) Pose estimation errors for 17.67 degree rotation of 100 perturbed curves of the standard guitar body shape.

Euclidean rotation under perturbation

Finally, we test the pose estimation for object under a small amount of perturbation. In Figure 4.10(a), a small subset of more or less equally spaced control points are

chosen from the original model shape to generate the B-spline curves approximating the shape. The control points are chosen such that the Hausdorff distance between the object shape and spline curve is less than a predefined threshold δ. We then moved these control points to generate random perturbations of the original curve. Each control point can assume randomly one of three states: $-\epsilon$, 0, and ϵ. ϵ should be chosen such that $\epsilon + \delta$ is within the stable region according to the stability analysis (Section 4.3). Here, ϵ is about 3 percent of object size and the total perturbation is about 6 percent of object size. Figure 4.10(b) shows five examples of such perturbations. We generated 100 perturbations and rotated each of them by 17.67 degrees. The pose estimation is then run between the rotated curve and the standard model shape. The estimation errors for 100 curves are shown in Figure 4.10(c). The mean error is 1.04707 degree, about 5.9 percent of the true angle. And the standard deviation is only 0.827283.

4.5 Assessment of Results

In this chapter we presented a novel model-based 2D-2D closed-form algebraic solution for coordinate transformation estimation for Euclidean, similarity, and non-uniform scale transformations. Besides its simplicity and other advantages discussed, experiments show that the proposed solution is relatively robust to missing data and perturbations. Under missing data the method outperforms the classical scatter matrix method. Also, it was demonstrated that the closed-form solution, being a function of the polynomial coefficients for the underlying models, is particularly appropriate for an analytical stability analysis.

Generalization of the method to affine transformation is an ongoing research. By Singular Value Decomposition a general affine transformation can be decomposed into a rotation, followed by a stretching and another rotation. The separation method needs two more intermediate coordinate systems to account for the three different factors. However, because the size of the simultaneous equations, the Maple system that we used failed to further simplify the system and come up with a full solution. One way to get around this is to use lower order terms to increase the number of simultaneous equations obtained from the coefficients. This requires a reliable way of handling the translation part first.

Although we only used the conic and the fourth degree polynomial estimators, an

extension to higher degree polynomials is straightforward. Another useful generalization along the same lines can be to determine the closed form estimator for 3D-3D pose problem which is inherently Euclidean.

Chapter 5

Free-form 3D Object Modeling by Implicit Algebraic Surfaces

5.1 Why Implicit Algebraic Surfaces?

The extraction of information from 3D data is still often difficult as it involves processing huge volumes of data. A generic model fitted to data sets can dramatically ease the required processing and simplify several 3D problems. In computer vision, objects in 3D images are mostly described by their *surfaces*. Ellipsoids, quadric surfaces, and super-quadrics are used as surface models. In essence, these surfaces are special forms of the more generic *implicit algebraic surfaces* [69, 30, 3]. In this section, a 3D object representation based on implicit algebraic surfaces is presented. Fundamental to the usefulness of this representation is fitting algebraic surfaces to 3D data sets. This problem will be solved by extending the 3L fitting algorithm for 2D curves to 3D. Our goal is to show that implicit algebraic surfaces become a viable 3D shape representation alternative with the new efficient 3D fitting technique. The 3D shape representation methodology developed here appears to be ideally suited to the following uses:

1. A single implicit polynomial can represent 3D surface data "well" if the surface shape is of moderate complexity, and it can provide a meaningful approximation, for many purposes including pose estimation, to a complex shape. For example, the 6th degree polynomial in Figure 5.4 provides a good representation for the given human heart data. Furthermore, essentially all the time required for this fitting is in the computation of 84 monomials for each data point. With efficient programs, this should

be less – perhaps considerably less – than a few seconds on a fast computer. Since each point is processed separately, this can also be done on using a signal processing chip. Implicit polynomial models are useful for visualization and for measuring shape geometry parameters of the represented object at a level of resolution appropriate to the representation (see Section 5.4 for examples).

2. These representations are useful for comparing two shapes. Since our implicit polynomial fitting approach is to fit an implicit polynomial $f(x,y,z)$ to the distance transform $d(x,y,z)$ of the data set Γ_0 in the vicinity of Γ_0, for any new data point $(\tilde{x},\tilde{y},\tilde{z})$ $d(\tilde{x},\tilde{y},\tilde{z})$ gives the approximate distance of $(\tilde{x},\tilde{y},\tilde{z})$ to Γ_0. Given two 3D data sets Γ_0 with K points and $\hat{\Gamma}_0$ with $\hat{K}$ points along with $f(x,y,z)$ and $\hat{f}(x,y,z)$, the respective polynomial fittings, the comparison of the shapes can be done in a number of ways, two of which are the following: (i) Compute the approximate sum of squared distances from the points in Γ_0 to the implicit polynomial representation $\hat{f}(x,y,z)$ for $\hat{\Gamma}_0$, or vice versa, i.e.,

$$\frac{1}{K}\sum_{(x,y,z)\in\Gamma_0}\hat{f}^2(x,y,z)$$

(ii) Compute the approximate sum of squared distances between smooth approximations f and $\hat{f}$ to Γ_0 and $\hat{\Gamma}_0$,

$$\frac{1}{K+\hat{K}}\sum_{(x,y,z)\in\Gamma_0\cup\hat{\Gamma}_0}[f(x,y,z)-\hat{f}(x,y,z)]^2$$

Note that this particular approximate distance measure, called PIMs (Polynomial Interpolated Measure), will be highly accurate even though corresponding coefficients of the two implicit polynomials may be very different. Both measures (i) and (ii) can be expressed as Mahalanobis distances in the polynomial coefficients (see [39] for details for the 2D case)

3. A complex object can be represented at arbitrarily fine resolution with low degree implicit polynomials by representing the data by overlapping polynomial patches, each covering only a portion of the object surface. Alignment of an entire object can be done in terms of the intrinsic coordinate systems for its patches.

4. Since the implicit polynomial representations are coordinate independent, the computational cost of tracking a data set which is moving and deforming is very low. Hence, implicit polynomial surfaces are an ideal representation for representing, tracking and registering a moving organ in medical data analysis [70].

5. For the preceding reasons, implicit polynomial patches are ideal for many applications such as object recognition, positioning, and metrological inspection for manufacturing automation, and indexing into 3D databases.

5.2 Implicit Algebraic Surfaces

An implicit algebraic surface is defined as the zero set of an implicit polynomial in 3 variables x, y, z. Using the formulation of Section 2.3 a 3D implicit polynomial surface of degree n is given by the following equation:

$$f(x,y,z) = \sum_{0\le i,j,k; i+j+k\le n} a_{ijk}x^i y^j z^k = \underbrace{a_{000}}_{H_0} + \underbrace{a_{100}x + a_{010}y + a_{001}z}_{H_1(x,y,z)}$$
$$+ \underbrace{a_{200}x^2 + a_{110}xy + a_{101}xz + a_{020}y^2 + a_{011}yz + a_{002}z^2}_{H_2(x,y,z)} + \ldots$$
$$+ \underbrace{a_{n00}x^n + \ldots a_{0n0}y^n + \ldots + a_{00n}z^n}_{H_n(x,y,z)} = \sum_{r=0}^{n} H_r(x,y,z) = 0, \qquad (5.1)$$

here $H_r(x,y,z)$ is a *homogeneous ternary polynomial (form)* of degree r in x, y, and z. After a variable ordering $z \prec y \prec x$, with graded lexicographic ordering, the surface equation can be rewritten as (T denotes matrix transpose)

$$f(x,y,z) = \mathbf{Y}^T\mathbf{a}, \qquad (5.2)$$

$$\mathbf{a} = [\, a_{000} \quad a_{100} \quad a_{010} \quad a_{001} \quad \ldots \quad a_{n00} \quad \ldots \quad a_{0n0} \quad a_{00n} \,]^T \qquad (5.3)$$

$$\mathbf{Y} = [\, 1 \quad x \quad y \quad z \quad \ldots \quad x^n \quad \ldots \quad y^n \quad \ldots \quad z^n \,]^T \qquad (5.4)$$

For a 3D object surface $\Gamma_0 = \{(x_i, y_i, z_i) | i = 1, ..., K\}$ an implicit representation is obtained by fitting an implicit polynomial surface f such that its zero set, $Z(f) = \{(x,y,z) | f(x,y,z) = 0\}$, is a good approximation to the object surface. For this purpose we will extend the 3L fitting algorithm to 3D. Provided that the analogous levels sets are generated properly, the theory of the 3L fitting extends naturally to 3D.

5.3 Level Set Generation in 3D

If the surface data is not sparse and closed, a 3D Euclidean Distance Transform can be applied to obtain the level sets. However this is not the case with most 3D data

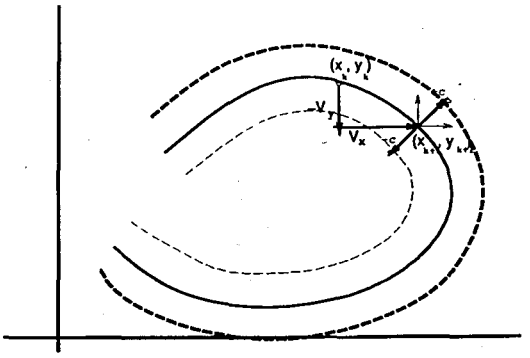


Figure 5.1: Level set geometry in 2D

sets. Moreover, computing the 3D Euclidean Distance Transform is computationally intensive. Other level set generation techniques which work in 2D usually fail in 3D. The principle dissimilarity between 2D and 3D lies in the fact that in 2D, the *edge chain*, a set of points forming the contour of an object, whether it is closed or not, has a natural ordering; it can be traversed in the clockwise or counterclockwise direction. Thus, the vector difference of adjacent points can be used for defining an absolute normal direction, i.e., one known to be pointing inside or outside the curve, for all points on the curve. Figure 5.1 illustrates how the two points in Γ_{-c} and Γ_{+c} can be obtained using the two unit component vectors $(V_x, -V_y)$ of the local tangent vector drawn from the two points on the curve Γ_0.

In 3D, there is no such natural ordering for 3D points data sets which may be helpful for deriving the sign of the unit normal vector. Our approach is to fit a plane to a set of points comprising the data point and the neighboring points within a given radius from the data point. In this case, the fitted plane is an approximation to the plane tangent to the surface in the vicinity of the data point and the plane normal can be used. However since the reference frame is not known, the plane normal does not give an absolute direction. To resolve this ambiguity an additional reference point known to be inside or outside the surface is needed. The ray $\mathbf{R}$ joining the reference point and the data point along with the tangent plane normal $\mathbf{N}$ gives the absolute orientation. More precisely, if $\mathbf{N} \cdot \mathbf{R} < \mathbf{0}$, vectors $\mathbf{N}$ and $\mathbf{R}$ point towards opposite sides of the tangent plane and towards the same side otherwise. In our experiments, we utilized generalized eigenvector fitting for fitting local tangent planes and used the plane normal $\mathbf{N}$ along with the center of mass as the additional reference point. Figure 5.2 shows the level sets for a head data set.

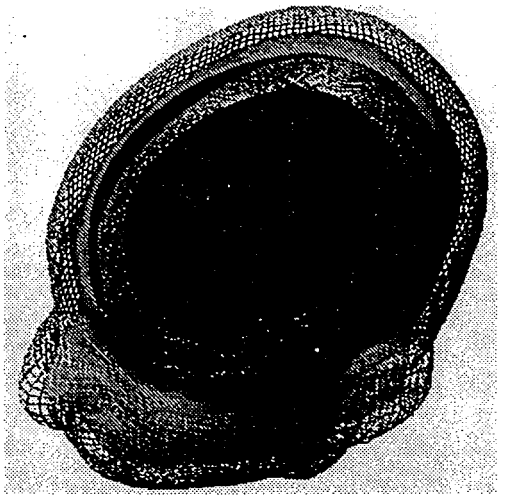


Figure 5.2: Level sets for head

5.3.1 Tangent Plane Fitting

Given a set of 3D data points

$$P = \{(x_1, y_1, z_1), \ldots, (x_n, y_n, z_n)\}$$

We want to find the plane $t(x, y, z) = F_1 + F_2x + F_3y + F_4z = 0$ passing through the points. Here $\mathbf{F} = (F_1, F_2, F_3, F_4)$ is a nonzero row vector of unknown plane parameters and $\mathbf{N} = (F_2, F_3, F_4)^T$ gives the plane normal. The geometric distance from a generic point $p_i = (x_i, y_i, z_i)^T$ to the plane is

$$d(p_i, \mathcal{Z}(t)) = \frac{\mid F_1 + F_2x_i + F_3y_i + F_4z_i \mid}{\parallel \mathbf{N} \parallel}$$

Minimization of mean squared distance (error)

$$E^2(F) = \overline{d}^2 = \frac{1}{n}\sum_{i=1}^{n} d^2(p_i, \mathcal{Z}(t)) = \frac{1}{n}\sum_{i=1}^{n} \frac{(F_1 + F_2x_i + F_3y_i + F_4z_i)^2}{F_2^2 + F_3^2 + F_4^2}$$

After some manipulation the problem becomes the well–known generalized eigen vector problem and the solution for $\mathbf{N}$ is given by the eigen vector corresponding to the minimum eigen value of the scatter matrix

$$\Sigma = \frac{1}{n}(X - \overline{X})^T(X - \overline{X})$$

where

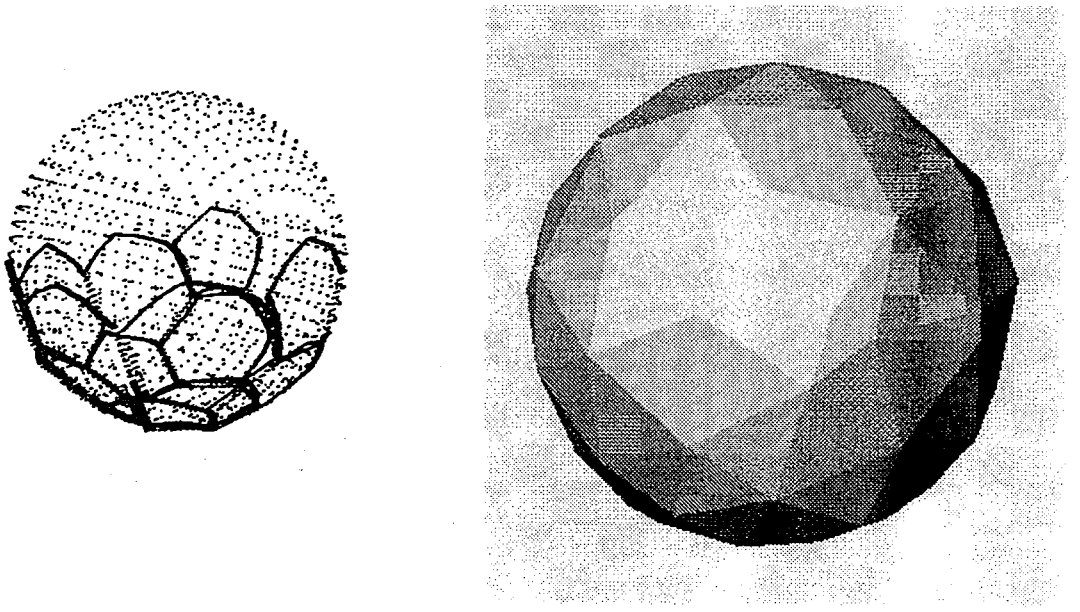
$$X - \overline{X} = \begin{bmatrix} x_1 - \overline{x} & y_1 - \overline{y} & z_1 - \overline{z} \\ \vdots & \vdots & \vdots \\ x_n - \overline{x} & y_n - \overline{y} & z_n - \overline{z} \end{bmatrix}$$

and $(\overline{x}, \overline{y}, \overline{z})$ is the center of the data set.

5.4 Example 3D Implicit Surface Models

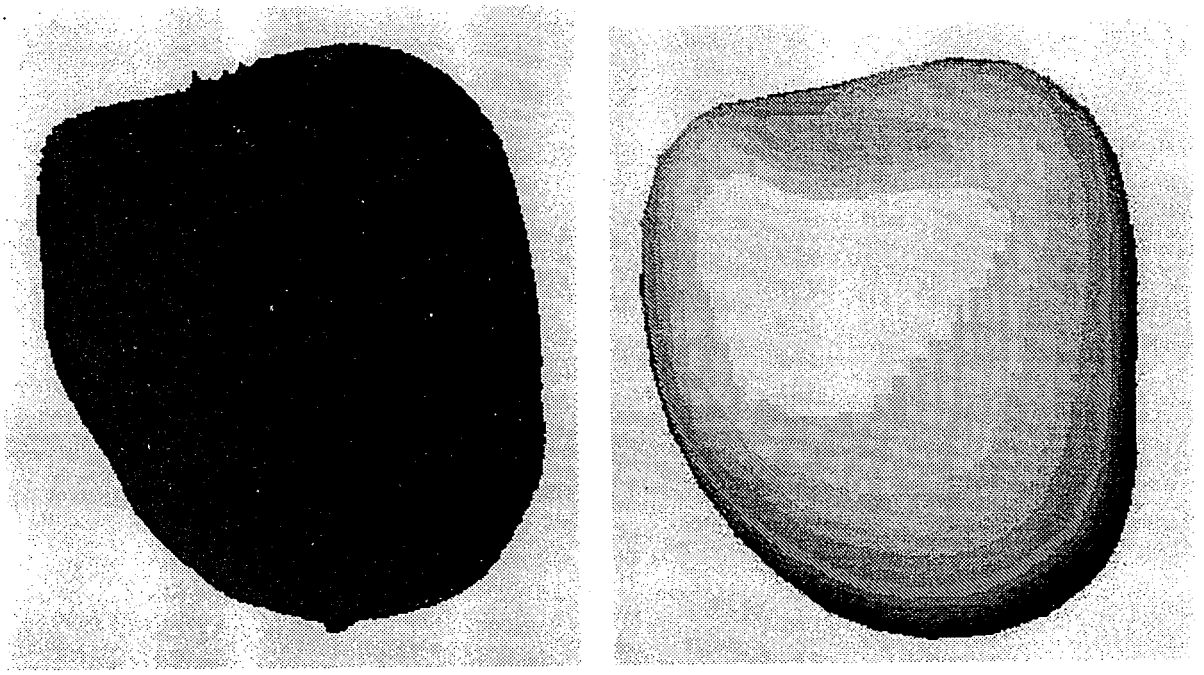
To illustrate the results of the proposed method, in Figures 5.3-5.6 a set of 3D objects and the 3D 3L implicit surface fits to them are given. The surfaces are ordered from relatively simple to more complicated. Notice that in order to capture the surface geometry well, we needed to increase the degree of the implicit surfaces as the data-set shapes get more complicated. Figure 5.3 shows a partial soccer ball (darker parts in (a) are the data set) and the $2nd$ degree implicit surface fit. Figure 5.4 displays the data set and the $6th$ degree fit for a heart (with 6480 data points). Figure 5.5 shows the data set and the $10th$ degree fit to a head (the data set has 12640 points). Finally, Figure 5.6 gives a data set (with 1250 points) and the $12th$ implicit surface fit for a face .

These figures illustrate the power of polynomials in representing objects which are either open or closed and with varying surface complexities. Besides the representation power, we observe some other pleasant properties of the implicit representation such as intrapolation (this is more apparent in the case of the ball) and smoothing (all shapes).



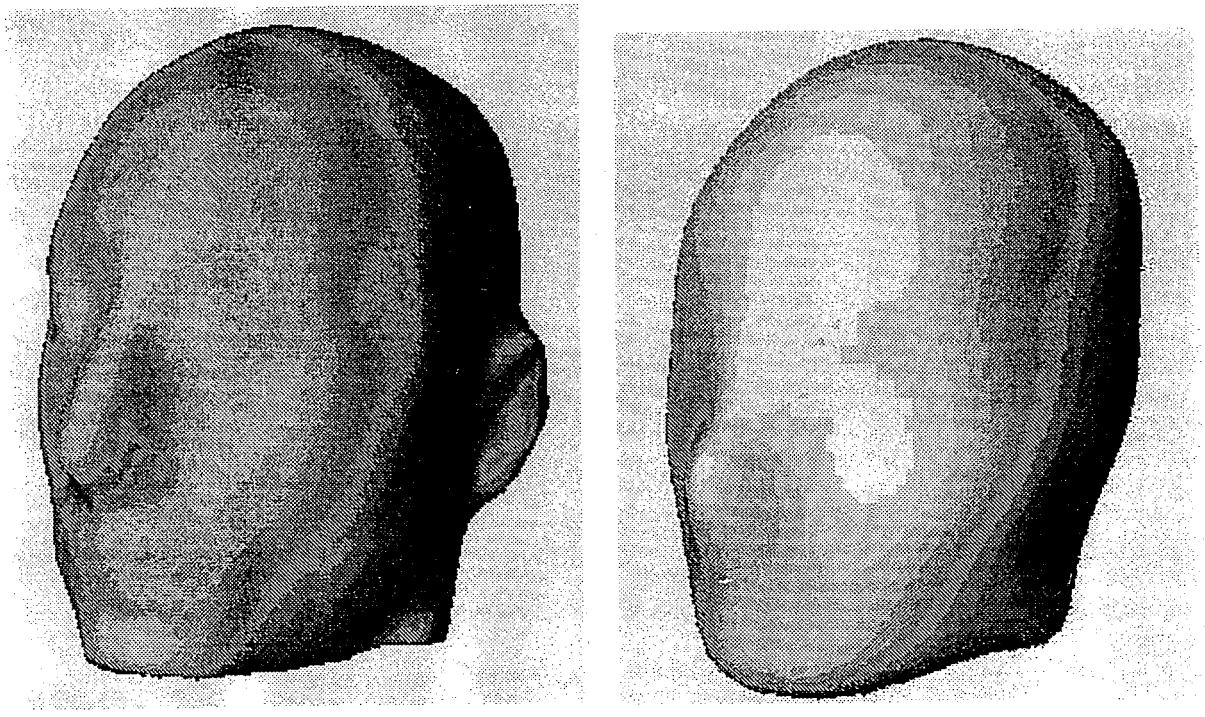
(a) (b)

Figure 5.3: (a) Ball data (darker part) with 6th degree fitting (lighter part) (b) 6th degree fit



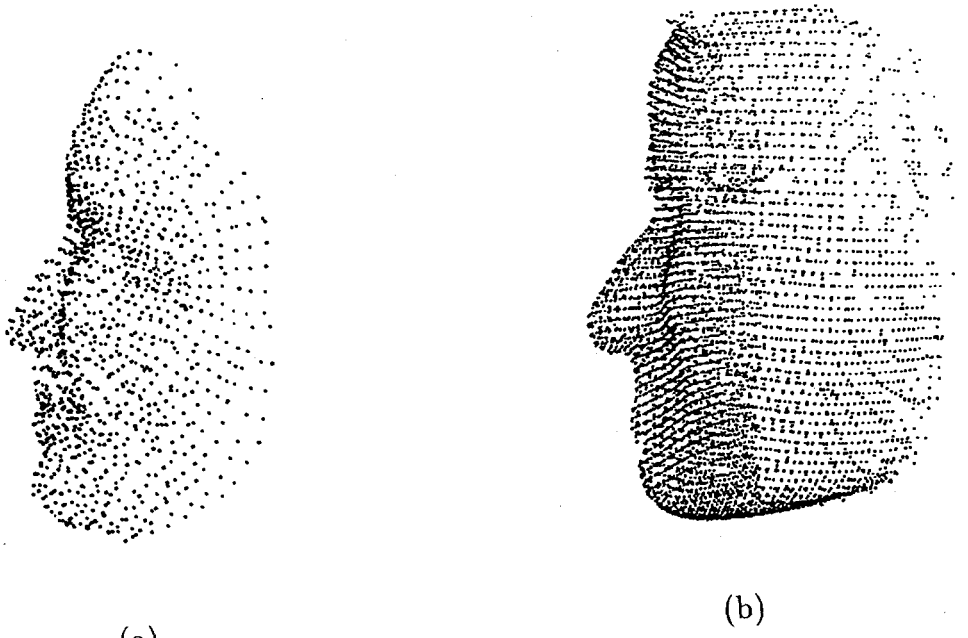
(a) (b)

Figure 5.4: (a) Heart (b) 6th degree fit



(a) (b)

Figure 5.5: (a) Head data set (b) 10th degree fit (ears discarded)



(a) (b)

Figure 5.6: (a) Face (b) 12th degree fit

Chapter 6

Affine Invariant 3L Fitting

6.1 Problem Definition

With the weak perspective camera model, 2D images of a 3D object are related to each other by an affine transformation [50, 51, 52]. More precisely, given a 3D object point, the coordinate transformation between its two 2D projections $\mathbf{X}$ and $\mathbf{X}'$ is given by

$$\mathbf{X}' = \mathbf{A}\mathbf{X} + \mathbf{T} \tag{6.1}$$

where $\mathbf{A}$ is a nonsingular 2x2 matrix and $\mathbf{T}$ is a 2x1 translation vector (see Section 2.2). Here, $\mathbf{A}$ and $\mathbf{T}$ contain a multiplicative factor which is inversely proportional to object-camera distance. This model is a good object-camera geometry approximation if the object is essentially planar or the object depth change is small relative to object camera distance (as in aerial images). Although it is a special case of the general perspective projection camera model, it is more general than the orthographic projection for which the image plane correspondence reduces to Equation 6.1 without aforementioned multiplicative factor.

Affine transformations have been extensively studied in computer vision and, as has been discussed in the previous parts of this thesis, within the implicit polynomial paradigm. If affine properties of implicit polynomials, such as affine invariants or pose estimators, are to be employed to determine the affine equivalance of two curves, a basic requirement is to have an *affine invariant fitting algorithm.* Given a data set and an affine transformation of it, an affine invariant fitting algorithm generates fittings to the original data set and the transformed one such that the resulting implicit polynomials are also related by the same affine transformation (see Section 4.2.2). This definition

has an important conceptual consequence in regard to what is meant by the quality of the fitting in the affine case. Visually an affine invariant fit may not match the performance of a Euclidean fit –psychologically we are very attached to the Euclidean space– but it is the mathematical consistency that we are after.

The 3L fitting algorithm is inherently Euclidean but not affine invariant. The reason for this is that the level set generation is based on a Euclidean invariant quantity, i.e., the distance of two points, which is not preserved under an affine transformation. As a result, after affine transformations of data sets the level sets generated by the Euclidean distance transform on the transformed data set are not identical to the same affine transformation of the corresponding level sets for the original data set.

In the context of 3L fitting there are two possible cures to this problem: removing the "affineness" of the data set by a scatter matrix normalization or replacing the distance in the level set generation by an affine invariant quantity.

6.2 Data Set Normalization

The scatter matrix (sample covariance matrix) $\mathbf{\Sigma}$ of a data set is a symmetric positive matrix. It can be rewritten as $\mathbf{\Sigma} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^{\mathbf{T}}$ where $\mathbf{Q}$ is an orthogonal matrix of normalized eigenvectors of $\mathbf{\Sigma}$ and $\mathbf{\Lambda}$ is the diagonal matrix of corresponding eigen values. If the transformation $\mathbf{A_w} = \mathbf{\Lambda}^{-1/2}\mathbf{Q}^{\mathbf{T}}$ is applied to the data points the scatter matrix of the resulting data set becomes the identity matrix $\mathbf{I}$. The application of transformation $\mathbf{A_w}$, sometimes called *whitening*, changes the dispersion of the data by making the spectrum of eigen vectors uniform.

If Γ_0 and $\hat{\Gamma}_0$ are two data sets related by an affine transformation, applying a whitening on both reduces the mathematical transformation between them to a rotation [37]. Therefore, after the whitening the "affineness" is removed and 3L fitting can be used without needing any modification.

An obvious shortcoming of the data set normalization is that subpixel pixel quantization, occlusion and other natural effects inherent in the image acquisition makes the scatter matrix unreliable. In addition, the assumption that the two data sets are pointwise in 1-1 correspondence is not realistic because of sampling and quantization in the formation of 2D images.

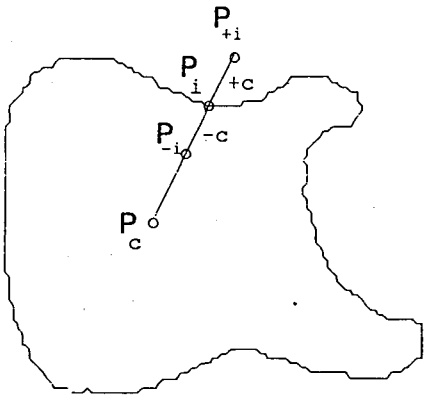


Figure 6.1: Line-segment ratio level set generation.

6.3 Affine Invariant Level Set Generation

As has been explained above, in the affine case the primary problem with the the 3L fitting is the use of Euclidean distance, a non-affine quantity, in generating level sets from a given data set. One remedy to this problem is to replace the distance by an *affine invariant quantity.* We propose two such quantities: *the line-segment ratio*, and *the affine curvature.*

6.3.1 Line-Segment Ratio

Under an affine transformation, lines become lines. Although the lengths of individual line-segments are not preserved, the ratio of the lengths of any two line-segments remains the same [49, 35]. This property can be used to formulate an affine invariant level set generation technique. As illustrated in Figure 6.1, from a given covariant point P_c, a line segment $\mid P_cP_i \mid$ (with length l) is drawn to a point P_i on the curve. Line-segments, such as $\mid P_iP_{-i} \mid$ and $\mid P_iP_{+i} \mid$ in the figure, along the same direction and starting at P_i with lengths c a constant multiple k of l (the length of $\mid P_cP_i \mid$) and $\mid P_cP_i \mid$ form a pair for which line-segment ratio is an affine invariant. Therefore for a given c, the end points of the two line-segments starting at P_i to either side of the curve give two points P_{+i} and P_{-i} on the two level sets. The level sets obtained this way will be affine invariant. The value k can be chosen as a certain constant ratio (such as 5%) of the contour point-covariant point distance l. Here the covariant point P_c can be the data mean or the intrinsic coordinate center [36] obtained by fitting a

preliminary implicit curve or another point with known coordinates before and after the affine transformation.

A drawback of this approach is its dependence on the robustness of the covariant point P_c. If, for instance the data mean is chosen to be the reference covariant point it is sensitive to arbitrary occlusions of the data set. Instead, an intrinsic coordinate center, obtained by fitting a preliminary implicit curve, can be used through an iterative approach. That is, first, a preliminary implicit curve is fitted to the data set and the resulting representation can be used to obtain the intrinsic object center which will then be used as the covariant point for computing an affine invariant fit.

6.3.2 Affine Curvature

Recently, the planar curve evolution, based on variants of the classical heat flow equation, was used for shape representation, smoothing and decomposition [12, 71, 57]. Fundamental to the curve evolution is invariant flows with respect to certain transformations. According to this approach, in affine invariant flows a point on the curve changes position in the direction normal to the curve and with a velocity proportional to the affine curvature at the point. Analogous to the Euclidean curvature κ, a Euclidean differential invariant, the affine curvature τ is an affine invariant differential quantity. It was shown that the affine curvature can be approximated by $\kappa^{1/3}$ where κ is the Euclidean curvature [57, 72].

Given $y = u(x)$, an explicit function representing the data curve $\Gamma_0 = \{(x_i, y_i) | i = 1, ..., K\}$, below are the curvatures for some transformation groups

Group	Curvature
Euclidean	$\frac{u_{xx}}{(1+u_x^2)^{3/2}}$
Similarity	$\frac{(1+u_x^2)u_{xxx}-3u_x u_{xx}^2}{u_{xx}^2}$
Special Affine	$\frac{P_1}{u_{xx}^{8/3}}$
Affine	$\frac{P_2}{P_1^{3/2}}$

where

$$P_1 = 3u_{xx}u_{xxxx} - 5u_{xxx}^2,$$

$$P_2 = 9u_{xx}^2 u_{xxxxx} - 45u_{xx}u_{xxx}u_{xxxx} + 40u_{xxx}^3.$$

A *special affine transformation* is an affine transformation for which the determinant

of the 2×2 matrix $\mathbf{A}$ (in Equation 6.1) is equal to unity. Note that since the determinant is unity, all invariants of the group of special affine transformations (without translations) are absolute weight invariants, as is the case for the group of rotations.

These ideas have a direct implication for affine-invariant level set generation. With this approach, for a given curve point P_i having affine curvature τ_i and normal $\vec{N}_i$, two points, each τ_i away from P_i along and opposite to the normal direction to the curve, form the corresponding inside and outside level set points. The level set generated so will be affine invariant. The affine curvature τ_i can also be used in other ways, as a constant in determining level value (the distance) at a point.

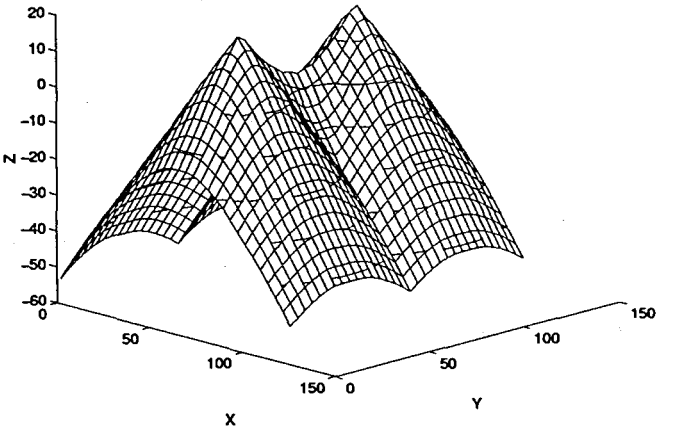
Determining the affine curvature, directly or as an approximation from the Euclidean curvature, is somewhat involved because noise, perturbations and occlusion make the local curvature computation, based on difference equations unstable. In medical imaging, this problem is circumvented by first computing a Euclidean distance transform of the curve in the vicinity of the curve, i.e., finding the distance of each pixel (within a given neighborhood) from the curve, and then using this information for the curvature estimation [73]. Figure 6.2) shows the Euclidean distance transform of a curve over the whole image. Analogous to the level set generation in the 3L fitting, the distance transform around the curve forms an explicit surface which makes the curvature estimation more stable. With the explicit surface $z = f(x, y)$, the Euclidean curvature is defined as

$$\kappa = -\frac{f_{xx}f_y^2 - 2f_{xy}f_xf_y + f_{yy}f_x^2}{(f_x^2 + f_y^2)^{3/2}}$$

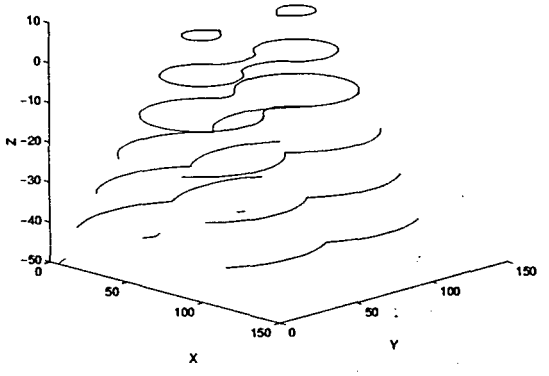
The main disadvantage of this technique is the computational cost of computing the distance transform. Another alternative is fitting an explicit or implicit polynomial locally or globally and using it to determinine the curvature at a given point (see Figure 6.3).

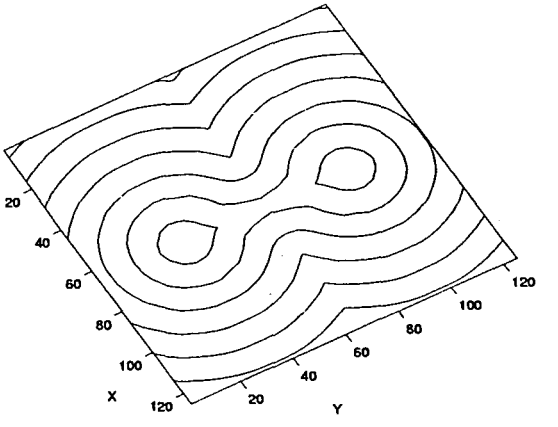
6.4 Experiments

In this section, we compare the performance of the Euclidean level set generation (distance transform) with the performance of our line-segment-ratio level set generation technique under affine transformations. For this purpose we used the four shapes, airplane, butterfly, car, and guitar in Figure 3.1. For each shape, first a $4th$ degree implicit polynomial is fitted to the given pixel level data set. Then, 100 random affine



(a) (b)





(c) (d)

Figure 6.2: (a) Peanut shape (b) Euclidean distance transform surface for the peanut (c) level sets in 3D (d) level set projections in 2D

transformations (as described in Section 3.3.3) are generated and the data set is transformed with these affine transformations. For each transformation, the resulting data set is quantizated from subpixel values to pixel values by rounding the x and y coordinates of the transformed points. This introduces a quantization noise. Then, a $4th$ degree implicit curve is fitted to the data set obtained in this way. Ideally, the polynomial obtained at this stage should be related to the polynomial representation for the original data set by the same affine transformation. That is, the two coefficient vectors should be the same up to the given affine transformation. We transform the coefficient vector, for the polynomial fit to transformed data, back to its normal position and try to determine the distribution for each coefficient (100 samples from the 100 affine transformations) by computing means and standard deviations. Theoretically the distributions should be an impulse with zero variance; therefore, smaller variance indicate better results. The experiment is run with each of the two level set generation techniques– which are applied prior to the implicit polynomial fittings– and their performances are compared. Tables 6.1, 6.2, 6.3, 6.4 give the actual values (the value for the original data set), means and standard deviations of the individual coefficients for the 4 shapes with the two level set generation techniques. Note that the corresponding coefficients may have different values (the same raws of the Coefficient columns in the tables (a) and (b) of each shape) with the two level set generations because, although the data set Γ_0 is the same Γ_{+c} and Γ_{-c} can be quite different with different level set generation techniques. The results indicate that the level set generation with the line-segment ratio technique gives much less coefficient variability.

6.5 Assessment of Results

In this chapter we discussed how to make the 3L fitting affine-invariant and proposed two theoretically sound techniques derived from two affine invariant quantities: a line-segment ratio based method, and a local affine curvature based method.

The line-segment ratio based level set generation technique is easily implementable. Yet the stability of the results depends on the robustness of the covariant point. The drawback of the affine curvature technique, however, is the difficulty of computing

Coefficient	Actual	Mean	Std
a00	-1.0202	-1.0623	0.3221
a10	2.8073	2.7016	0.3519
a01	43.8368	42.1145	2.2066
a20	-30.7494	-29.3235	5.6359
a11	51.4314	48.1990	9.2220
a02	202.1709	204.0876	19.5688
a30	5.0325	4.6849	1.0972
a21	-192.3056	-184.6405	19.6451
a12	-484.5704	-455.7016	87.2006
a03	602.6677	570.0975	281.3221
a40	61.2979	58.4404	10.0538
a31	65.9884	61.0090	20.6475
a22	21.6180	14.4137	43.4018
a13	-2912.5210	-2682.0589	853.7351
a04	6890.2700	6224.4336	1603.6172

(a)

Coefficient	Actual	Mean	Std
a00	0.0625	0.1509	0.0641
a10	2.1189	1.5735	0.4789
a01	31.2716	29.6464	1.0376
a20	-32.5868	-31.1479	2.0945
a11	49.5480	50.4607	3.7091
a02	88.1154	65.9655	17.2355
a30	6.2323	6.8007	0.5549
a21	-162.5114	-151.1471	10.1599
a12	-477.2975	-436.1843	39.6397
a03	692.4871	615.2459	118.4975
a40	61.4914	58.6781	3.6577
a31	84.3208	71.3743	10.2151
a22	258.6494	254.1725	20.5205
a13	-3441.6466	-3316.8648	265.9839
a04	7278.0686	7165.4870	404.7065

(b)

Table 6.1: Airplane coefficients with (a) Euclidean levels (b) Line-segment ratio levels

Coefficient	Actual	Mean	Std
a00	1.9657	1.8366	0.2291
a10	2.6635	2.6415	0.1784
a01	-3.6185	-3.5638	0.1678
a20	-57.1241	-54.3310	3.8971
a11	-37.9649	-37.0030	1.4945
a02	-53.5356	-50.8525	3.7621
a30	-11.1062	-10.8509	0.4387
a21	-63.1903	-59.7212	5.0104
a12	76.8760	72.7667	5.9752
a03	6.9941	6.9869	0.3864
a40	86.9147	82.5621	5.9340
a31	-67.7285	-62.2968	7.3122
a22	859.2228	817.3656	54.8020
a13	-181.5614	-170.7549	14.2208
a04	95.9684	91.0485	6.8019

(a)

Coefficient	Actual	Mean	Std
a00	1.5520	1.5687	0.0315
a10	2.4859	2.4841	0.0692
a01	-3.3377	-3.3156	0.0698
a20	-51.2094	-50.5541	0.4236
a11	-31.8558	-31.1091	0.5049
a02	-48.1472	-47.4265	0.4272
a30	-10.3952	-10.3736	0.1651
a21	-55.3715	-53.8364	0.7743
a12	67.4735	65.6529	0.9299
a03	6.7884	6.8560	0.1654
a40	79.0341	77.9663	0.6716
a31	-68.4994	-66.7971	1.4144
a22	777.2183	760.6989	7.0116
a13	-171.0681	-166.7621	2.2831
a04	87.7273	86.1514	0.8235

(b)

Table 6.2: Butterfly coefficients with (a) Euclidean levels (b) Line-segment ratio levels

Coefficient	Actual	Mean	Std
a00	-5.5128	-5.8014	1.2880
a10	18.1777	17.4457	1.4867
a01	-22.7970	-21.1667	3.5477
a20	-25.8031	-24.6004	4.3443
a11	84.1747	80.4165	6.5020
a02	-14.3962	4.4115	50.8725
a30	-62.3398	-59.7994	4.9816
a21	109.0076	107.2934	8.3874
a12	-7.0742	-2.7964	16.5719
a03	86.8660	70.0386	48.6400
a40	141.5323	139.4419	4.8500
a31	-186.6551	-172.0760	31.1121
a22	156.9160	138.4258	66.8092
a13	-620.1910	-595.9759	56.6005
a04	1552.8886	1362.3485	403.5763

(a)

Coefficient	Actual	Mean	Std
a00	-0.7802	-0.8042	0.2693
a10	13.6394	13.4995	0.6549
a01	-22.9654	-22.6379	0.9897
a20	-46.9679	-46.6277	1.5643
a11	60.1554	59.4203	2.9633
a02	-160.1501	-155.8525	10.0820
a30	-48.2173	-47.7017	2.0703
a21	100.2885	99.4533	3.2125
a12	40.8230	40.5430	8.8895
a03	139.4030	137.3470	14.1310
a40	160.3745	159.2825	3.0232
a31	-121.4214	-119.1885	9.1137
a22	574.4982	569.4466	33.2432
a13	-328.5251	-324.4590	39.8290
a04	2416.9708	2360.8191	82.2481

(b)

Table 6.3: Car coefficients with (a) Euclidean levels (b) Line-segment ratio levels

Coefficient	Actual	Mean	Std
a00	0.0207	-0.0756	0.1719
a10	0.9032	1.5347	0.8619
a01	7.7153	7.9665	1.1478
a20	-8.9326	-9.6977	1.1531
a11	17.1512	17.1249	2.2877
a02	-37.5726	-26.2214	16.8211
a30	-3.8141	-4.9539	1.5595
a21	-58.5733	-57.0042	6.3250
a12	586.5377	548.5959	71.6219
a03	27.4501	17.7117	22.5097
a40	15.7200	17.1750	2.0863
a31	-95.5385	-92.3136	10.6791
a22	703.6581	661.4404	87.4427
a13	38.2230	24.7458	49.9338
a04	2651.8365	2403.4048	327.9536

(a)

Coefficient	Actual	Mean	Std
a00	-0.0031	0.0128	0.0361
a10	7.6699	7.9158	0.2904
a01	4.9486	5.1154	0.4343
a20	-17.3045	-17.8289	0.6038
a11	0.4567	-0.9357	1.8934
a02	-48.2664	-38.4301	4.8841
a30	-15.3223	-15.7748	0.5945
a21	-12.0135	-9.7401	1.8880
a12	-86.5742	-111.1605	17.8639
a03	-68.1881	-77.6075	12.7768
a40	34.3023	34.9465	0.9959
a31	-3.5308	1.3897	3.6789
a22	-109.8835	-133.0542	21.7313
a13	-34.6383	-26.5129	21.9551
a04	1601.0894	1350.2371	111.4659

(b)

Table 6.4: Guitar coefficients with (a) Euclidean levels (b) Line-segment ratio levels

a stable affine curvature, directly or through the Euclidean curvature approximation which is only approximately affine invariant. As mentioned above, the curvature estimation through distance transform formation is computationally costly. If, instead, local explicit or implicit polynomials are used to determine the curvature, the curvature may not be smooth (a desirable property for the level sets) especially when higher degree polynomials are used. Note that higher degree approximations capture the shape information, thus the curvature information better. Figure 6.3 illustrates this situation on the butterfly shape where the approximate affine curvature along the curve (x axis) is computed by fitting $2nd, 4th, 8th, 12th$ degree implicit polynomials to the whole shape.

Nevertheless, the experimental results indicate that one of the proposed techniques improves considerably on using the Euclidean distance transform for the level set generation and the other technique looks promising. More thorough experiments are needed to be run, including a test of the affine curvature based level set generation technique, and testing the performance of both techniques under occlusion.

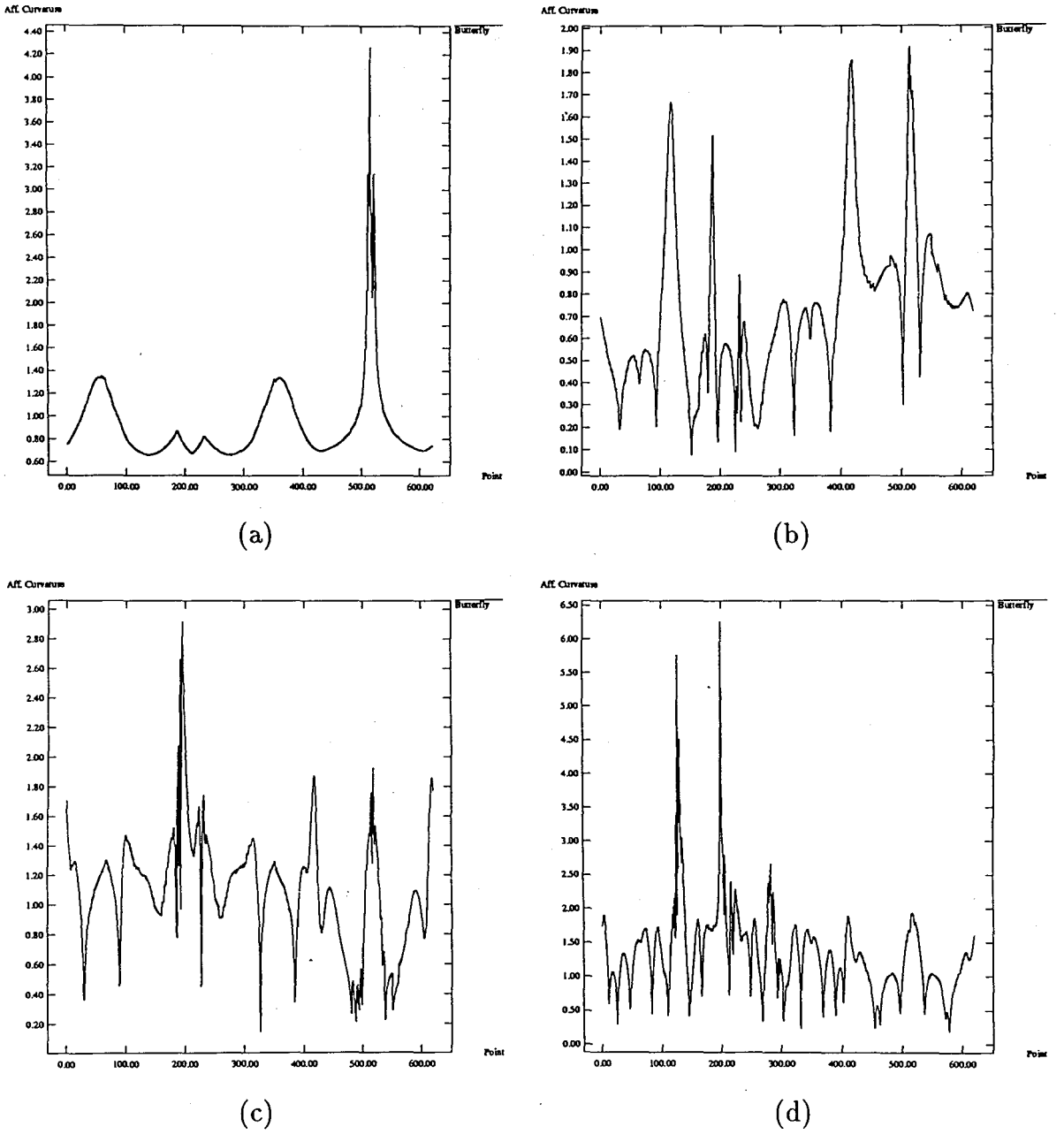


Figure 6.3: Affine curvature estimations with global implicit polynomial fits the points (x axis) along the curve and the curvature estimates at each point. (a) $2nd$ degree (b) $4th$ degree (c) $8th$ degree (d) $12th$ degree

Chapter 7

Conclusion

In this thesis, we discussed our four contributions having as common theme the exploration of different aspects of and problems in using implicit algebraic curves and surfaces for solving some fundamental shape-related computer vision problems, particularly object recognition and object alignment. Because of the structure of the thesis, conclusions regarding contributions on individual topics, and possible extensions and improvements on the presented techniques, have been given in the related chapters. In this final part, we present a more philosophical view of the prospects for implicit shape representation, and we mention implications of our own work on solving some of the new challenging technological problems where shape (object) representation plays a pivotal role.

2D and 3D shape understanding has become tremendously important with the emergence of new cutting edge technologies such as pictorial and video database use and management, 3D reconstruction for virtual reality, and medical imaging, in addition to its well–established status in relatively older fields such as robot vision and industrial inspection. Lately, the challenge for researchers on implicit polynomials has been to change its image as an elegant but mostly impractical approach and extend its application areas in the directions required by the new technologies. As we mentioned, some of the previously acute problems of implicit polynomials, e.g., the lack of fast, robust, repeatable implicit curve and surface fitting, have recently been satisfactorily solved and our work has contributed in this regards. Our research, too, largely supports the view that implicit polynomials are one of the leading models for shape representation.

However, the shape is still the most elusive ingredient in image understanding as it

requires a clear object background separation and is subject to deformations. Deeper philosophical repercussions– which engineers usually tend to ignore– of the concept of shape, or, more generally the notion of class, and shape classification, point to even more formidable difficulties: Mundy [74] interprets Wittgenstein as showing that "the idea of basing classes on common attributes cannot account for many of the linguistic categories which are in common use" [75] and Wantanabe as stating that "similarity of attributes between two objects is not a valid basis for classification!" [76]. A technical problem with the algebraic models is that modeling image plane correspondences by algebraic transformations, such as affine transformations, is just an approximation that may fail in some cases, because many 3D objects seen in 2D images are not planar, they are free-form and contain volume. These are possibly the main obstacles before implicit representation in its quest to become a widely embraced technique.

What is promising? Our research points to the following broad research directions and application areas for algebraic implicit curve and surface representation. Some of these became reasonably solvable as a result of the work presented in this thesis and other recent works of our colleagues at the Brown University LEMS Laboratory and they also form the main body of research currently carried out there.

1. In pictorial databases, the pose estimation and algebraic invariants can be used for indexing 2D shapes and recognizing, or at least limiting the number of possible 2D shape candidates in the database.

2. In 3D, algebraic implicit surfaces can be used in 3D reconstruction from video, and the resulting shape representation can be used as an index into video databases.

3. Implicit polynomials, by their intrinsic coordinate systems can serve for 3D object alignment which may have potential applications in representing, tracking and registering a moving organ in medical data analysis.

4. In computer graphics, generation of 3D meshes from unorganized point data sets can be accomplished by first fitting implicit polynomials to patches of points and fusing this initial representations into a single surface from which a 3D mesh can be generated.

5. Implicit representation may serve as a basis for industrial design, reverse engineering, and inspection of free-form objects.

Bibliography

[1] C. L. Bajaj and I. Ihm. "Smoothing polyhedra using implicit algebraic splines,". In *SIGGRAPH*, July 1992.

[2] H. Civi, C. Christhopher, and A. Ercil. "The classical theory of invariants and object recognition using algebraic curve and surfaces,". Technical Report FBE-IE-06/97-07, Bogazici University, July 1997. Under review for *Int. Journal of Computer Vision.*

[3] D. Keren, D.B. Cooper, and J. Subrahmonia. "Describing complicated objects by implicit polynomials,". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 38–53, January 1994.

[4] Z. Lei and H. Civi. "Closed-form object pose estimation using algebraic shape representation,". Technical Report LEMS-161, Brown University LEMS Lab., March 1997. Under review for *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

[5] J. Subrahmonia, D.B. Cooper, and D. Keren. "Practical Reliable Bayesian Recognition of 2D and 3D Objects Using Implicit Polynomials and Algebraic Invariants,". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 505–519, May 1996.

[6] 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*, November 1991.

[7] M. M. Blane, Z. Lei, D. B. Cooper, and H. Civi. "The 3L algorithm for fitting implicit polynomial curves and surfaces to data,". Technical Report LEMS TR-160, Brown University LEMS Lab., 1997. Under review for *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

[8] 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, FL, December 1996.

[9] Z. Lei, H. Civi, and D.B. Cooper. "Free-form object modeling and inspection,". In *Proceedings, Automated Optical Inspection for Industry SPIE's Photonics China '96*, Beijing, China, November 1996.

[10] J. Tarel, H. Civi, and D.B. Cooper. "Pose estimation of free-form 3D objects without point matching using algebraic surface models,". Technical Report LEMS-168, Brown University LEMS Lab., September 1997. Under review for IEEE International Workshop on Model-Based 3D Image Analysis.

[11] M. Hebert, J. Ponce, T. Boult, and A. Gross, eds. *Object representation in computer vision.* Springer Lecture Notes in Computer Science Series. Springer-Verlag, 1995.

[12] B.B. Kimia, A.R. Tannenbaum, and S.W. Zucker. "Shapes, shocks, and deformations, I: The components of shape and the reaction-diffusion space,". *Int. Journal of Computer Vision*, August 1995.

[13] H. Rom and G. Medioni. "Hierarchical Decomposition and Axial Shape Description.,". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):973–981, 1993.

[14] D. Cox, J. Little, and D. O'Shea. *Ideals, varieties, and algorithms.* Springer-Verlag, 1996.

[15] I.R. Shafarevich. *Basic algebraic geometry.* Springer-Verlag, 1977.

[16] G. Farin. *Curves and surfaces for CAGD: a practical guide.* Academic Press, third edition, 1993.

[17] C.M. Hoffmann. "Implicit curves and surfaces in CAGD,". *IEEE Computer Graphics and Applications*, Jan 1993.

[18] C.M. Hoffmann. *Geometric and solid modeling.* Morgan Kaufmann Publishers, 1989.

[19] T. W. Sederberg and D. C. Anderson. "Implicit Representation of Parametric Curves and Surfaces,". *Computer Vision, Graphics and Image Processing*, pages 72–84, 1984.

[20] M. Hall and J. Warren. "Adaptive polygonalization of implicitly defined surfaces,". *Computer Graphics and Applications*, pages 33–42, 1990.

[21] A. Hilton, A.J. Stoddart, J. Illingworth, and T. Windeatt. "Marching triangles:range image fusion for complex object modelling,". In *Proceedings ICIP*, 1996.

[22] W. Schroeder, K. Martin, and B. Lorensen. *The visualization toolkit: an object-oriented approach to 3D graphics.* Prentice Hall, 1996.

[23] G. Taubin. "An accurate algorithm for rasterizing algebraic curves,". *IEEE Computer Graphics and Applications*, 14(2):14–23, March 1994.

[24] D. Forsyth, J. L. Mundy, A. Zisserman, and C. M. Brown. "Projectively invariant representation using implicit algebraic curves,". In *Proceedings, First European Conference on Computer Vision*, 1990.

[25] D. Keren. "Using symbolic computation to find algebraic invariants,". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(11):1143–1149, November 1994.

[26] S. Petitjean. "Algebraic geometry and object representation in computer vision,". In M. Hebert, J. Ponce, T. Boult, and A. Gross, eds., *Object Representation in Computer Vision*, pages 155–165. Springer-Verlag, 1995.

[27] R.O. Duda and P.E. Hart. *Pattern classification and scene analysis.* Wiley, 1973.

[28] V. Pratt. "Direct Least Squares Fitting of Algebraic Surfaces,". *Computer Graphics*, July 1987.

[29] D.B. Cooper and N. Yalabik. "On the computational cost of approximating and recognizing noise-perturbed straight lines and quadratic arcs in the plane,". *IEEE Transactions Computer*, pages 1020–1032, October 1976.

[30] G. Taubin, F. Cukierman, S. Sullivan, J. Ponce, and D.J. Kriegman. "Parameterized families of polynomials for bounded algebraic curve and surface fitting,". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, March 1994.

[31] M. Pilu, A. Fitzgibbon, and R. Fisher. "Ellipse-specific direct least-square fitting,". In *Proceedings of International Conference on Image Processing*, Lausanne,Switzerland, September 1996.

[32] Z. Lei and D.B. Cooper. "Linear programming fitting of implicit polynomials,". *Under review for IEEE Transactions on Pattern Analysis and Machine Intelligence*, May 1996.

[33] Z. Lei and D.B. Cooper. "New, faster, more controlled fitting of implicit polynomial 2D curves and 3D surfaces to data,". In *Proceedings of Computer Vision and Pattern Recognition Conference*, San Francisco, CA, June 1996.

[34] J. L. Mundy, A. Zisserman, and D. Forsyth, eds. *Applications of invariance in computer vision: second joint European-US workshop, Ponta Delgada, Azores, Portugal, October 9–14, 1993: proceedings*, volume 825, New York, NY, USA, 1994. Springer-Verlag Inc.

[35] A. Zisserman J.L. Mundy, ed. . *Geometric invariance in machine vision.* MIT Press, 1992.

[36] G. Taubin and D.B. Cooper. "2D and 3D object recognition and positioning with algebraic invariants and covariants,". In B.R. Donald, D. Kapur, and J.L. Mundy, eds., *Symbolic and Numerical Computation for Artificial Intelligence.* Academic Press, NYC, 1992.

[37] G. Taubin and D.B. Cooper. "Object recognition based on moment (or algebraic) invariants,". In J. Mundy and A. Zisserman, eds., *Geometric Invariance in Computer Vision*, pages 375–397. MIT Press, Cambridge, Massachusetts, 1992.

[38] I. Weiss. "Geometric invariants and object recognition,". *Int. Journal of Computer Vision*, 10(3):207–231, 1993.

[39] Z. Lei, T. Tasdizen, and D. B. Cooper. "PIMs and invariant parts for shape recognition,". Technical Report LEMS-163, Brown University LEMS Lab., April 1997. accepted for Sixth International Conference on Computer Vision (ICCV), Bombay, India, January 4-7, 1998.

[40] M. Barzohar, D. Keren, and D. B. Cooper. "Recognizing groups of curves based on new affine mutual geometric Invariants, with applications to Recognizing Intersecting Roads in aerial Images,". In *12th International Conference on Pattern Recognition*, October 1994.

[41] D. B. Cooper and Z. Lei. "On representation and invariant recognition of complex objects based on patches and parts,". In M. Hebert, J. Ponce, T. Boult, and A. Gross, eds., *Object Representation in Computer Vision*, pages 139–153. Springer Lecture Notes in Computer Science Series # 994, 1995.

[42] Z. Lei, D. Keren, and D.B. Cooper. "Computationally fast Bayesian recognition of complex objects based on mutual algebraic invariants,". In *Proceedings of IEEE International Conference on Image Processing*, San Francisco, CA, June 1995.

[43] K. Siddiqi, J. Subrahmonia, D.B. Cooper, and B.B. Kimia. "Part-based Bayesian recognition using implicit polynomial invariants,". In *Proceedings of Int. Conference on Image Processing*, pages 360–363, October 1995.

[44] R. M . Bolle, D. B. Cooper, Y. P. Hung, and P. N. Belhumeur. "On optimally combining pieces of information, with applications to estimating 3D complex-object position from range data,". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(5):1028–1052, 1989.

[45] R.M. Haralick, H. Joo, C. Lee, X. Zhuang, V.G. Vaidya, and M.B. Kim. "Pose estimation from corresponding point data,". *IEEE Transactions on Pattern analysis and Machine Intelligence*, 19:1426–1446, 1989.

[46] J. Ponce, D.J. Kriegman, S. Petitjean, S. Sullivan, G. Taubin, and B. Vijayakumar. "Representations and algorithms for 3D curved object recognition,". In A.K.Jain and P.J. Flynn, eds., *Three-dimensional object recognition systems.* Elsevier Science Publishers, 1993.

[47] M. Unel and W. A. Wolovich. "A unique decomposition of implicit polynomial curves,". Technical Report LEMS-166, Brown Univeristy LEMS Lab., July 1997.

[48] W. A. Wolovich and M. Unel. "The determination of implicit polynomial canonical curves using quartic factorizations,". Technical Report LEMS-162, Brown University LEMS Lab., March 1997. Under review for *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

[49] G.B. Gurevich. *Foundations of the theory of algebraic invariants.* P. Noordhoff, 1964.

[50] O. Faugeras. "Stratification of three-dimensional vision: projective, affine, and metric representations,". *Journal of Optical Society of America-A*, 12(3):465–484, 1995.

[51] O. D. Faugeras. *Three-dimensional computer vision: a geometric viewpoint.* MIT Press, 1993.

[52] L.S. Shapiro, A. Zisserman, and M. Brady. "3D motion recovery via affine epipolar geometry,". *International Journal of Computer Vision*, 16:147–182, 1995.

[53] G. Borgefors. "Distance transformations in arbitrary dimensions,". *Computer Vision, Graphics and Image Processing*, 27(5):321–345, 1984.

[54] P. E. Danielsson. "Euclidean distance mapping,". *Computer Graphics and Image Processing*, 14:227–48, November 1980.

[55] L.J.V. Gool, T. Moons, E. Pauvels, and A. Oosterlinck. "Foundations of semi-differential invariants,". *Int. Journal of Computer Vision*, Jan 1993.

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

[57] P. Olver, G. Sapiro, and A. Tennenbaum. "Differential invariant signatures and flows in computer vision: a symmetry group approach,". In B.M.H. Romeny, ed. , *Geometry-driven diffusion in computer vision.* Kluwer Academic Press, 1995.

[58] D. Hilbert. *Theory of algebraic invariants.* Cambridge University Press, 1993.

[59] J.H. Grace and A. Young. *The algebra of invariants.* Cambridge Univ. Press, 1903.

[60] J. A. Dieudonné and J. B. Carrell. *Invariant theory, old and new.* Academic Press, 1971.

[61] J.P.S. Kung and G-C. Rota. "Invariant theory of binary forms,". *Bullettin of the American Mathematical Society*, 10(1):26–85, 1984.

[62] B. Sturmfels. *Algorithms in invariant theory.* Springer-Verlag, 1993.

[63] G. Rayna. *REDUCE:software for algebraic computation.* Springer-Verlag, 1987.

[64] R. I. Hartley. "An algoritm for self calibration from several views,". In *Proceedings CVPR*, 1994.

[65] S. Mann and R.W. Picard. "Video orbits of the projective group:a new perspective on image mosaicing,". Technical Report No.338, MIT Media Lab, 1997.

[66] D.H. Ballard and C.M. Brown. *Computer vision.* Prentice Hall, 1982.

[67] B.K.P. Horn, H.M. Hilden, and S. Negahdaripour. "Closed-form solution of absolute orientation using orthonormal matrices,". *Journal of Optical Society of America-A*, 5(7):1127–1135, 1987.

[68] Q. Luong and O. Faugeras. "The fundamental matrix:theory, algorithms, and stability analysis,". *Int. Journal of Computer Vision*, (17):43–75, 1996.

[69] S. Sullivan, L. Sandford, and J. Ponce. "Using geometric distance fits for 3-D object modeling and recognition,". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1183–1196, December 1994.

[70] N. Ayache. "Medical computer vision, virtual reality and robotics: promising research tracks,". Technical report, INRIA-EPIDAURE Project, 1994.

[71] K. Siddiqi and B.B. Kimia. "Parts of visual form: computational aspects,". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, May 1995.

[72] G. Sapiro and A. Tennenbaum. "Area and length preserving geometric invariant scale-spaces,". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17:67–72, January 1995.

[73] H. Tek and B. B. Kimia. "Volumetric segmentation of medical images by three-dimensional bubbles,". *Computer Vision and Image Understanding*, 64(3):305–322, November 1996.

[74] J. L. Mundy. "Object recognition:the search for represenatation,". In M. Hebert, J. Ponce, T. Boult, and A. Gross, eds., *Object Representation in Computer Vision*, pages 19–50. Springer Lecture Notes in Computer Science Series # 994, 1995.

[75] L. Wittgenstein. *Philosophical investigations.* The Macmillan Co., 1953.

[76] S. Wantanabe. *Pattern recognition, human and mechanical.* John Wiley and Sons, 1985.