Robust and Repeatable Fitting of Implicit Polynomial Curves to Point Data Sets and to Intensity Images

by

Tolga Tasdizen

B.S., Bogazici University, 1995

Sc.M., Brown University, 1997

Thesis

Submitted in partial fulfillment of the requirements for

the Degree of Doctor of Philosophy

in the Division of Engineering at Brown University

May 2001

This dissertation by Tolga Tasdizen

is accepted in its present form by the Division of

Engineering as satisfying the

dissertation requirements for the degree of Doctor of Philosophy

Date_____ _____

David B. Cooper, Director

Recommended to the Graduate Council

Date_____ _____

Benjamin B. Kimia, Reader

Date_____ _____

David Mumford, Reader

Approved by the Graduate Council

Date_____ _____

Peder J. Estrup

Dean of the Graduate School and Research

Vita

Tolga Tasdizen was born on August 1, 1973, in Istanbul, Turkey. He graduated on the honor roll with the Bachelor of Science degree from the Electrical and Electronics Engineering Department at Bogazici University, Turkey in 1995. He received the ScM degree in Engineering from Brown University in 1997. He has published several papers in the leading conferences in his field. He also has published a paper titled "Improving the Stability of Algebraic Curves for Applications" in the IEEE Transactions on Image Processing journal. His paper, "Boundary Estimation from Intensity/Color Images with Algebraic Curve Models" won the Best Student Paper Honorable Mention Award at the Fifteenth International Conference on Pattern Recognition, Barcelona, September 2000. He has been a Research Assistant at the Laboratory for Engineering Man/Machine Systems at Brown University since 1995. He has also held Teaching Assistantship positions for the Division of Engineering at Brown University. He worked as a research consultant at Panasonic Technologies Inc. during the summer of 1997. His research interests include computer vision, pattern recognition, artificial intelligence and algebraic geometry.

Contents

List of Tables

List of Figures

Chapter 1

INTRODUCTION

This thesis makes contributions towards the solutions of three computer vision problems using a linear algebra framework. The two major contributions are: A robust implicit polynomial curve estimation method from point data is developed. A new concept of implicit polynomial curve estimation from intensity image data (boundary curvelet detection) is introduced. We also make minor contributions in finding a set of "parts" for a shape that can be used for recognition purposes under occlusion and in methods of measuring the degree of similarity between pairs of shapes through implicit polynomial curves fitted to these shapes.

Algebraic $2D$ curves and $3D$ surfaces, also called *implicit polynomial (IP)* curves and surfaces, are powerful for shape recognition and single-computation pose estimation because of their fast fitting and functions of their coefficients which are invariant under Euclidean or Affine transformations. These properties and invariant functions of algebraic curves have been studied extensively in [59, 44, 22, 79, 20, 37, 40, 73, 72, 45, 46, 58, 74, 75, 76, 7]. Algebraic curves are usually compared with *Fourier Shape Descriptors* in terms of their usefulness for computer vision. Significant advantages over *Fourier Shape Descriptors* are

their applicability to non-star shapes, to open curves, to curves that contain gaps, and to unordered curve data. Under circumstances where these issues are not relevant, polynomials based on Fourier analysis may be very effective, and an interesting formulation relating Fourier series and polynomials is given in [68]. In [84], conversions between parametric and implicit forms are studied. *B-splines* [19], are also compared to implicit polynomial curves. They have been used for affine invariant curve matching [33, 12] and for modeling objects from image curves [13]. Here we briefly summarize some of the differences between these approaches. *B-splines* are a very efficient way of curve representation due to their boundedness, continuity, local controllability and affine invariance. *B-splines* have been used extensively in computer graphics and computer-aided design due to these properties; however, their use for shape recognition purposes has been limited because of the non-uniqueness of their parameters. In [33] affine invariant moments for *B-splines* are introduced for curve matching. The two main types of *Fourier Shape Descriptors* are: (i) that which represents a shape as a radius as a function of angle, and (ii) that which uses a complex valued function to represent the coordinates of the points along the curve as a function of arc-length. There are certain disadvantages to both approaches. (i) is limited to the set of *star-shapes* which can be represented by a single-valued radius as a function of angle; however, many interesting shapes do not fall into this category. (ii) requires the input data sets to be an ordered set of points. (i) can not be directly used for open shapes, a preprocessing step to artificially close the data curve is required. (ii) can be used for open curves, but serious difficulties with arc-length normalization arise. Arc-length parameterization is the main drawback of (ii) because arc-length can increase significantly if noise is added to the curve. Both have problems with varying data point density and gaps in the data. Implicit polynomials do not suffer from any of the problems listed above: they are directly applicable

to non-star shapes, open curves, unordered data sets and are robust to noisy data sets and inhomogoneously spaced data points. The main advantage of *Fourier Shape Descriptors* over algebraic curves has been their better stability when fit to data because they are an *explicit* representation. The significance of this is that, in cases where algebraic curves can not be fit to data with high repeatability, good recognition based on algebraic invariants is not possible.

A principal focus of this thesis is the stability issue with Implicit Polynomial Curves. We study the problem and provide a solution in two parts. After an analysis of the instability of the roots of polynomials, *gradient-one fitting* is introduced which regularizes the zero set curve of the fitted polynomials and stabilizes the coefficients of the implicit polynomial to some extent. By regularization of the zero set curve we are referring to: (i) continuity along the data tangent directions which generally amounts to a single piece of the zero set curve representing the data, and (ii) robustness to noise in the data. *gradient-one fitting* was derived from the ideas previously introduced with $3L$ fitting, [45, 7], and is practically equivalent to $3L$ fitting in terms of the results produced. However, it provides a different point of view on the stability issues of polynomials and also generalizes trivially to fitting implicit polynomials directly to intensity images. This generalization is non-trivial, if not impossible, for $3L$ fitting. The second part of the solution to stability problems with polynomials is the introduction of *Ridge Regression* regularization for implicit polynomial curve fitting. We use ridge regression methods from statistical estimation theory [30, 85] to regularize implicit polynomial fitting and thus further improve fitting repeatability. *Ridge regression* can be applied to any least squares fitting method; however, we study the case where it is applied to *gradient-one fitting*. All of the concepts and algorithms associated with *gradient-one fitting* and *ridge regression* easily generalize to data sets in $3D$ and implicit

polynomial surfaces.

Shape based object recognition and indexing into large image databases are two applications of interest to us for implicit polynomial curves. Shape-based indexing into image databases is still a very new field of research. Symmetry-based indexing of shape databases [63, 64, 43] is an interesting example of the application of model based shape representation to this field. In this thesis we introduce *Polynomial Interpolated Measures* as a fast and accurate way to compare pairs of shapes through algebraic curves. *Polynomial Interpolated Measures* are meant to complement a new set of invariant functions of the IP coefficients [75] for object recognition and shape-based indexing. These measures are not affected by possible unstabilities in implicit polynomial coefficients and they provide good interpolation properties through intervals of missing data. Partial occlusion is an important hurdle for shape-based computer vision applications. Partial occlusion occurs when an object is behind another object with respect to the camera and is partially obscured in the image. Can such objects be recognized from the information present in the image? We introduce *maximum length implicit polynomial patches* as invariant descriptors of a shape. These patches provide a redundant set of features that describe the parts of a shape. The usefulness of invariant patches stems from the simple idea that even if parts of a shape are missing possibly due to local failure of an edge detection step, or if there is occlusion, there will be parts of the shape that are not affected by these complications. Consequently, any shape descriptor computed only from such unaffected parts will still be a useful feature. Finding maximum length invariant patches is easier than finding physically meaningful parts of shapes, a more ambitious approach to dealing with occlusion.

The theory we develop in Chapters 2-4 is built on the assumption that silhouettes of objects are available as sets of (x, y) pairs. This amounts to at least assuming the presence

of a segmentation step. However, segmentation is not a trivial issue and assuming that pre-segmented images will be available in a general application is not realistic. Thus, a way of finding partial or complete boundary curves from images without relying on a segmentation step is desired. The last contribution of this thesis is in using implicit polynomial curves for edge detection and grouping purposes. We introduce a new way of fitting polynomials to an intensity images such that their level sets represent curves of edges that might be present in the images. This fitting is a generalization of *gradient-one fitting*, also a contribution of this thesis. This development should provide the missing link between the implicit polynomial curve framework of object recognition/pose estimation/shape based indexing and real world data. We propose to substitute this new fitting for edge detectors and edge grouping methods for finding robust shape features in images that can be used for object recognition. Thus, we propose this new type of fitting as an edge curve finder. It performs successfully on a wide range of images because it is "almost" parameter independent. It is also not plagued by the combinatorics of saliency optimization involved in edge grouping.

1.1 Implicit Polynomial Curves

Formally, a planar algebraic curve is specified by the zero set of a 2D polynomial of degree d given by

$$\begin{aligned} f_d(x,y) = \sum_{0\le u,v;u+v\le d} a_{uv}x^u y^v = & a_{00} + a_{10}x + a_{01}y + a_{20}x^2 + a_{11}xy + a_{02}y^2 + \\ & a_{30}x^3 + a_{21}x^2y + a_{12}xy^2 + a_{03}y^3 + \ldots + \\ & a_{d0}x^d + a_{(d-1)1}x^{d-1}y + \ldots + a_{0d}y^d = 0 \end{aligned} \tag{1.1}$$

The *homogeneous binary polynomial* of degree r in x and y is a form, e.g., $a_{20}x^2 + a_{11}xy + a_{02}y^2$ is the 2^{nd} degree form. The homogeneous polynomial of degree d is the so-called leading form. An algebraic curve of degree 2 is a conic, degree 3 a cubic, degree 4 a quartic, and so on. Polynomial $f_d(x, y)$ can also be represented in vector form. Define the *coefficient vector* $\mathbf{a}$ and the *monomial vector* $\mathbf{y}$ for point (x, y) as

$$\mathbf{a} = \begin{pmatrix} a_{00} \\ a_{10} \\ a_{01} \\ \cdots \\ a_{d0} \\ a_{(d-1)1} \\ \cdots \\ a_{0d} \end{pmatrix} \quad \mathbf{y} = \begin{pmatrix} 1 \\ x \\ y \\ \cdots \\ x^d \\ x^{d-1}y \\ \cdots \\ y^d \end{pmatrix} \tag{1.2}$$

which have dimension $\underbrace{\frac{(d+1)(d+2)}{2}}_{p} \times 1$. Then

$$f_d(x, y) = \mathbf{y}^T \mathbf{a} \tag{1.3}$$

where superscript T denotes vector and matrix transpose. In general, the vector notation is convenient for implicit polynomial fitting since fitting can be set within a linear least squares framework as detailed in Section 2.1. Other representations, such as the complex representation, is convenient for pose estimation and obtaining a complete set of Euclidean invariants in 2D [75], the covariant conics decomposition is useful for 2D affine alignment

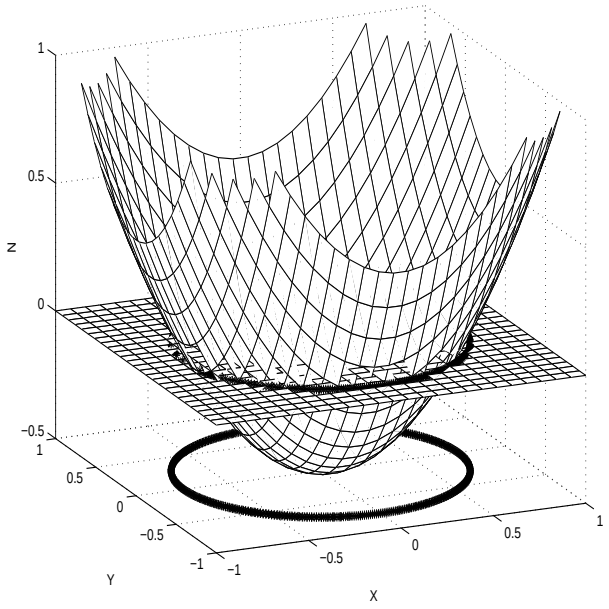


Figure 1.1: A circle can be represented as the zero set of a second degree polynomial in x and y.

and invariants [76], and the tensor representation [74] provides a useful framework for pose estimation in 3D.

A shape is represented by the zero set of $f_d(x)$, i.e., the set of points (x, y) satisfying the IP equation $f_d(x, y) = 0$ which is the intersection of the surface defined by an explicit polynomial $z = f_d(x, y)$ with the plane $z = 0$; see Figure 1.1.

1.2 Summary of Contributions

The two major contributions of this thesis are:

• The developments of the concepts and algorithms for *gradient-one fitting* regularized by *ridge regression* for robustly fitting implicit polynomial curves to point data.

• The developments of the concepts and algorithms for boundary curvelet detection by automatically fitting implicit polynomial curves to windows of intensity images without the need for prior edge detection or segmentation. The grouping of these curvelets to obtain

longer and more informative curves.

A minor contribution is :

• The further developments of *Polynomial Interpolated Measures* for comparing shapes using implicit polynomial curves fitted to them and *maximum length invariant patches* as tools for object recognition in the presence of missing data and occlusion.

1.3 Overview of the Thesis

Chapter 2 starts with a discussion of the stability issue of implicit polynomial curves and develops the *gradient-one fitting* algorithm for robustly fitting implicit polynomial curves to point data. Unstable subspaces of implicit polynomial coefficients and the global regularization of implicit polynomial fitting techniques using *ridge regression*, a regularization technique in statistics similar to principal component analysis, is discussed in Chapter 3. Chapter 4 discusses two tools for object recognition with implicit polynomial curves: *polynomial interpolated measures* which is a way of locally comparing two shapes through implicit polynomial curves fitted to them, and *maximum length invariant patches* which are implicit polynomial curve patches which can be used for object recognition under occlusion. Chapter 5 introduces a new type of implicit polynomial curve fitting: fitting to windows of image intensity data without prior segmentation or edge detection. Edge curve detection with the fitted implicit polynomial curves is also discussed in Chapter 5. Chapter 6 discusses how to easily piece together curvelets obtained with the fitting described in Chapter 5 to obtain longer curves. Finally, in Chapter 7, we summarize the contributions of this thesis and discuss a few directions for future research.

Chapter 2

IMPLICIT POLYNOMIAL CURVE FITTING

The classical least-squares fitting of algebraic curves, Section 2.1, especially the more interesting cases of fitting higher degree polynomials, suffers three major problems: local inconsistency with the continuity of the dataset; local over-sensitivity of the polynomial zero set around the data to small data perturbations; instability of the coefficients due to excessive degrees of freedom in the polynomial. Researchers have investigated various approaches to fitting in order to improve upon the classical least-squares method. Substituting an approximate Euclidean distance for algebraic distance [79] is much more stable than the classical least squares algorithm, in many cases gives useful fits, but in other cases the improvement is not sufficient to solve these major problems. Similarly, the use of the exact Euclidean distance provides better results than the algebraic distance [73]; nevertheless the fitting is sometimes not stable enough and the minimization is solved iteratively, a time consuming process. Another attempt to improve the stability of the fit was the development of fitting algorithms which ensure that the obtained zero set is bounded [38, 80, 58], but

the last one is for $2nd$ degree curves only, and increased stability for all and fitting speed for the former two are still desired. Non-linear parameterizations of polynomials that are guaranteed to satisfy certain topological properties [39] – boundedness and having a zero set that is contained within another shape such as an ellipse – are interesting, and their relative merits need to be studied further. The problem of an excessive number of parameters in implicit polynomial curve and surface representations was first studied in [72] in the framework of Bayesian estimation.

2.1 Classical Least-Squares Fitting

The classical and simplest way to fit an algebraic curve to a set of data points $\left\{(x_k, y_k)\right\}_{k=1}^{m}$ is to minimize the algebraic distance

$$E_{\text{algebraic}} = \sum_{k=1}^{m} f_d(x_k, y_k)^2. \tag{2.1}$$

Let $\mathbf{y_k}$ be the monomial vector for point (x_k, y_k) which can be obtained by the substitutions $x = x_k$ and $y = y_k$ in equation (1.2). Then using the vector representation of f_d which was introduced in equation (1.3) we obtain

$$\begin{aligned} E_{\text{algebraic}} &= \sum_{k=1}^{m} (\mathbf{y_k}^T \mathbf{a})(\mathbf{y_k}^T \mathbf{a}) \\ &= \sum_{k=1}^{m} \mathbf{a}^T \mathbf{y_k} \mathbf{y_k}^T \mathbf{a} \\ &= \mathbf{a}^T \left(\sum_{k=1}^{m} \mathbf{y_k} \mathbf{y_k}^T \right) \mathbf{a} \end{aligned} \tag{2.2}$$

Define the *monomial matrix* for the given set of data points as the $p \times m$ matrix which has as its columns the monomial vectors $\mathbf{y_k}$ for $1 \leq k \leq m$

$$\mathbf{M} = \begin{bmatrix} \mathbf{y_1} & \mathbf{y_2} & \cdots & \mathbf{y_m} \end{bmatrix} = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_m \\ y_1 & y_2 & \cdots & y_m \\ & & \cdots & \\ x_1^d & x_2^d & \cdots & x_m^d \\ x_1^{d-1}y_1 & x_2^{d-1}y_2 & \cdots & x_m^{d-1}y_m \\ & & \cdots & \\ y_1^d & y_2^d & \cdots & y_m^d \end{bmatrix} \tag{2.3}$$

$\mathbf{M}$ is also called the *design matrix*, and

$$\begin{aligned} \mathbf{S} = \mathbf{M}\mathbf{M}^T &= \begin{bmatrix} \mathbf{y_1} & \mathbf{y_2} & \cdots & \mathbf{y_m} \end{bmatrix} \begin{bmatrix} \mathbf{y_1}^T \\ \mathbf{y_2}^T \\ \cdots \\ \mathbf{y_m}^T \end{bmatrix} \\ &= \sum_{k=1}^{m} \mathbf{y_k}\mathbf{y_k}^T \end{aligned} \tag{2.4}$$

is the $p \times p$ *scatter matrix* of the monomials. $\mathbf{S}$ is symmetric non-negative definite by definition provided $m \geq p$. Substituting equation (2.4) in equation (2.2) we obtain

$$E_{\text{algebraic}} = \mathbf{a}^T\mathbf{M}\mathbf{M}^T\mathbf{a} = \mathbf{a}^T\mathbf{S}\mathbf{a} \tag{2.5}$$

To avoid the trivial zero solution in the minimization of $E_{\text{algebraic}}$ in equation (2.5), a constraint such as $\|\mathbf{a}\|^2 = 1$ is imposed which modifies the problem to

$$\min_{\mathbf{a}} \left(\mathbf{a}^T \mathbf{S} \mathbf{a} + \lambda (\mathbf{a}^T \mathbf{a} - 1) \right) \tag{2.6}$$

with the introduction of Lagrange multiplier λ. The solution to equation (2.6) is given by the unit eigenvector associated with λ_{min}, the smallest eigenvalue of $\mathbf{S}$ [79]. Consequently, the classical least-squares fitting algorithm consists of computing the monomial scatter matrix $\mathbf{S}$ from a set of data points, and then finding the unit eigenvector of $\mathbf{S}$ associated with its smallest eigenvalue. Although this algorithm is affine covariant [20, 79], most of the time it is not of any practical use due to the following problems: The fitted zero set does not respect the continuity of the original data set as illustrated in Figure 2.1(a)-(d) and also Figure 3.3(a) and (d). This problem undermines the use of classical fitting for obtaining good representations of the data. Moreover, results are highly sensitive to small errors in the data. Even small perturbations in the data can lead to zero sets that have no resemblance to the zero sets prior to the perturbations in the data, Figure 2.1(a)-(d). Even with low order degrees, depending on the structure of the given data set, $\mathbf{S}$ may not provide a stable unique eigenvector under small perturbations. For example, several eigenvalues can have similar values to the smallest one, and thus the solution will span a subspace in the coefficient space when small perturbations are added to the data set. Consequently, classical fitting is also practically useless for shape recognition purposes based on the coefficients of the fitted polynomial curves.

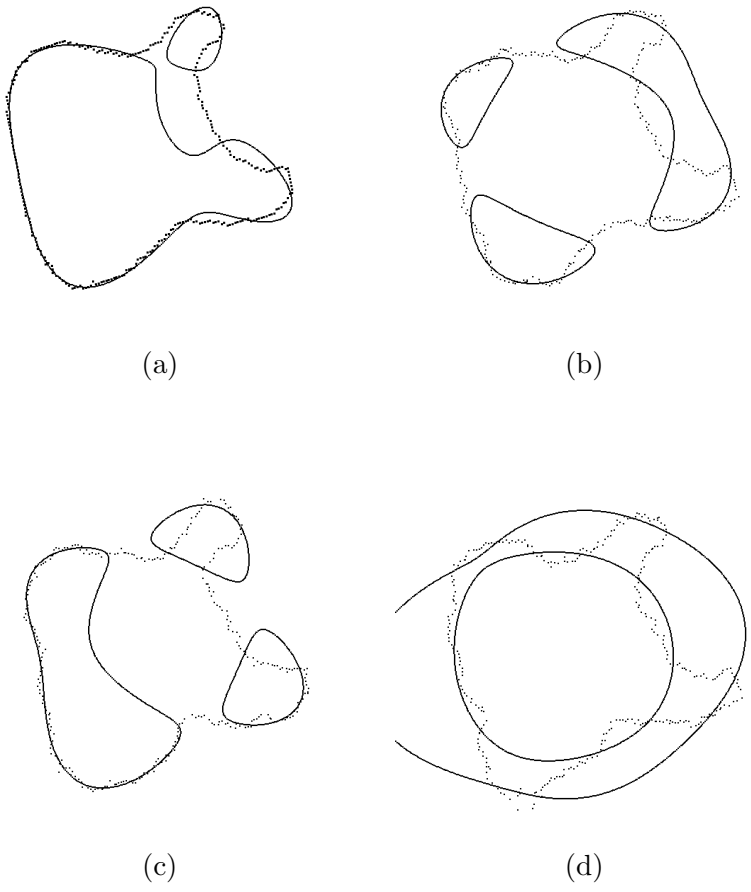


Figure 2.1: Classical least-squares algorithm gives unstable 4^{th} degree IP fits when the data is perturbed with white noise which has standard deviation equal to %5 of the shape size. (a) is the original data set and fit, (b)-(d) are perturbed versions of the same data set and fits.

-1.00000	-5.00000	-8.91725	-13.99236 ± 2.518831 i
-2.00000	-6.00001	-20.84691	-16.73074 ± 2.812621 i
-3.00000	-6.99970	-10.09527 ± 0.643501 i	-19.50244 ± 1.940331 i
-4.00000	-8.00727	-11.79363 ± 1.652331 i	

Table 2.1: Roots of the perturbed Wilkinson Polynomial

2.2 Pathological Polynomials

Although we are interested in 2D polynomials, i.e., functions of x and y, it is instructive to first study stability in the 1D case. It is well known that some 1D polynomials, in particular polynomials of high degree, are ill-conditioned. Consider the pathological example due to Wilkinson [1]: $(x+1)(x+2)\ldots(x+20) = x^{20} + 210x^{19} + \ldots + 20!$. This polynomial has very large coefficients and its roots are -1, -2, -3,..., -20. An accurate calculation of the perturbed roots as given in [1] to five decimals after a tiny change of 2^{-23} is applied to the coefficient of x^{19}, are shown in Table 2.1. Observe that the perturbations of the bigger roots are disproportionately large compared to the tiny change applied to the coefficient of x^{19}. Though this example demonstrates how ill conditioned some polynomials are, it does not mean that all polynomials are so, and as a consequence that all algorithms using high degree polynomials have to be rejected as apriori unstable. In fact, we will demonstrate that it is possible to work in a subspace of non-pathological polynomials. First, let's try to understand the pathology of the Wilkinson polynomial. A plot of this polynomial would show varying oscillation amplitude between its roots. This type of ill-conditioned behavior of polynomials is well-known in the context of interpolation theory. Indeed, the Wilkinson polynomial is an example of Lagrange interpolation at 20 points, and it is known that Lagrange interpolation suffers from oscillation problems between data points. This is the so-called Runge problem [18]. One known solution is to change the way the interpolation is

carried out. Hermite interpolation, where the first derivative of the polynomial is controlled in addition to the value of the polynomial at each given point, can be proven to converge properly for all continuous functions when the number of sampling points and thus the degree of the polynomial increases.

We are referring to interpolation theory and Hermite polynomials because they provide us with very useful insight in trying to improve the classical least-squares fitting algorithm. In essence, the problem with polynomials is that the functional relationship between its coefficients and its roots is highly non-linear. Let $p_n(x)$ be a polynomial defined as: $p_n(x) = \sum_{0 \leq j \leq n} a_j x^j = a_0 + a_1 x + a_2 x^2 + \ldots + a_n x^n$. Roots x_k of this polynomial are defined by $p_n(x_k) = 0$. This last equation can be seen as an implicit equation for root x_k where this root is a function of coefficients a_j. To determine the sensitivity of this root to small changes of the coefficients, let us apply a small change Δa_j to a. We are interested in determining Δx, the change root x_k undergoes. We want to find Δx for which $p_n(x_k + \Delta x, a_j + \Delta a) = 0$. Expand p_n around $x = x_k$ by a first order Taylor series approximation:

$$\begin{aligned} p_n(x_k + \Delta x, a_j + \Delta a) =& p(x_k, a_j) + \frac{\partial p_n}{\partial x}(x = x_k)\Delta x + \frac{\partial p_n}{\partial a_j}\Delta a \\ p_n(x_k + \Delta x, a_j + \Delta a) =& \frac{\partial p_n}{\partial x}(x = x_k)\Delta x + \frac{\partial p_n}{\partial a_j}\Delta a \end{aligned}$$

Using $\frac{\partial p_n}{\partial a_j} = x^j$ we then obtain

$$\frac{\Delta x}{\Delta a} = -\frac{x_k^j}{\frac{\partial p_n}{\partial x}(x = x_k)} \tag{2.7}$$

Equation (2.7) has important consequences. It is desired that small or large changes in the coefficients produce small or large changes, respectively, in the roots, and vice versa. Thus we should require that $\| \frac{\Delta x_k}{\Delta a_j} \|$ be as close to 1 as possible. Due to the numerator x_k^j, we

Root	Estimated Perturbation	Root	Estimated Perturbation
-1	-9.7998×10^{-25}	-11	-5.5366×10^{0}
-2	9.7620×10^{-18}	-12	2.3663×10^{1}
-3	-1.9477×10^{-13}	-13	-7.2188×10^{1}
-4	2.6102×10^{-10}	-14	1.5890×10^{2}
-5	-7.2448×10^{-08}	-15	2.5261×10^{2}
-6	6.9438×10^{-6}	-16	2.8699×10^{2}
-7	3.0308×10^{-4}	-17	-2.2702×10^{2}
-8	7.1163×10^{-3}	-18	1.1868×10^{2}
-9	-1.0006×10^{-1}	-19	-3.6837×10^{1}
-10	9.0528×10^{-1}	-20	5.1379×10^{0}

Table 2.2: Estimates for root perturbations.

see that x_k should be close to values 1.0 or -1.0; otherwise, the effect of a small coefficient perturbation has a larger effect on roots with large absolute values. This explains why roots with large absolute values are less stable than others for the Wilkinson polynomial, Table 2.1. Due to the denominator of equation (2.7), we deduce that the sensitivity to a small coefficient perturbation is also directly dependent on the value of the first derivative of the polynomial at the root location. The Wilkinson's polynomial has derivatives 19!0!, $-18!1!$, 17!2!, ..., $-0!19!$ at -1, -2, -3, ..., -20 respectively. These huge variations in the first derivative $\frac{dp_n}{dx}$ contributes to the instability of the roots with respect to coefficient perturbations. Using equation (2.7), we predict, with a first order Taylor expansion, the perturbations of the roots of the Wilkinson polynomial when 2^{-23} is added to $a_{19} = 210$. The root perturbation predictions are tabulated in Table 2.2. These values are in good accordance with the differences between the original roots, $-20, \ldots, -1$ and the real perturbed roots, Table 2.1. Tables 2.1 and 2.2 provide an experimental validation of equation (2.7).

2.3 Gradient-one Fitting

The insight developed in Section 2.2 into how polynomials can be ill-conditioned, enables us to determine a fuzzy subset of the polynomial space that is the set of well-conditioned polynomials. How can we define a well-conditioned polynomial? For the problem at hand, it is a polynomial for which the relationship between its roots and its coefficients is such that small changes in one induces small changes in the other and larger changes induce larger changes. In Section 2.2, it was argued that a 1D polynomial should have root values and first derivative values at the root locations, all close to 1.0 or -1.0. Intuitively, we can extend this result to 2D polynomials: a set of polynomials satisfying these constraints exactly in 2D are the powers of the unit circle: $\frac{1}{2n}\big((x^2+y^2)^n-1\big)$. Members of the set of polynomials "close" to these polynomials in the coefficient space can be considered to be well-conditioned.

An implicit polynomial fitting algorithm is stable if its outputs are well-conditioned implicit polynomials. Consequently, such an algorithm will have to **bias** its output towards such polynomials. The first requirement for stable fitting is to apply a data set standardization to force the data points to be close to the unit circle, and thus indirectly to force the zero set of the polynomial to be as close as possible to the unit circle. The data set standardization consists of centering the data-set center of mass at the origin of the coordinate system and then scaling it by dividing the coordinates of each point by the square root of the average of the eigenvalues of the 2×2 matrix of second order moments. Given a set of data points $\Big\{(x_k, y_k)\Big\}_{k=1}^{m}$, the center of mass is (c_x, c_y) where $c_x = \frac{1}{m}\sum_{k=1}^{m} x_k$ and $c_y = \frac{1}{m}\sum_{k=1}^{m} y_k$. Then the matrix of second order moment averages of the centered data

set is

$$\frac{1}{m}\begin{bmatrix} \sum_{k=1}^{m}(x_k - c_x)^2 & \sum_{k=1}^{m}(x_k - c_x)(y_k - c_y) \\ \sum_{k=1}^{m}(x_k - c_x)(y_k - c_y) & \sum_{k=1}^{m}(y_k - c_y)^2 \end{bmatrix} \tag{2.8}$$

and we choose to define the object size s as the square root of the average of the eigenvalues of the matrix in equation (2.8). This is a Euclidean invariant measure of the object size. Thus by data set standardization we are setting this measure of object size to 1. Then, the set of standardized data points are $\left\{(\frac{x_k - c_x}{s}, \frac{y_k - c_y}{s})\right\}_{k=1}^{m}$. In the rest of this chapter and in Chapters 3 and 4 we will assume that data sets have been standardized.

The second requirement is to control the value of the first derivatives along the zero set, i.e, the gradient of the 2D polynomial:

$$\nabla f_d = \begin{pmatrix} \frac{\partial f_d}{\partial x} \\ \\ \frac{\partial f_d}{\partial y} \end{pmatrix} \tag{2.9}$$

By definition, the gradient vector along the zero set curve of the polynomial is always perpendicular to this curve. Thus, if we can compute the local tangent to the data curve at each point of the data set, we propose to force the implicit polynomial gradient to be perpendicular to the local tangent and with unit norm. This will force the zero set of the polynomial to respect the local continuity of the data set. The calculation of the tangent to the data set at a point does not pose a serious problem. If the data set is ordered as a curve, we calculate local tangents to the data using the lines going through consecutive data points. If the data is not ordered, a fast distance transform [81, 74] can be used to generate level sets as in the 3L fitting algorithm [45, 7] or to indirectly calculate tangent directions. Another possibility that was implemented in our lab is to compute local tangent

directions at a point by using all data points that are within a certain distance from it. Local normal directions are then perpendicular to this direction and a final step checks the consistency of the normal vectors; in other words, it modifies the normal vectors such that they all point towards one side of the curve. This approach was successfully tested on data sets with moderate amounts of noise. When working with real images, level sets may also be generated as described in [48]; or if the input to the fitting algorithm comes from an edge detector, edge orientations can be used as the tangent directions. We develop our own methods for direct fitting of implicit polynomials to intensity images in Chapter 5. In the case of open curves where no notion of inside/outside is available, both cases can be considered resulting in two fitted IP curves which have coefficient vectors related by multiplication with -1. We do not apply any smoothing in computing the tangents even in the presence of noise; indeed, it is the fitting process which takes care of smoothing the fluctuations in the tangent direction along the curve given that there are enough points on the dataset; at least a few times the dimensionality of the IP coefficient vector.

The proposed fitting technique is set as a least-squares problem with the following additional constraints: Local tangential and normal directional derivatives of the IP must be as close as possible to 0 and 1, respectively. These constraints add two terms to $E_{\text{algebraic}}$ in equation (2.1) to yield

$$E_{\text{grad}} = \sum_{k=1}^{m} \left(f_d(x_k, y_k)^2 + \mu^2 \Big(\big(\mathbf{n_k}^T \nabla f_d(x_k, y_k) - 1 \big)^2 + \big(\mathbf{t_k}^T \nabla f_d(x_k, y_k) \big)^2 \Big) \right) \tag{2.10}$$

where $\mathbf{t_k}$ and $\mathbf{n_k}$ are the local tangent and normal vectors to the data set at (x_k, y_k) computed as described above. Thus, $\mathbf{t_k}^T \nabla f_d(x_k, y_k)$ and $\mathbf{n_k}^T \nabla f_d(x_k, y_k)$ are the derivatives of f_d in the $\mathbf{t_k}$ and $\mathbf{n_k}$ directions at (x_k, y_k), respectively. μ^2 is the relative weight on

the gradient cost terms with respect to the f_d^2 cost term. By using the vector notation introduced in equation (1.3) in the polynomial gradient equation (2.9), we deduce the vector form of the gradient:

$$\nabla f_d(x,y) = \begin{pmatrix} \frac{\partial}{\partial x}\mathbf{y}^T\mathbf{a} \\ \frac{\partial}{\partial y}\mathbf{y}^T\mathbf{a} \end{pmatrix} = \underbrace{\begin{pmatrix} \frac{\partial \mathbf{y}^T}{\partial x} \\ \frac{\partial \mathbf{y}^T}{\partial y} \end{pmatrix}}_{2\times p} \underbrace{\mathbf{a}}_{p\times 1}$$

$$= (\nabla \mathbf{y}^T)\mathbf{a}$$

We can obtain $\frac{\partial \mathbf{y}}{\partial x}$ and $\frac{\partial \mathbf{y}}{\partial y}$, the x and y partial derivative monomial vectors, from equation (1.3)

$$\frac{\partial \mathbf{y}}{\partial x} = \frac{\partial}{\partial x}\begin{pmatrix} 1 \\ x \\ y \\ \cdots \\ x^d \\ x^{d-1}y \\ \cdots \\ xy^{d-1} \\ y^d \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ \cdots \\ dx^{d-1} \\ (d-1)x^{d-2}y \\ \cdots \\ y^{d-1} \\ 0 \end{pmatrix} \quad , \quad \frac{\partial \mathbf{y}}{\partial y} = \frac{\partial}{\partial y}\begin{pmatrix} 1 \\ x \\ y \\ \cdots \\ x^d \\ x^{d-1}y \\ \cdots \\ xy^{d-1} \\ y^d \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ \cdots \\ 0 \\ x^{d-1} \\ \cdots \\ (d-1)xy^{d-2} \\ dy^{d-1} \end{pmatrix} \tag{2.11}$$

Let $\nabla \mathbf{y_k}$ be the $p \times 2$ gradient monomial matrix at point (x_k, y_k). Then $\nabla \mathbf{y_k} \mathbf{n_k}$ is the $p \times 1$ normal derivative monomial vector and $\nabla \mathbf{y_k} \mathbf{t_k}$ is the $p \times 1$ tangent derivative monomial vector. Finally, $\mathbf{a}^T \nabla \mathbf{y_k} \mathbf{n_k}$ and $\mathbf{a}^T \nabla \mathbf{y_k} \mathbf{t_k}$ are the derivatives of f in the normal and tangent

directions. After substitution in equation (2.10), we expand E_{grad} in vector notation as

$$\begin{aligned} E_{\text{grad}} =& \mathbf{a}^T \Big(\underbrace{\sum_{k=1}^{m} \mathbf{y_k} \mathbf{y_k}^T}_{\mathbf{M}\mathbf{M}^T = \mathbf{S}} \Big) \mathbf{a} \\ &+ \mu^2 \mathbf{a}^T \Big(\underbrace{\sum_{k=1}^{m} \nabla \mathbf{y_k} \mathbf{n_k} \mathbf{n_k}^T \nabla \mathbf{y_k}^T}_{\mathbf{M_n}\mathbf{M_n}^T = \mathbf{S}_n} \Big) \mathbf{a} - 2\mu^2 \mathbf{a}^T \Big(\underbrace{\sum_{k=1}^{m} \nabla \mathbf{y_k} \mathbf{n_k}}_{\mathbf{G}_n} \Big) + \mu^2 m \\ &+ \mu^2 \mathbf{a}^T \Big(\underbrace{\sum_{k=1}^{m} \nabla \mathbf{y_k} \mathbf{t_k} \mathbf{t_k}^T \nabla \mathbf{y_k}^T}_{\mathbf{M_t}\mathbf{M_t}^T = \mathbf{S}_t} \Big) \mathbf{a} \end{aligned} \tag{2.12}$$

The first, second and third lines of the above equation correspond to the expansions of the first, second and third terms terms in equation (2.10), respectively. In this equation, $\mathbf{M}$ and $\mathbf{S}$ are the monomial design matrix and the monomial scatter matrix as introduced before in equations (2.3) and (2.4). We will use similar constructions for the matrices $\mathbf{M_n}$, $\mathbf{M_t}$, $\mathbf{S_n}$, and $\mathbf{S_t}$ which are introduced in equation (2.12). The normal monomial matrix and the normal scatter matrix are given as

$$\mathbf{M_n} = \begin{bmatrix} \nabla \mathbf{y_1} \mathbf{n_1} & \nabla \mathbf{y_2} \mathbf{n_2} & \dots & \nabla \mathbf{y_m} \mathbf{n_m} \end{bmatrix}, \ \mathbf{S_n} = \mathbf{M_n} \mathbf{M_n}^T \tag{2.13}$$

and the tangent monomial matrix and the tangent scatter matrix as

$$\mathbf{M_t} = \begin{bmatrix} \nabla \mathbf{y_1} \mathbf{t_1} & \nabla \mathbf{y_2} \mathbf{t_2} & \dots & \nabla \mathbf{y_m} \mathbf{t_m} \end{bmatrix}, \ \mathbf{S_t} = \mathbf{M_t} \mathbf{M_t}^T. \tag{2.14}$$

$\mathbf{M_n}$ and $\mathbf{M_t}$ are of dimension $p \times m$, $\mathbf{S_n}$ and $\mathbf{S_t}$ are of dimension $p \times p$. $\mathbf{G_n}$ is the sum of the gradients of the monomials in the normal direction and has size $p \times 1$. Define a new

overall monomial matrix

$$\mathcal{M} = \underbrace{\begin{bmatrix} \mathbf{M} & \mu\mathbf{M_n} & \mu\mathbf{M_t} \end{bmatrix}}_{p \times 3m}, \tag{2.15}$$

and a new overall scatter matrix

$$\mathcal{S} = \mathcal{M}\mathcal{M}^T = \underbrace{\mathbf{M}\mathbf{M}^T + \mu^2\mathbf{M_n}\mathbf{M_n}^T + \mu^2\mathbf{M_t}\mathbf{M_t}^T}_{p \times p}, \tag{2.16}$$

and a target vector composed of m 0's for the f^2 terms, m μ's for the normal derivative terms and m 0's for the tangent derivative terms

$$\mathbf{b} = \begin{pmatrix} 0 & \dots 0 & \mu & \dots & \mu & 0 & \dots & 0 \end{pmatrix}^T. \tag{2.17}$$

Then E_{grad} can be rewritten as

$$\begin{aligned} E_{\text{grad}} =& || \mathcal{M}^T\mathbf{a} - \mathbf{b} ||^2 \\ =& \left(\mathcal{M}^T\mathbf{a} - \mathbf{b}\right)^T \left(\mathcal{M}^T\mathbf{a} - \mathbf{b}\right) \\ =& \mathbf{a}^T\mathcal{M}\mathcal{M}^T\mathbf{a} - 2\mathbf{a}^T\mathcal{M}\mathbf{b} + \mathbf{b}^T\mathbf{b} \end{aligned} \tag{2.18}$$

The second term is equivalent to $-2\mathbf{a}^t\mathbf{G_n}$ because the only non-zero elements in $\mathbf{b}$ are at the positions corresponding to $\mathbf{M_n}$. The coefficient vector that minimizes E_{grad} is the estimator

$$\mathbf{a} = \mu^2(\mathcal{M}\mathcal{M}^T)^{-1}\mathcal{M}\mathbf{b} = \mu^2\mathcal{S}^{-1}\mathbf{G_n} \tag{2.19}$$

This minimization is a linear least squares problem. We named this algorithm *gradient-one fitting*. Like Hermite interpolation [18], *gradient-one fitting* is Euclidean invariant, see

Section 3.3, respectively, *but not affine invariant*. *Gradient-one fitting* is also *scale invariant* since the data standardization step sets our Euclidean invariant measure of the size of the data to 1 before fitting. Data set standardization introduces a numerical advantage by improving the condition number[1] of the scatter matrix $\mathcal{S} = \mathbf{S} + \mu^2(\mathbf{S_N} + \mathbf{S_T})$ of the problem (2.19). The condition number gives an idea of the numerical stability of linear algorithms such as the computation of the inverse of a matrix [25]. Data standardization improves the stability of the fits; however, if the estimator for the non-standardized data is needed, the estimator for the standardized data should be computed and then back-transformed into the original coordinate system [78].

The necessity to introduce information about the first derivatives was first pointed out in [45, 7] and handled in a linear way with the so-called 3-levels (3L) fitting algorithm. The idea of the 3L fitting is to constrain the polynomial to fit not only the data set but also two level sets of the distance transform of this data set, thus preventing the presence of singularities of the polynomial f in the vicinity of the data to be fitted. Therefore, indirectly, 3L fitting places soft constraints on the gradient of the fitted implicit polynomial curve. In fact, it can be proved that the *gradient-one fitting* algorithm is similar to the 3L fitting algorithm expanded to the first order with respect to the inter-level distance parameter. Although, *gradient-one* and $3L$ fitting algorithms are almost equivalent in view of fitting IP curves to point data sets, *gradient-one fitting* has a significant advantage over $3L$ fitting in the case of fitting polynomial models directly to intensity images. *Gradient-one fitting* can easily be generalized to this case whereas the same generalization for $3L$ does not hold. We will elaborate further on this point in Section 5.3.

To gain insight into how local consistency is achieved by controlling the gradient across

[1]The condition number of a matrix is the ratio of its largest eigenvalue to its smallest eigenvalue.

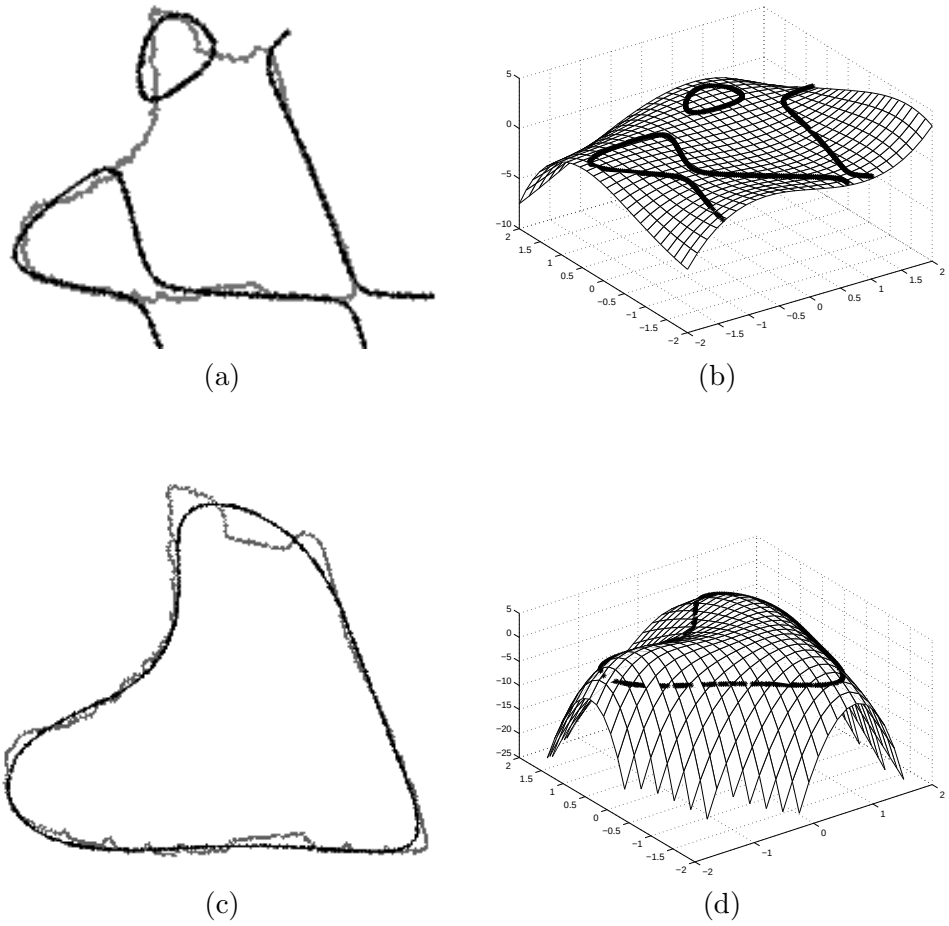
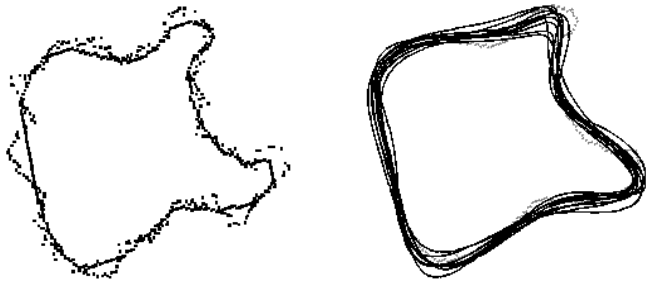


Figure 2.2: Comparison of implicit polynomial zero sets and polynomial graphs obtained by classical fitting (a)-(b) to *gradient-one fitting* (c)-(d).



(a) (b)

Figure 2.3: (a) An example of a data set perturbed with white noise having standard deviation % 5 of the shape size, (b) 10 superimposed 4^{th} degree polynomial fits with the *gradient-one fitting* algorithm to perturbed data sets like the one in (a).

the data set, we examine Figure 2.2. It can be deduced from Figure 2.2(d) that the gradient direction along the zero set obtained by *gradient-one fitting* consistently points into the shape whereas in Figure2.2(b) it can be seen that this direction switches between inwards and outwards. The zero set from the solution of equation (2.6) is broken into pieces as can be seen in Figure 2.2(a) whereas in Figure. 2.2(c) the zero set is a smooth representation of the data curve. Also notice that in the vicinity of the data, the surface in Figure 2.2(b) is flatter than is the surface in Figure 2.2(d) which means that with small perturbations of the data, classical fitting is prone to much larger changes in the zero set. In addition to better stability of the zero set and better shape representation power, *gradient-one fitting* also provides better interpolation properties which allow implicit polynomial curves to be robust to a certain amount of missing data along the curve. The stability of the zero set achieved by the *gradient-one fitting* algorithm is an important improvement over classical fitting techniques. It can be seen in Figure 2.3(b) that the zero sets of the resulting fits are stable under local data perturbations. The changes in the fitted implicit polynomial curves are much smaller in Figure 2.3(b) compared to those in Figure 2.1(a)-(d). Further comparisons of the results obtained by classical least-squares fits and *gradient-one* fits can

be found in Figure 3.3.

Parameter μ has important effects on the properties of the fits. It controls the relative weight of the gradient constraint with respect to the algebraic distance constraint. The effect of the gradient constraint on the zero set of the fit is a smoothing of the high curvature areas. Figure 2.4 is an example of smoothing of the zero set when μ is increasing. In all our examples, μ is fixed to $\frac{1}{7}$ which gives satisfying results as shown in Figure 3.3(b) and (e). This value is a good trade-off between the accuracy of the representation and the stability of the fitted parameters. However, better stability of the estimated polynomial coefficients can be achieved with equal weights on the gradient and data fit constraints as shown in Figure 2.4 because the resulting fits will be "closer" to the set of well behaved polynomials, the powers of the unit circle. Thus, the optimal choice of μ depends on the application, with values closer to 1 preferred for object recognition and smaller values preferred for shape representation. In a more general framework, μ can be made a user-specified function along the length of the curve providing more control for interactive curve representation purposes.

Information about the higher order derivatives such as curvature can be incorporated into *gradient-one fitting* to provide additional constraints. The polynomial $f(x, y)$ defines a parameterized surface $\big(x, y, f(x, y)\big)$ as in Figure 1.1. We want to match the curvature of the zero set curve of the implicit polynomial curve to the curvature measured from the data set curve. In measuring curvature from the input data set the decisive factor is the accuracy of the data; if the set of data points smoothly determine a curve then using curvature information in the fitting will in general improve the results. However, with increasing noise in the input data, curvature terms will become susceptible to noise before the tangent and normal derivative terms because they are higher order derivatives and differentiation amplifies noise. If curvature information is going to be used in the fitting, we have to derive

an expression for the curvature of the zero set curve of the implicit polynomial. Classical differential geometry of parameterized surfaces [71] provides the answer. For a planar curve $\big(x(t), y(t)\big)$ such as the zero set of the polynomial, curvature is given by

$$\kappa = \frac{x'y'' - x''y'}{\Big((x')^2 + (y')^2\Big)^{3/2}} \tag{2.20}$$

where x' and x'' denote first and second order derivatives with respect to the parameter t. Let $\big(x(t), y(t)\big)$ be the parametric form of the implicit polynomial curve. This form of the implicit polynomial curve is not directly available; however, to make use of equation (2.20) we only need to derive expressions for the derivatives. The tangent vector to the IP curve, $\mathbf{T}(t) = \big(x'(t), y'(t)\big)$, can be found up to a scale factor as follows: Differentiate $f\big(x(t), y(t)\big) = 0$ with respect to t and use the chain rule to get

$$\frac{\partial f}{\partial x}x' + \frac{\partial f}{\partial y}y' = 0$$

The above equation will hold for all t if

$$\begin{aligned} x' &= -\frac{\partial f}{\partial y} \\ y' &= \frac{\partial f}{\partial x} \end{aligned} \tag{2.21}$$

The other piece of information necessary for the computation of equation (2.20) is $\mathbf{T}'(t) = \big(x''(t), y''(t)\big)$. Using the chain rule once more, we obtain

$$\mathbf{T}' = \frac{d\mathbf{T}}{dt} = \frac{\partial \mathbf{T}}{\partial x}x' + \frac{\partial \mathbf{T}}{\partial y}y' \tag{2.22}$$

Using $x' = -\frac{\partial f}{\partial y}$ and $y' = \frac{\partial f}{\partial x}$, we find

$$\begin{aligned}\frac{\partial \mathbf{T}}{\partial x} &= \Big(-\frac{\partial^2 f}{\partial y \partial x}, \frac{\partial^2 f}{\partial x^2} \Big) \\ \frac{\partial \mathbf{T}}{\partial y} &= \Big(-\frac{\partial^2 f}{\partial y^2}, \frac{\partial^2 f}{\partial x \partial y} \Big)\end{aligned}$$

Substitute these results in equation (2.22) to obtain

$$\mathbf{T}' = \Big(\frac{\partial^2 f}{\partial y \partial x}\frac{\partial f}{\partial y} - \frac{\partial^2 f}{\partial y^2}\frac{\partial f}{\partial x}, \ -\frac{\partial^2 f}{\partial x^2}\frac{\partial f}{\partial y} + \frac{\partial^2 f}{\partial x \partial y}\frac{\partial f}{\partial x} \Big)$$

Since $\mathbf{T}'(t) = \big(x''(t), y''(t)\big)$, we observe that

$$\begin{aligned}x'' &= \frac{\partial^2 f}{\partial y \partial x}\frac{\partial f}{\partial y} - \frac{\partial^2 f}{\partial y^2}\frac{\partial f}{\partial x} \\ y'' &= -\frac{\partial^2 f}{\partial x^2}\frac{\partial f}{\partial y} + \frac{\partial^2 f}{\partial x \partial y}\frac{\partial f}{\partial x}\end{aligned} \tag{2.23}$$

Substituting the results in equations (2.21) and (2.23) in equation (2.20), we get

$$\kappa = \frac{\frac{\partial^2 f}{\partial x^2}\big(\frac{\partial f}{\partial y}\big)^2 - 2\frac{\partial^2 f}{\partial x \partial y}\frac{\partial f}{\partial x}\frac{\partial f}{\partial y} + \frac{\partial^2 f}{\partial y^2}\big(\frac{\partial f}{\partial x}\big)^2}{\Big(\big(\frac{\partial f}{\partial x}\big)^2 + \big(\frac{\partial f}{\partial y}\big)^2\Big)^{3/2}} \tag{2.24}$$

κ is measured from the input data at every point and we want to force the right-hand side of equation (2.24) to equal these values at the data points. The right-hand side of equation (2.24) is nonlinear and can not be incorporated in our linear framework as it is, but an approximation to it can be. Recall that in *gradient-one fitting* we compute the unit local normals $\mathbf{n} = (n_x, n_y)$ to the data set and ask that the gradient of the polynomial be in these directions with magnitude one. Thus we are forcing f to have $\frac{\partial f}{\partial x} = n_x$, $\frac{\partial f}{\partial y} = n_y$ and $\sqrt{\big(\frac{\partial f}{\partial x}\big)^2 + \big(\frac{\partial f}{\partial y}\big)^2} = 1$ as closely as possible. If f is of sufficient degree to provide a good approximation, then these conditions will roughly hold and we can substitute these

measurements from the data in to equation (2.24) as approximations to the real derivatives of f which yields

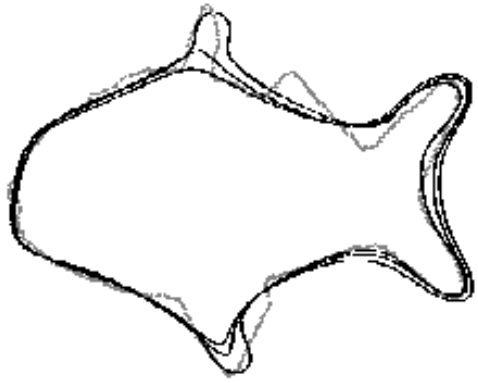
$$\kappa \approx \frac{\partial^2 f}{\partial x^2} n_y^2 - 2\frac{\partial^2 f}{\partial x \partial y} n_x n_y + \frac{\partial^2 f}{\partial y^2} n_x^2. \tag{2.25}$$

Equation (2.25) provides constraints on the second derivatives of f based on curvature information in a linear manner which can be easily incorporated into the *gradient-one fitting* framework. This can be accomplished by defining monomial and scatter matrices for the higher degree fitting terms as in equations (2.13) and (2.14) and incorporating these in the overall design and scatter matrices in equations (2.15) and (2.16).

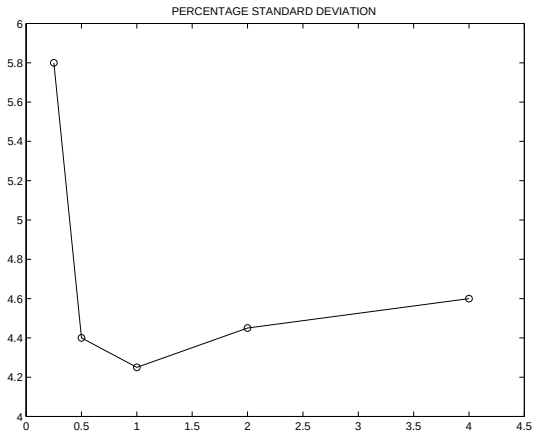
2.4 Invariance Properties of Gradient-one Fitting

The question of the invariance of the fitting algorithm to Euclidean transformations of the data is important to insure repeatability of the results under such transformations. In this section, we show that the *gradient-one fitting* is rotation and translation invariant. When a Euclidean transformation is applied to the data set, vector $\mathbf{y}$ of monomials is transformed as $\mathbf{y}' = \mathbf{V}(\theta, t_x, t_y)\mathbf{y}$, where the $p \times p$ transformation matrix $\mathbf{V}$ is a function of θ, the applied rotation angle, and (t_x, t_y), the applied translation. The zero set of the implicit polynomial is defined as

$$\mathbf{a}^T \mathbf{y} = 0 \ , \tag{2.26}$$



(a)



(b)

Figure 2.4: (a) 6^{th} degree IP fits with the *gradient-one fitting* algorithm for three different values of μ. (b) The average percentage standard deviation of the coefficients with respect to their average norm under colored noise (see Section 3.6.1) for increasing values of μ.

and the zero set after the coordinate transformation is given by

$$\mathbf{a'}^T\mathbf{y'} = 0. \tag{2.27}$$

After the substitution $\mathbf{y} = \mathbf{V}^{-1}\mathbf{y'}$ in equation (2.26), we have

$$\mathbf{a}^T\mathbf{V}^{-1}\mathbf{y'} = 0.$$

Comparing this result with equation (2.27), we observe that

$$\mathbf{a'}^T = \mathbf{a}^T\mathbf{V}^{-1}.$$

Taking the transpose of this equation we obtain

$$\begin{aligned}\mathbf{a'} &= (\mathbf{a}^T\mathbf{V}^{-1})^T \\ &= (\mathbf{V}^{-1})^T\mathbf{a} \\ &= (\mathbf{V}^T)^{-1}\mathbf{a}\end{aligned}$$

From this equation, we see that the transformed coefficients are $\mathbf{a'} = (\mathbf{V}^T)^{-1}\mathbf{a}$. Substituting for $\mathbf{a}$ from equation (2.19) gives

$$\begin{aligned}\mathbf{a'} &= (\mathbf{V}^T)^{-1}\mathbf{a} \\ &= (\mathbf{V}^T)^{-1}\Big(\mathbf{S} + \mu(\mathbf{S_n} + \mathbf{S_t})\Big)^{-1}\mathbf{G_n} \\ &= (\mathbf{V}^T)^{-1}\Big(\mathbf{S} + \mu(\mathbf{S_n} + \mathbf{S_t})\Big)^{-1}\underbrace{\mathbf{V}^{-1}\mathbf{V}}_{\mathbf{I}}\mathbf{G_n}.\end{aligned}$$

Now use the matrix identity $(\mathbf{ABC})^{-1} = \mathbf{C}^{-1}\mathbf{B}^{-1}\mathbf{A}^{-1}$ to obtain

$$\mathbf{a}' = \Big(\mathbf{V}\mathbf{S}\mathbf{V}^T + \mu\big(\mathbf{V}\mathbf{S}_\mathbf{n}\mathbf{V}^T + \mathbf{V}\mathbf{S}_\mathbf{t}\mathbf{V}^T\big)\Big)^{-1}\mathbf{V}\mathbf{G}_\mathbf{n} \tag{2.28}$$

From Section 2.1 equation (2.4), the monomial scatter matrix $\mathbf{S}$ is a summation of matrices of the form $\mathbf{y}\mathbf{y}^T$, so it transforms as

$$\begin{aligned}\mathbf{S}' &= \sum \mathbf{y}'\mathbf{y}'^T = \sum \mathbf{V}\mathbf{y}\mathbf{y}^T\mathbf{V}^T = \mathbf{V}\Big(\sum \mathbf{y}\mathbf{y}^T\Big)\mathbf{V}^T \\ &= \mathbf{V}\mathbf{S}\mathbf{V}^T\end{aligned}$$

Similarly, the matrices containing the information on the normals and tangents transform as $\mathbf{S}'_\mathbf{t} = \mathbf{V}\mathbf{S}_\mathbf{t}\mathbf{V}^T$ and $\mathbf{S}'_\mathbf{n} = \mathbf{V}\mathbf{S}_\mathbf{n}\mathbf{V}^T$ using the fact that the computations of the normal and tangent directions are Euclidean covariant. Thus, the overall scatter matrix $\mathcal{S}$ in equation (2.19) transforms as

$$\mathcal{S}' = \mathbf{V}\mathcal{S}\mathbf{V}^T \tag{2.29}$$

$\mathbf{G}_\mathbf{n}$ transforms like $\mathbf{y}$ since it is a weighted sum of monomials, thus $\mathbf{G}'_\mathbf{n} = \mathbf{V}\mathbf{G}_\mathbf{n}$. Using these observations equation (2.28) simplifies to

$$\mathbf{a}' = \Big(\mathbf{S}' + \mu(\mathbf{S}'_\mathbf{n} + \mathbf{S}'_\mathbf{t})\Big)^{-1}\mathbf{G}'_\mathbf{n}$$

which is exactly the result that would be obtained from fitting to the transformed data using equation (2.19). Consequently, *gradient-one fitting* is Euclidean invariant. Notice that the Euclidean properties of $\mathbf{V}(\theta, t_x, t_y)$ is used only for the computation of the normal components. This leads to a possible extension to affine invariant fitting if a method to

robustly compute affine invariant normals is developed.

Chapter 3

RIDGE REGRESSION REGULARIZATION OF IMPLICIT POLYNOMIAL CURVE FITTING

3.1 Unstable Subspaces

Although, local stability of the zero set around the data is excellent with *gradient-one fitting*, there is still significant room for improvement in the stability of the coefficients of the implicit polynomial and the global behavior of the polynomial. Coefficient vectors in certain subspaces of the coefficient space may produce very similar zero sets around the data set while differing elsewhere. As an example, assume that the data is a set of aligned points along $x - y = 0$, and that we are trying to fit a full conic. If we do the fit many times subject to small perturbations of the data, we can observe that the resulting coefficient

vectors span a 3 dimensional subspace containing the solutions $x(x-y)$ and $y(x-y)$ as well as $x-y$. This is a consequence of the fact that each of these three solutions and all of their linear combinations fit the original data set equally well. The global instability of polynomials is also evident in the extra pieces of the zero set that lie away from the data, see Figure 3.1. Indeed, these pieces are extremely sensitive to small perturbations in the data even though the zero set around the data is stable. Of course, it should be kept in mind that the examples obtained with *gradient-one fitting* shown in Figure 3.1 are still much more stable under data perturbations compared to classical least squares fitting examples in Figure 2.1 in the sense that the pieces of the zero set curve that actually represent the input data are robust to noise.

We now examine global instability problems. The overall scatter matrix $\mathcal{S}$, defined in equation (2.16), is symmetric non-negative definite since it is a sum of scatter matrices, and thus can be written as

$$\mathcal{S} = \mathbf{U}^T \mathbf{\Lambda} \mathbf{U}$$

where $\mathbf{U}$ is a rotation in the coefficient space. The elements of $\mathbf{\Lambda}$ and the columns of $\mathbf{U}$ are the eigenvalues and eigenvectors of $\mathcal{S}$, respectively. If there is exact collinearity in the data, $\mathcal{S}$ will be singular and one or more eigenvalues will be 0. A much more common problem is near singularity where some eigenvalues of are very small compared to others and $\mathcal{S}$ has a very large condition number. Eigenvectors of $\mathcal{S}$ associated with the very small eigenvalues do not contribute to the polynomial significantly around the dataset. Such vectors multiplied with large scalars get added into the solution in pursuit of slightly better solutions because Least Squares Estimation produces the coefficient vector $\mathbf{a}$ that globally

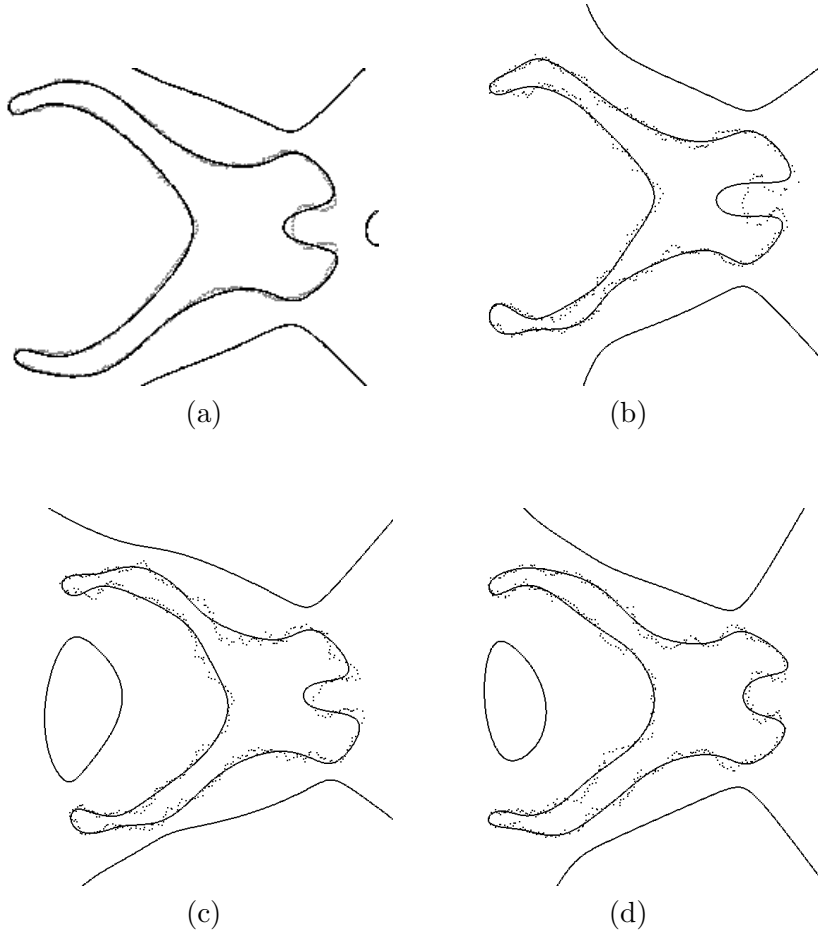


Figure 3.1: With *gradient-one fitting* pieces of the implicit polynomial zero set that represent the data are stable with respect to perturbations to the input data. However, global instability of the fitted polynomials is apparent in the other pieces of the zero set. (a) is the original shape and the implicit polynomial curve fitted to it, (b)-(d) are data sets perturbed with white noise that has standard deviation equal to % 5 of the input shape size. Zero set curves of the fitted polynomials have been superimposed on the data shapes. Notice that the piece of the zero set between the handles of the pliers is present in only two of the fits.

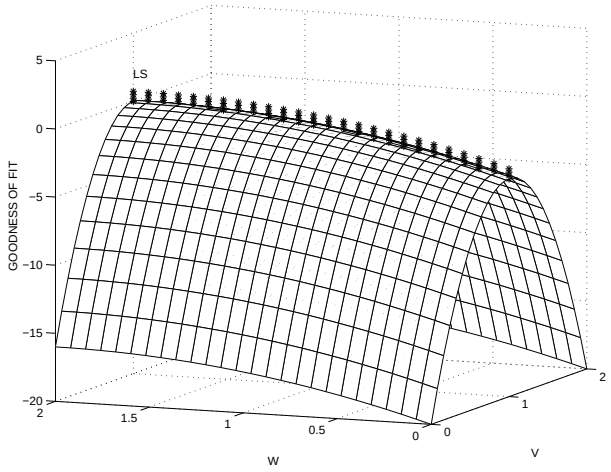


Figure 3.2: Graph of an error function of two variables; here V is the stable variable while W is relatively unstable. The unstable ridge is marked by a heavier line.

minimizes the error function in equation (2.10). This results in very large variances for coefficients in the subspaces spanned by these eigenvectors. In Figure 3.2, the graph of a goodness of fit function in two variables is shown. Notice that the function drops off relatively steeply with the stable variable V, but changes only very slowly with unstable W. Thus, the solution of LSE which seeks the highest point on the graph, marked LS in the Figure 3.2, moves along the unstable *ridge* (heavy line in Figure 3.2) with the addition of small amounts of noise to the data. Consequently, the variance of the variable W due to noise is much larger than that of V. What we desire is that scalars multiplying such eigenvectors be pushed to zero rather than up to unstably-canceling infinities. This requires modifying LSE as we explain next.

3.2 Ridge Regression

As stated in Section 3.1, we would like variables that do not contribute significantly to the fit to be forced to attain values as close to zero as possible while other variables are

effectively unchanged. Since the solution has to move along the ridge shown in Figure 3.2, the stabilization of the least square is known as *Ridge Regression* (RR) [30, 29, 85]. The method of *ridge regression* improves the condition number of $\mathcal{S}$. The modified coefficient vector $\mathbf{a}_\kappa$ is obtained by

$$\mathbf{a}_\kappa = \mu(\mathcal{S} + \kappa\mathbf{D})^{-1}\mathbf{G_n} \tag{3.1}$$

where $\mathbf{D}$ is a positive definite and symmetric matrix and κ is the ridge regression parameter. Although $\mathbf{D}$ could in principle be chosen as any positive definite matrix, we restrict ourselves to the simple case where $\mathbf{D}$ is a diagonal matrix. The addition of a diagonal matrix $\mathbf{D}$ to the scatter matrix $\mathcal{S}$ has the effect of adding a bias which produces coefficient vectors with smaller norms, i.e., smaller $\| \mathbf{a}_\kappa \|$. In this sense, *ridge regression* is analogous to weight decay regularization used in training Neural Networks [6]. We choose the elements of $\mathbf{D}$ to be functions of the sum of squared values of the monomials. In other words, $\mathbf{D}$ is a function of the main diagonal of $\mathcal{S}$. A specific choice for the elements of $\mathbf{D}$ that meets the rotational invariance requirements and which has a desired limiting behavior is proposed and explained in further detail in Section 3.3. Notice that as κ is increased, $\mathcal{S} + \kappa\mathbf{D}$ approaches $\mathbf{D}$, and $\mathbf{a}_\kappa$ approaches the limit

$$a_\infty = \lim_{\kappa\to\infty} \frac{\mu}{\kappa}\mathbf{D}^{-1}\mathbf{G_n}.$$

We examine the limiting behavior of $\mathbf{G_n}$ in Section 3.4. Equation (3.1) biases the solution closer to $\mathbf{G_n}$ with increasing κ values. Lets return to the example given in Section 3.1. If

the data set consists of m points aligned around the line $x - y = 0$ with noise, we have

$$\mathbf{G_n} = \begin{bmatrix} 0 & m & -m & 2\bar{x} & \bar{y} - \bar{x} & -2\bar{y} \end{bmatrix}$$

For simplicity, we can assume for now that $\mathbf{D}$ is the identity matrix. Thus, if the data set is centered at the origin, the solution obtained by *ridge regression* is biased towards

$$\begin{bmatrix} 0 & 1 & -1 & 0 & 0 & 0 \end{bmatrix}$$

which is the equation of the line $x - y = 0$ we are searching for. It can easily be shown [85] that

$$\mathbf{a}_\kappa = \mathbf{U}\boldsymbol{\Delta}\mathbf{U}^T\mathbf{a} \tag{3.2}$$

where $\boldsymbol{\Delta}$ is a diagonal matrix of shrinkage factors, $\mathbf{U}$ is as defined in Section 3.1 and $\mathbf{a}$ is the original least squares estimator. Notice that *ridge regression* is independent of the exact manner in which $\mathbf{a}$ is obtained. We base our *ridge regression* results on the *gradient-one* estimator; however any other least squares estimator such as the $3L$ estimator could be substituted in equation (3.2). For example, this universal nature of the *ridge regression* method has allowed its successful application to fitting implicit polynomial surfaces to sets of points in $3D$. *Ridge regression* modifies the estimator by first rotating it to obtain uncorrelated components, shrinking each component by some amount and finally restoring the original coordinate system by another rotation. The crucial point is the amount of shrinkage applied to each component. If $\mathbf{D}$ in equation (3.1) were chosen to be the identity

matrix, then it is shown in [85] that

$$\mathbf{\Delta} = \begin{bmatrix} \delta_1 & 0 & \dots & 0 \\ 0 & \delta_2 & \dots & 0 \\ & & \dots & \\ 0 & 0 & \dots & \delta_p \end{bmatrix}, \ \delta_i = \frac{\lambda_i}{\lambda_i + \kappa} \tag{3.3}$$

where κ is the ridge regression parameter and λ_i are the eigenvalues of $\mathcal{S}$, i.e., the diagonal components of $\mathbf{\Lambda}$. The shrinkage factor δ_i multiplies the *i'th* eigenvalue of $\mathcal{S}^{-1}$ which is λ_i^{-1}, thus the *i'th* eigenvector is shrunk by a factor of δ_i in the solution. Since the eigenvectors related to the very small eigenvalues of $\mathcal{S}$ are unstable, we would like to shrink them while leaving other eigenvectors largely unaffected. With equation (3.1), this is accomplished as shown by equation (3.3). Consider a simple case similar to the one depicted in Figure 3.2 where there are two variables one of which is significantly less stable than the other. This would result in an ill-conditioned matrix $\mathcal{S}$ with eigenvalues,e.g., $\lambda_1 = 1$ and $\lambda_2 = 10^{-4}$, and $\mathcal{S}^{-1}$ having eigenvalues 1 and 10^4 which are the reciprocals of λ_1 and λ_2, respectively. If we select $\kappa = 10^{-3}$ we obtain the shrinkage factors $\delta_1 = 0.999$ and $\delta_2 = 0.0909$. Thus, the eigenvalues of $(\mathcal{S} + \kappa\mathbf{I})^{-1}$ will be $1 \times 0.999 = 0.999$ and $10^4 \times 0.0909 = 909$. Notice that the stable eigenvector corresponding to the larger eigenvalue of $\mathcal{S}$ (equivalently the smaller eigenvalue of $\mathcal{S}^{-1}$) remains relatively unchanged whereas the condition number is improved from $\frac{10^4}{1} = 10^4$ to $\frac{909}{0.999} \approx 909$, an approximately 11 fold improvement. We address the question of choosing the value of κ in Section 3.5.

Figure 3.3(c) and (f) shows fits of degrees 6 and 8 obtained by *gradient-one fitting* regularized by *ridge regression*. Comparing these results with the results from standard *gradient-one fitting* shown in Figure 3.3(b) and (e), we observe two important properties

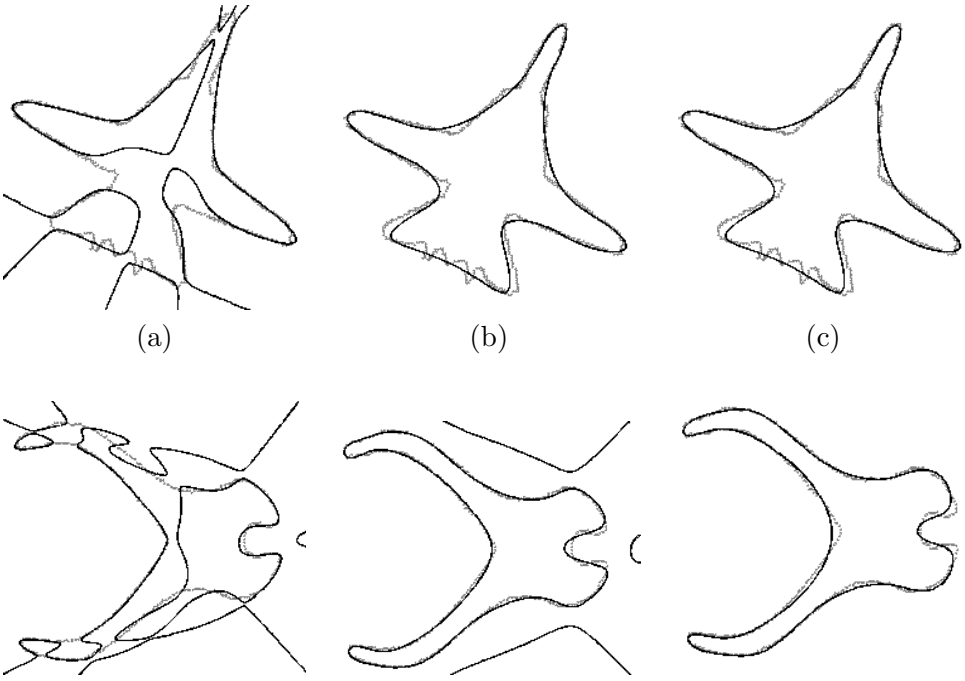


Figure 3.3: (a) and (d): Classical Least Squares Fitting Algorithm. (b) and (e): *Gradient-one fitting* Algorithm. (c) and (f): *Gradient-one* fitting regularized by *ridge regression*. Degree 6 and 8 are used for the airplane and pliers shapes, respectively. Notice that there are no extra components in (c) and (f). The values of κ were chosen interactively by the user.

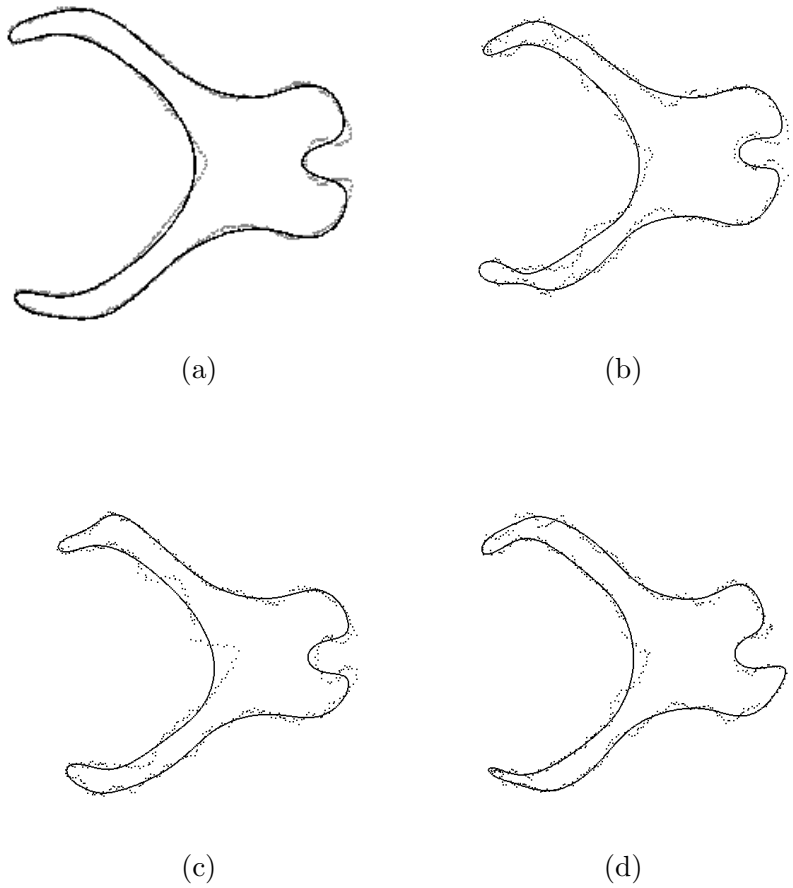


Figure 3.4: Modifying *gradient-one fitting* with *ridge regression* improves its stability with respect to perturbations of the input data. As in Figure 3.1 (a) is the original shape and white noise with standard deviation % 5 of the shape size has been added to (b)-(d). The fitted implicit polynomial curves are superimposed on the data sets in the figure. The value of κ is fixed for all 4 examples, but was chosen manually by the user.

of *ridge regression* regularization: (i) the extra pieces of the zero set in the fit to the pliers shape is gone and both fits are bounded, and (ii) the smoothing introduced around the data set is negligible. These properties follow from the fact that stable eigenvectors in the solution are left largely unaffected by *ridge regression* while unstable ones are shrunk to insignificant values. Comparing Figure 3.4 with Figure 3.1 it can be observed that the artifacts of global instability evident with standard *gradient-one fitting* do not exist in the results obtained by modifying the *gradient-one* estimator with *ridge regression*. In the

examples shown in Figure 3.3 and Figure 3.4 the value of κ was chosen manually. The effect of increasing the parameter κ from 0 to higher values is shown in Figure 3.5. Notice that the unbounded pieces that are close to the data in fitting with no *ridge regression*, $\kappa = 0$, start to move away with increasing κ. Actually, these pieces totally disappear and the polynomial zero set becomes bounded. Recall that in Section 3.1 it was pointed out that unboundedness and extra pieces of the zero set were symptoms of the instability in fitting. Thus, *ridge regression* achieves the goal of getting rid of these effects. In Section 3.6, we present results of experiments that show the quantitative improvement in stability obtained by *ridge regression* which we believe is strongly linked to the qualitative improvements summarized above. We also show in Section 3.4 that a fit to data for a closed shape will converge to a bounded IP curve as κ goes to infinity.

3.3 Rotational Invariance of Ridge Regression

In Section 2.4 *gradient-one fitting* was shown to be invariant under Euclidean transformations. Now we will show that *ridge regression* when used to modify *gradient-one fitting* or any other Euclidean invariant fitting method, can be designed so that these invariance properties are preserved. The matrix $\mathbf{D}$ must be of a special form to preserve the rotational invariance property in *ridge regression*. If we apply the same substitutions to equation (3.1) as those applied to equation (2.19) in Section 2.4, we obtain

$$\begin{aligned}
\mathbf{a}'_\kappa &= (\mathbf{V}^T)^{-1}\mathbf{a}_\kappa \\
&= (\mathbf{V}^T)^{-1}\Big(\mathcal{S} + \kappa\mathbf{D}\Big)^{-1}\mathbf{V}^{-1}\mathbf{V}\mathbf{G}_\mathbf{n} \\
&= \Big(\mathbf{V}\mathcal{S}\mathbf{V}^T + \mathbf{V}\mathbf{D}\mathbf{V}^T\Big)^{-1}\mathbf{V}\mathbf{G}_\mathbf{n}
\end{aligned}$$

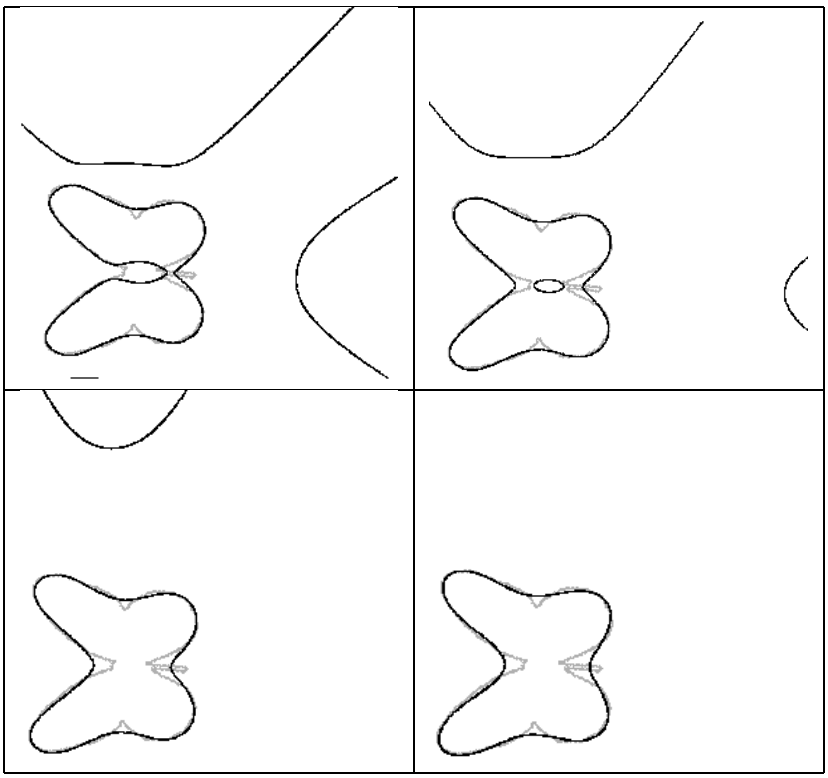


Figure 3.5: 6^{th} degree IP fits with the *gradient-one fitting* algorithm and *ridge regression* for increasing values of parameter κ.

Ridge regression fitting will be Euclidean invariant if

$$\mathbf{a}'_{\kappa} = \left(\mathcal{S}' + \mathbf{D}'\right)^{-1} \mathbf{G_n}'$$

which is exactly (3.1) in the transformed reference system. According to equation (2.29) the scatter matrix $\mathcal{S}$ transforms as $\mathbf{V}\mathcal{S}\mathbf{V}^T$ under an Euclidean transformation $\mathbf{V}(\theta, t_x, t_y)$. Thus it is necessary to have

$$\mathbf{V}\mathbf{D}\mathbf{V}^T = \mathbf{D}'$$

for *ridge regression* to preserve the Euclidean invariance properties of fitting. This means that the invariance of the algorithm to Euclidean transformations dictates the structure of the matrix $\mathbf{D}$. It is known that if the Euclidean transformation is reduced to a pure rotation, $\mathbf{V}$ can be decomposed as $\mathbf{V} = \mathbf{B}^{-\frac{1}{2}}\mathbf{R}\mathbf{B}^{\frac{1}{2}}$ [79]. Dealing with rotation invariance alone is justified because the centering of the data in the standardization step before fitting takes care of arbitrary translations. $\mathbf{B}$ is the diagonal matrix of binomial coefficients whose v'th element along the diagonal is

$$\mathbf{B}_{vv} = \frac{(i+j)}{!i!j!} \ , \ v = j + \frac{(i+j+1)(i+j)}{2}. \tag{3.4}$$

$\mathbf{R}$ is a block diagonal rotation matrix. Upon substitution of $\mathbf{V}$ in $\mathbf{V}\mathbf{D}\mathbf{V}^T = \mathbf{D}'$, we have

$$\mathbf{B}^{-1/2}\mathbf{R}\mathbf{B}^{1/2}\mathbf{D}\mathbf{B}^{1/2}\mathbf{R}^T\mathbf{B}^{-1/2} = \mathbf{D}'.$$

Multiplying both sides of this equation by $\mathbf{B}^{1/2}$ from the left and the right, we obtain

$$\mathbf{R}\mathbf{B}^{1/2}\mathbf{D}\mathbf{B}^{1/2}\mathbf{R}^T = \mathbf{B}^{1/2}\mathbf{D}'\mathbf{B}^{1/2}$$

This simplifies to

$$\mathbf{R}\mathbf{B}\mathbf{D}\mathbf{R}^T = \mathbf{B}\mathbf{D}'$$

since $\mathbf{D}$ is diagonal as well as $\mathbf{B}$. It is sufficient for satisfying the previous equation that $\mathbf{D}$ is block by block, the inverse of $\mathbf{B}$. Therefore, a $\mathbf{D}$ sufficient for rotation invariance is:

$$\mathbf{D}_{vv} = \alpha_{i+j}\frac{i!j!}{(i+j)!}\ ,\ v = j + \frac{(i+j+1)(i+j)}{2} \tag{3.5}$$

where α_{i+j} is a parameter for the $i+j$*'th* block. There are $d+1$ parameters when dealing with d'th degree polynomials, corresponding to the $d+1$ blocks and forms. We are free to set these parameters in a Euclidean invariant way as we will discuss next.

In its simplest form ridge regression uses the identity matrix for $\mathbf{D}$. Solving $\mathbf{a}_\kappa = \mu(\mathcal{S} + \kappa\mathbf{I})^{-1}\mathbf{G_n}$ is equivalent to minimizing

$$E(\mathbf{a})_{rr} = E(\mathbf{a}) + \kappa \parallel a \parallel^2$$

where $E(\mathbf{a})$ is the original error function being minimized such as the gradient-1 fitting error function, equation (2.10). To prove this recall the vector form of the gradient-1 fitting error function given by equation (2.18):

$$E_{\text{grad}} = \mathbf{a}^T\mathcal{M}\mathcal{M}^T\mathbf{a} - 2\mathbf{a}^T\mathcal{M}\mathbf{b} + \mathbf{b}^T\mathbf{b}$$

where $\mathcal{M}\mathcal{M}^T = \mathcal{S}$ and $\mathcal{M}\mathbf{b} = \mathbf{G_n}$. Then

$$\begin{aligned} E(\mathbf{a})_{rr} &= E_{\text{grad}}(\mathbf{a}) + \kappa \parallel a \parallel^2 \\ &= \mathbf{a}^T\mathcal{M}\mathcal{M}^T\mathbf{a} - 2\mathbf{a}^T\mathcal{M}\mathbf{b} + \mathbf{b}^T\mathbf{b} + \kappa \parallel a \parallel^2 \\ &= \mathbf{a}^T\mathcal{M}\mathcal{M}^T\mathbf{a} + \kappa\mathbf{a^T I a} - 2\mathbf{a}^T\mathcal{M}\mathbf{b} + \mathbf{b}^T\mathbf{b} \\ &= \mathbf{a}^T\Big(\mathcal{M}\mathcal{M}^T + \kappa\mathbf{I}\Big)\mathbf{a} - 2\mathbf{a}^T\mathcal{M}\mathbf{b} + \mathbf{b}^T\mathbf{b} \end{aligned}$$

which only modifies $\mathcal{S}$ to $\mathcal{S} + \kappa\mathbf{I}$. Consequently, the solution is modified from $\mathcal{S}^{-1}\mathbf{G_n}$ to $(\mathcal{S}+\kappa\mathbf{I})^{-1}\mathbf{G_n}$. In other words, ridge regression regularization adds a term to the error that is proportional to the squared length of the parameter vector; thus, favoring shorter coefficient vectors. This is very closely related to *weight decay* regularization used to overcome problems of *over fitting* in iterative optimization schemes [6]. Note that this is not Euclidean invariant. If we set all α to 1 then $\mathbf{D} = \mathbf{B}^{-1}$ and we are minimizing

$$E(\mathbf{a})_{rr} = E(\mathbf{a}) + \kappa\mathbf{a^T B^{-1} a}$$

which can be shown with the same arguments as above using the fact $\mathbf{B}$ is diagonal. In this case, the weighted length of the coefficient vector, which is Euclidean invariant, is added on to the error function. This choice of $\mathbf{D}$ is Euclidean invariant; however, the results obtained from ridge regression are improved with the following choice of $\mathbf{D}$. Using the binomial coefficients once more, we set each of these parameters to the invariantly weighted sum of the diagonal elements of $\mathcal{S}$ associated with the $i+j$'th form. For the data set $\Big\{(x_k, y_k)\Big\}_{k=1}^{m}$, α_{i+j} is the weighted total scattering of the terms in the *i+j'th* degree form:

$$\alpha_{i+j} = \sum_{r,l\geq 0; r+l=i+j} \frac{(r+l)!}{r!l!} \sum_{k=1}^{m} x_k^{2r} y_k^{2l} \tag{3.6}$$

This choice of α_{i+j} is again Euclidean invariant and is equivalent to solving

$$E(\mathbf{a})_{rr} = E(\mathbf{a}) + \kappa \mathbf{a}^{\mathbf{T}} \mathbf{D} \mathbf{a}$$

We have found that this choice brings significant improvements in power of shape representation over simply setting $\alpha_{i+j} = 1$ for all i, j. In other words, this choice of $\mathbf{D}$ yields a better tradeoff between the desired effect of removing extra pieces of the zero set of the polynomial and the undesirable effect of smoothing the zero set. The elements of the diagonal matrix $\mathbf{D}$ can be obtained by combining equation (3.5) and equation (3.6).

In problems where invariance is not of concern, Principal Component Methods [35] which do not provide any freedom in the choice of $\mathbf{D}$, can be used alternatively. Since invariance is a major concern for us, we choose to work in the more general framework of *ridge regression*.

3.4 Boundedness Properties and Limiting Behavior of Ridge Regression

The limit of the solution of (3.1) as κ goes to infinity is

$$\mathbf{a}_\infty = \mathbf{D}^{-1} \mathbf{G}_\mathbf{n} \tag{3.7}$$

up to a scale factor. It turns out that the polynomial specified by $\mathbf{a}_\infty$ has important properties. Indeed when the data shape is closed and the degree of the fitted implicit polynomial curve is even, the IP curve converges to the curve given by $\mathbf{a}_\infty$ which is always bounded, as the example in Figure 3.6. The proof that follows is based on the divergence

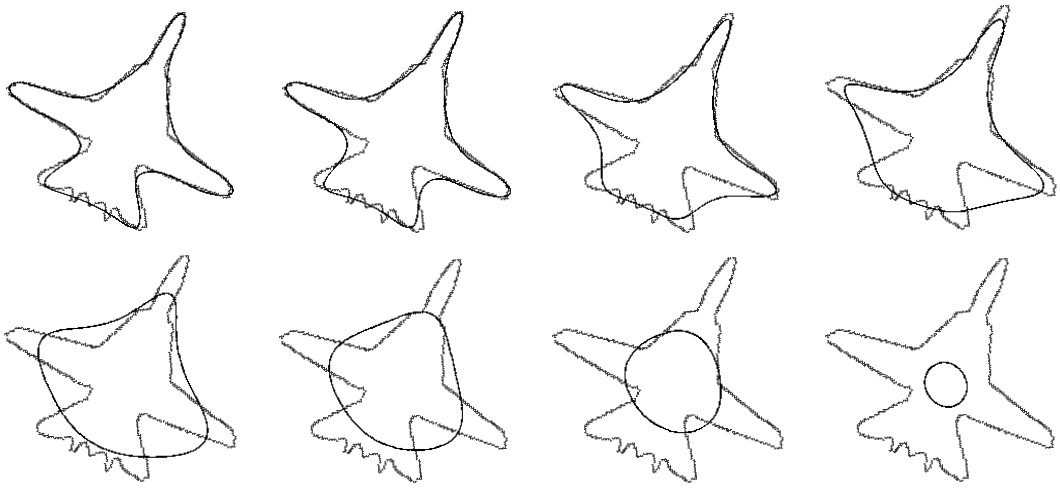


Figure 3.6: Left, 6^{th} degree polynomial fits with the *gradient-one fitting* algorithm and *ridge regression* for increasing values of parameter κ ($\kappa = 0$, $\kappa = 0.0001$, 0.001, 0.01, 0.05, 0.5, 2.0 and 32, respectively). We can observe that the fitted zero set is becoming smoother and converging to an approximately circular shape.

theorem for closed 2D curves. We can substitute for $\mathbf{D}$ and $\mathbf{G_n}$ in equation (3.7) from equation (3.5) and equation (2.12), respectively. The components a_{ij} of vector $\mathbf{a}_\infty$ which are given by the summation in the matrix multiplication in equation (3.7) can be approximated as an integral along a contour C when the data shape is closed and the sampling of the curve is not too coarse:

$$a_{ij} = \frac{(i+j)!}{\alpha_{i+j} i! j!} \oint_C \mathbf{n}^T \nabla (x^i y^j)$$

where $\mathbf{n}$ is the normal vector to the curve. Since C is a closed contour, by applying the divergence theorem and using the vector identity $\nabla \cdot \nabla g = \nabla^2 g$ we obtain

$$\begin{aligned} a_{ij} &= \frac{(i+j)!}{\alpha_{i+j} i! j!} \int \int_R \nabla^2 (x^i y^j) dx dy \\ &= \frac{(i+j)!}{\alpha_{i+j} i! j!} \int \int_R i(i-1) x^{i-2} y^j + j(j-1) x^i y^{j-2} dx dy \end{aligned}$$

where ∇^2 is the Laplacian operator and the R is the region bounded by C. Using (1.3), and introducing the monomial vector $\tilde{\mathbf{y}}$, the zero set of $\mathbf{a}_\infty$ is

$$\mathbf{a}_\infty^T \tilde{\mathbf{y}} = \sum_{0 \le i+j \le d} a_{ij} \tilde{x}^i \tilde{y}^j = 0$$

To prove that the zero set of this polynomial is always bounded, it is enough to show that the leading form of this polynomial is always strictly positive [38]. Lets denote the leading form, the form of degree d, by h_d. By using the two previous equations we find

$$h_d = \frac{1}{\alpha_d} \Big(\sum_{i,j \ge 0, i+j=d} \frac{(i+j)!}{i!j!} \tilde{x}^i \tilde{y}^j \int\int_R i(i-1)x^{i-2}y^j + j(j-1)x^i y^{j-2} dxdy \Big).$$

Using $i + j = d$ and after some rearranging we obtain

$$\begin{aligned} h_d = \frac{d(d-1)}{\alpha_d} \Big(& \int\int_R \sum_{i=0}^{d} \frac{(d-2)!i(i-1)(x\tilde{x})^{i-2}(y\tilde{y})^{d-i}(\tilde{x})^2}{i!(d-i)!} dxdy + \\ & \int\int_R \sum_{i=0}^{d} \frac{(d-2)!(d-i)(d-i-1)(x\tilde{x})^{i}(y\tilde{y})^{d-i-2}(\tilde{y})^2}{i!(d-i)!} dxdy \Big). \end{aligned}$$

Notice that the in the first summation the terms corresponding to $i = 0, 1$ are 0 and in the second summation the terms corresponding to $i = d, d-1$ are 0. Using this observation and making the substitution $e = d - 2$ in both summations and the substitution $u = i - 2$ only in the first summation we get

$$\begin{aligned} h_d = \frac{d(d-1)}{\alpha_d} \Big(& \tilde{x}^2 \int\int_R \sum_{u=0}^{e} \frac{e!}{u!(e-u)!} (x\tilde{x})^u (y\tilde{y})^{e-u} \, dxdy + \\ & \tilde{y}^2 \int\int_R \sum_{i=0}^{e} \frac{e!}{i!(e-i)!} (x\tilde{x})^i (y\tilde{y})^{e-i} \, dxdy \Big). \end{aligned} \tag{3.8}$$

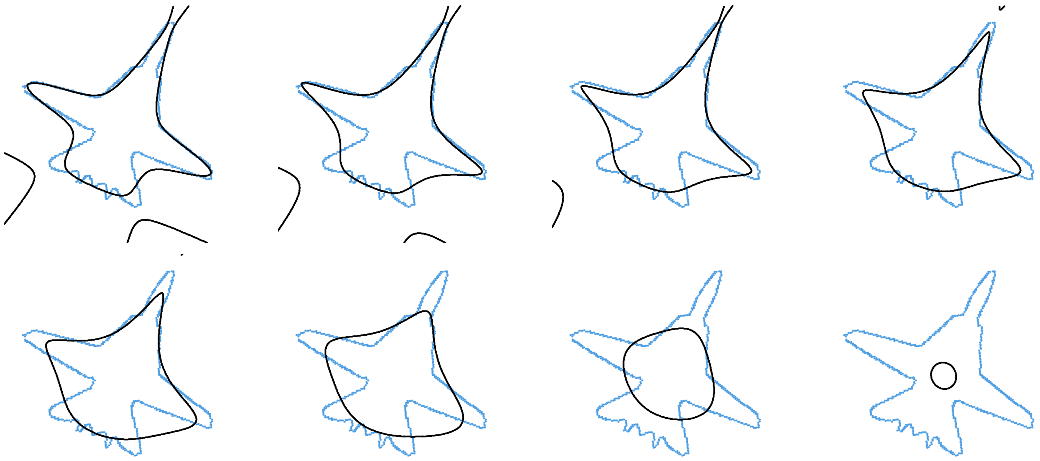


Figure 3.7: The limiting implicit polynomial zero set for an odd degree (*5'th*) polynomial fitted to a closed data shape as $\kappa \rightarrow \infty$.

Making use of the binomial expansion

$$(a+b)^n = \sum_{i=0}^{n} \frac{n!}{i!(n-i)!} a^i b^{n-i}$$

we simplify equation (3.8) to

$$h_d = \frac{d(d-1)}{\alpha_d}(\tilde{x}^2 + \tilde{y}^2) \int\int_R (x\tilde{x} + y\tilde{y})^{d-2} dxdy. \tag{3.9}$$

The leading form given by equation (3.9) is always positive for d even. As an important consequence of this proof, it is always possible to find some $\kappa > 0$ such that the implicit polynomial of even degree fitted to a closed shape (which is not sampled too coarsely) has a bounded zero set.

The limiting coefficient vector for odd degree implicit polynomial curves is an interesting case since odd degree implicit polynomial curves can never be bounded. Figure 3.7 shows the evolution of a *5'th* degree implicit polynomial curve fitted to the same data set as in

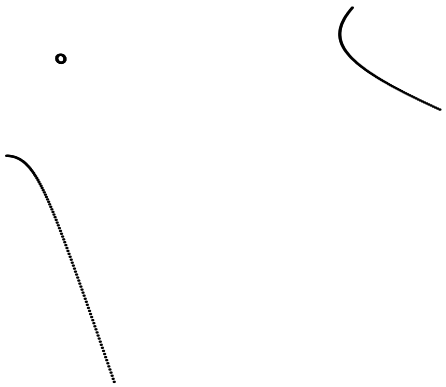


Figure 3.8: A global look at the *5'th* degree implicit polynomial curve of Figure 3.7 for very large values of the ridge regression parameter κ.

Figure 3.6. The behavior of the zero set around the data is very similar in both cases; however, a more global look at *5'th* degree implicit polynomial curve for very large values of the ridge regression parameter κ, Figure 3.8, reveals a fundamental difference: the zero set has asymptotes as expected whereas the *6'th* degree implicit polynomial curve has to be bounded which follows from the proof of the preceding paragraph.

3.5 Choosing the Ridge Regression Parameter

The bias of an estimator is the distance between the true value of the parameter being estimated and the expected value of the estimator, $\| \mathbf{a}_{true} - \overline{\mathbf{a}_\kappa} \|$. The variance of an estimator is its expected square deviation from its expected value, $\overline{\| \mathbf{a}_\kappa - \overline{\mathbf{a}_\kappa} \|^2}$. κ controls the *bias-variance tradeoff* [31, 85]. Usually, the variance is significantly reduced by deliberately introducing a small amount of bias so that the net effect is a reduction in total mean squared error which is defined as $bias^2 + variance$. Selection of the parameter κ in practice can be done in one of two ways depending on what the resulting fit will be used for:

Choosing κ for Shape Modeling. Here the main goal of fitting is to obtain a good repre-

sentation of the shape without too much smoothing, with bounded zero sets and without extraneous pieces in the zero set. In Figure 3.6, it can be seen that increasing κ results in first smoothing high curvature parts of the shape and then convergence to a bounded shape that does not visually represent the data. So the aim here is to choose the smallest possible value of κ that gets rid of unstable artifacts like unboundedness, see Figure 3.3, Figure 3.4 and Figure 3.9 for examples where κ was chosen in this manner. This can be done iteratively since fitting for modeling can usually be done off-line. Parameter κ can be increased from 0 to larger values until significant amounts of error start to be introduced into the fit. Polynomial Interpolated Measure (PIM) which is discussed in Chapter 4 can be used to track this error as a difference in the polynomial at $\kappa = 0$ and at the value of κ under consideration. Figure 3.9 demonstrates that *ridge regression* works successfully for data shapes of different complexities.

Choosing κ for Recognition. Here the main goal is to minimize the total mean squared error of estimator $\mathbf{a}_\kappa$. Such an optimal value of κ is empirically shown to exist and is found in Section 3.6.

3.6 Object Recognition and Stability Experiments

In this section, we will demonstrate the improvements obtained in the stability of the estimated implicit polynomial coefficients under perturbations of the input data with *ridge regression*. We will also present results of object recognition tests performed for different values of the ridge regression parameter κ and show that there is an empirical optimal value for this parameter in view of object recognition.

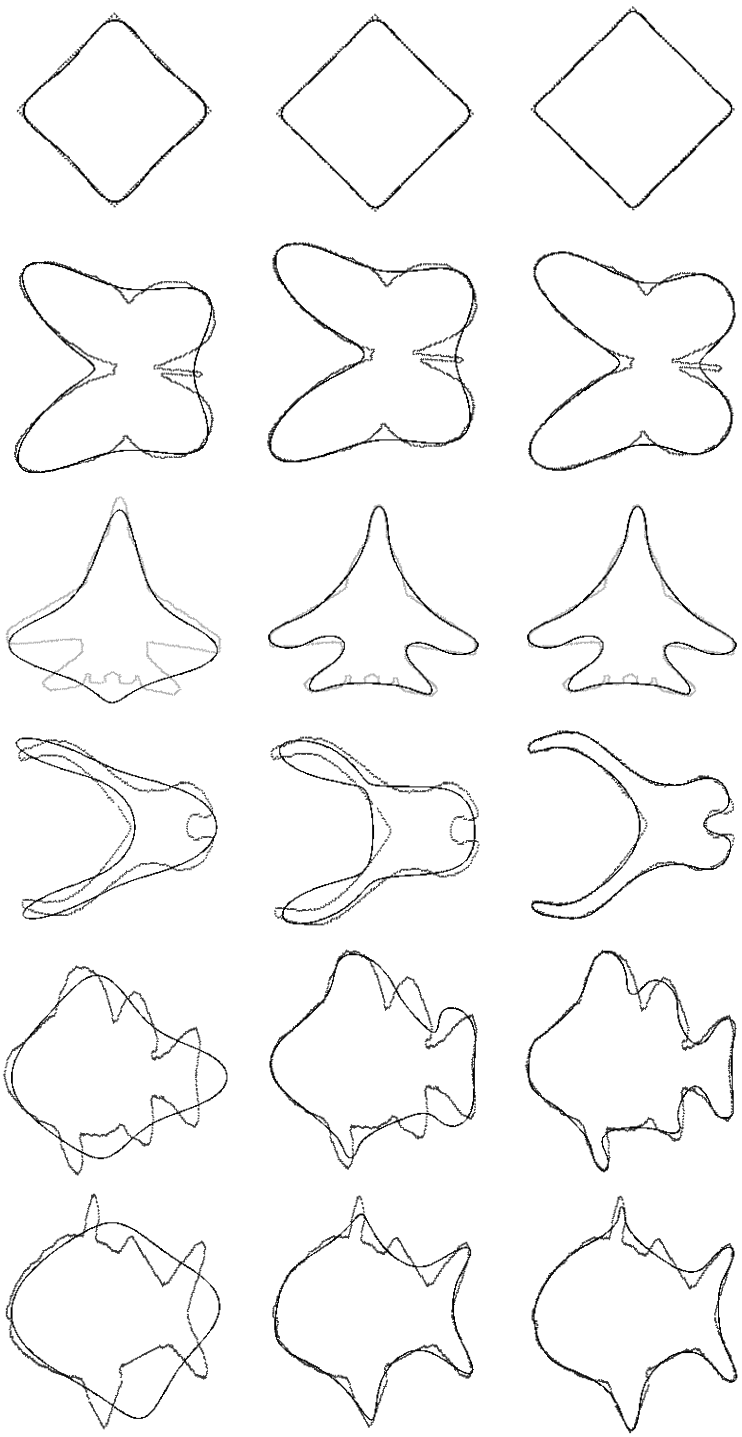


Figure 3.9: Fits for 4^{th}, 6^{th}, and 8^{th} degrees with shapes of different complexities. No extra components are close to data sets. The ridge regression parameter was chosen manually for each shape in this example.

3.6.1 Perturbation Models

Before we present experimental results, it is important to clarify how the perturbed data sets in these experiments were generated. Most researchers in the field of computer vision use white noise in their experiments on shape recognition, and thus most algorithms are optimized to handle this type of noise. In this model, the noise added to each point in the data is independent of the noise added to the other points. It is relatively easier to obtain analytical results when dealing with this noise model, but is not always an appropriate analysis of reality. White noise when used with very small standard deviations is good for simulating quantization errors; however, it is not a good model for generating deformed copies of a shape as might be sketched by a human or as might appear after segmentation from an image of an object taken under slightly different viewing conditions. We would like to be able to model these variations of shape more accurately since our motivation is to use IP fitting for indexing into image databases by query by sketch and query by example. Figure 3.10(a) and (b) show the silhouette of a fish perturbed with white noise with standard deviations 0.05 and 0.1, respectively. It is clear that these shapes, especially the latter, cannot represent the shape variations we desire. The solution we propose is simply to use colored noise instead of white noise. First generate a white noise sequence equal in length to the number of data points. Then convolve this sequence with an averaging window of length 0.15 times the number of data points. Finally add this sequence of scalars multipled by unit vectors in the direction perpendicular to the data at each point. The shapes in figure 3.10(c) and (d) were obtained with this method. Comparing these with Figure 3.10(a) and (b), it appears that colored noise models represent meaningful shape distortions whereas white noise can only represent quantization errors. The arbitrary choice of setting the length of the averaging window to be 0.15 times the length of the data sequence can be changed to

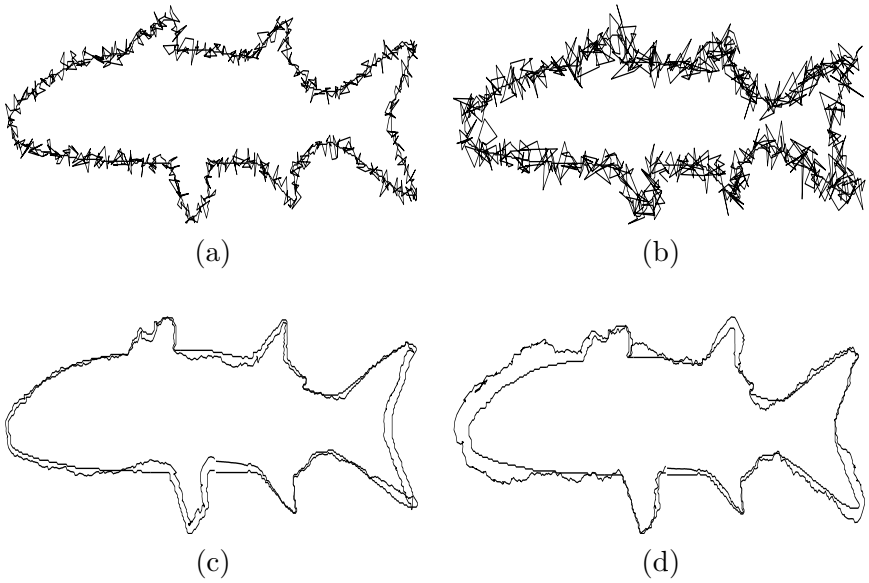
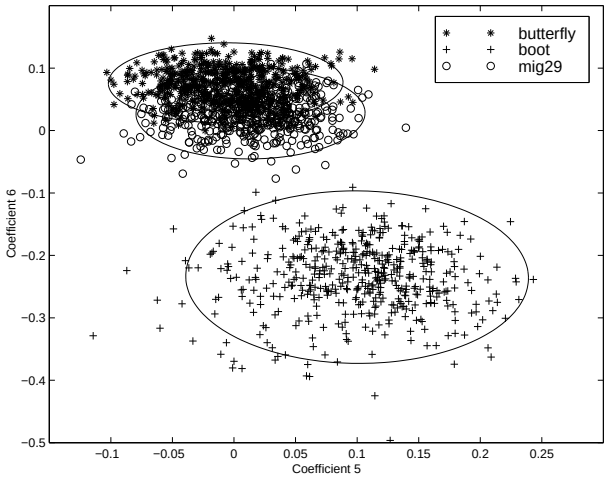


Figure 3.10: Comparison of noisy data simulation using white noise (a)-(b) with standard deviations 0.05 and 0.1, respectively, and colored noise (c)-(d) with standard deviations 0.05 and 0.1, respectively.

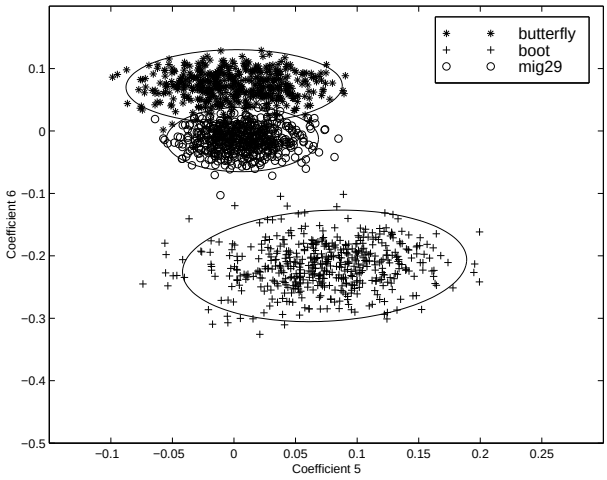
obtain different effects in the distortion produced. Another type of perturbation used in our experiments is missing data where a random point on the given shape is picked and a number of consecutive points are removed. Removing intervals introduces much stronger perturbations then removing an equal number of randomly spaced points.

3.6.2 Scattering in Coefficient Space Under Data Perturbations

In Figure 3.11(a), a 2-dimensional subspace of the coefficients obtained by *gradient-one fitting* are shown. The scatter for this pair of coefficients for the *mig29* and *butterfly* overlap significantly. By using *ridge regression*, we have smaller scattering radiuses for all shapes and thus an improvement in distinguishing these two shapes, see Figure 3.11(b). Note that the scatter plots for other coefficients for these two objects are already well separated.



(a)



(b)

Figure 3.11: Scattering of two of the polynomial coefficients under colored noise with 0.1 standard deviation for 3 shapes. (a) *Gradient-one fitting*, (b) *Gradient-one* fitting regularized by *ridge regression*, $\kappa = 0.001$. Notice that two of the shapes overlap significantly in (a), whereas in (b) they are more distinct. The scales of the axis in both plots are the same.

Figure 3.12: Objects used in the recognition experiments.

3.6.3 Object Recognition Experiments

Various object recognition experiments were performed to verify that *ridge regression* improves object recognition performance. 27 objects shown in figure 3.12, including real world objects and artificial free-form shapes ranging from simple to complex, were used for all of the experiments outlined in this section. It is important to note that some objects have very similar shapes such as the fighter aircrafts, eels, and fishes. This makes object recognition for this set of objects a non-trivial task.

Recognition performance was tested under various perturbation models which are combinations of colored noise, missing data and rotation as explained in Section 3.6.1. Given a perturbation model, 1000 samples (perturbed shapes) are generated from each base shape. Each sample is fit with an implicit polynomial curve using the methods outlined in the previous sections, thus producing a sample in coefficient-vector space for each perturbed shape. Then, a recently developed complete set of invariants [75] is computed for each coefficient-vector sample. One of the most important advantages for recognition of this specific set of invariants is that each invariant is either a linear or quadratic function of the coefficients or an angle determined by a pair of components of the coefficient-vector. This leads us to believe that they should out-perform highly non-linear algebraic invariants in robustness. Finally, a mean and full covariance matrix in the invariant space is learned for each object. Test sets (100 samples of each object) are generated in the same manner independently of the training set.

Average recognition rates for the 27 objects are plotted against the logarithm of the ridge regression parameter κ in Figure 3.13. Recognition rates obtained without using *ridge regression* are shown with the horizontal lines. In Figure 3.13(a) $4th$ degree implit polynomial

curves were used with a perturbation model of 10% colored noise [1] and random rotations combined. Optimal choice of the ridge regression parameter provides approximately 3% increase over the already high rate of 96.5%. Note that there is an optimal value of κ, this is expected since κ controls the *bias-variance* tradeoff in invariant space and some value of κ has to minimize $bias^2 + variance$. The following experiments verify this fact with the further important implication that for this set of objects, best recognition performance is obtained using approximately $\kappa = 10^{-3}$ regardless of the degree of the implicit polynomial curve or the perturbation model being used. One question that has to be investigated in future research is if this optimal value of κ will generalize to larger sets of objects.

The experiments presented in Figure 3.13(b) use a stronger perturbation model combining 10% colored noise, 10% missing data and random rotations. Both $4th$ and $6th$ degree polynomials were tested. For degree 4, optimal choice of κ provides 7% improvement in recognition achieving approximately 97%. For degree 6, a much more substantial 16% improvement is obtained raising the best recognition performance to approximately 99%. These top rates are impressive when one looks at some typical perturbed samples generated in this experiment, Fig 3.14. Note that random rotations are omitted in Fig 3.14 for easy comparison with the original shape. Using $6th$ degree implicit polynomial curves provides only a 2% advantage in recognition over using using $4th$ degree; moreover for some non-optimal values of κ and with no *ridge regression* it actually does worse. There are two important deductions here:

1. Since $6th$ degree implicit polynomial curves have more coefficients (degrees of freedom) they are more prone to problems of unstable subspaces then $4th$ degree implicit polynomial curves, especially for simpler shapes that might not require a $6th$ degree

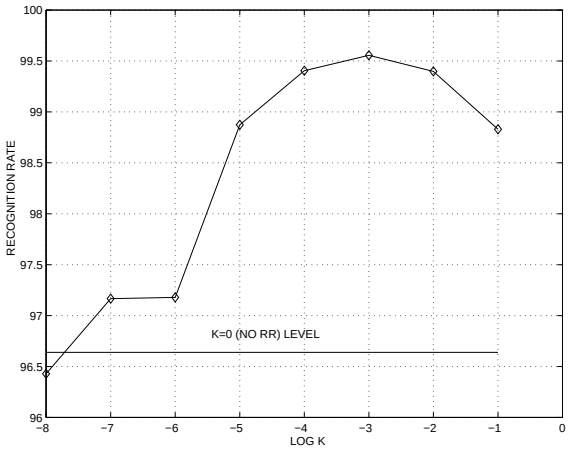
[1] % noise refers to noise which has standard deviation specified as a percantage of the shape size.

polynomial. Since this is exactly the problem *ridge regression* sets out to solve, the observation made above is totally expected.

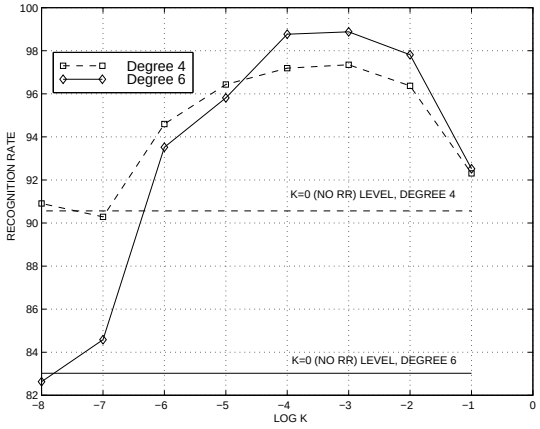
2. It might seem tempting to restrict object recognition to the use of $4th$ degree implicit polynomial curves; however, as will be made clear in the next example there are much more substantial gains to be made with the use of higher degrees in some cases.

We now use even a stronger model of perturbation, by keeping the 10% colored noise and rotation and doubling the amount of missing data to 20%. Robustness to missing data crucially depends on a good representation. Figure 3.13(c) confirms this statement; $4th$ degree implicit polynomial curves yield a top recognition rate of approximately 88%, $6th$ degree implicit polynomial curves are able to improve this rate to approximately 94%. Having established that using high degree implicit polynomial curves are necessary in certain problems, it is also very important to once more realize the crucial role played by *ridge regression* in the success of high degree implicit polynomial curves; using the optimal value of κ provided a gain of over 35% compared to no *ridge regression*, with $6th$ degree implicit polynomial curves in this example.

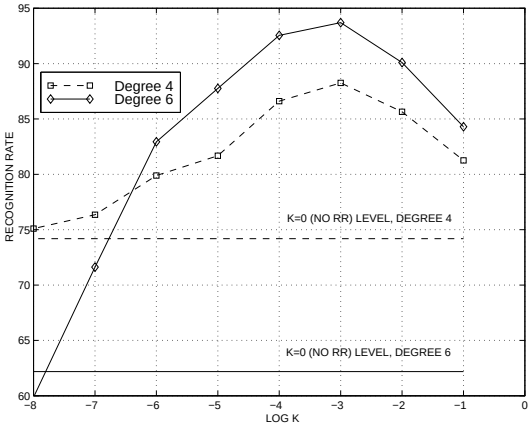
In the continuing quest for achieving maximum stability in the representation of curve data by algebraic curves (i.e., the zero sets of polynomials in x and y) and in the stability of the polynomial coefficients, Chapters 2 and 3 of this thesis make two important contributions. The first is an understanding of the role of data standardization and the gradient-constraint in improving representation and coefficient stability. This also sheds light on why the 3L fitting algorithm [45] is so much more stable than previous fitting algorithms. The second contribution is the use of rotation-invariant *ridge regression*, in the fitting, for improving the stability of both the representation and the coefficients even further. *Ridge regression* drives those portions of the polynomial zero set, that are not



(a)



(b)



(c)

Figure 3.13: 1000 perturbations of each object are used as the training set. Another 100 independent perturbations of each object are used as the test set. Perturbation models are (a) 10% colored noise + rotation, (b) 10% colored noise + 10% missing data + rotation and (c) 10% colored noise + 20% missing data + rotation.

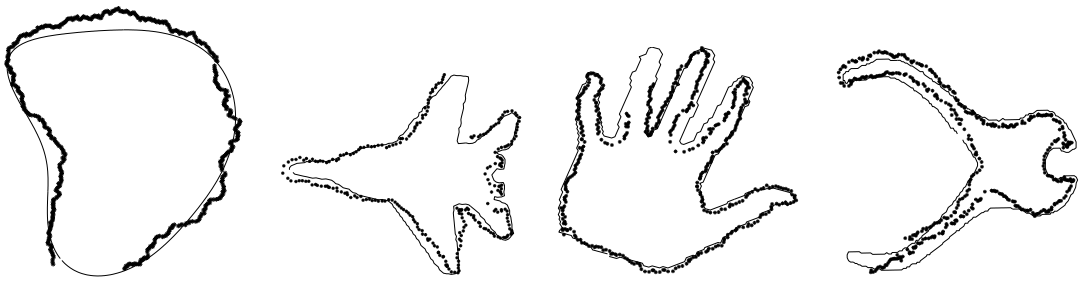


Figure 3.14: A few shapes perturbed with 10% colored noise and 10% missing data.

related to the curve data, far from the data. It also shrinks to near-zero those polynomial coefficients not important for representing the curve data. The remaining coefficients are stable and result in increased stability when used for pose-invariant object recognition or object pose estimation.

Chapter 4

POLYNOMIAL INTERPOLATED MEASURES (PIM) AND INVARIANT PATCHES

In Section 3.6.3 we have demonstrated one approach to object recognition based on vectors of invariants computed using the coefficients of implicit polynomials fitted to data sets. While this works robustly using $3L$ or *gradient-one fitting* regularized by *ridge regression*, the fact remains that implicit polynomial coefficients are global curve descriptors. Hence, no matter how robust the curve fitting algorithms are, polynomial coefficients are still susceptible to significant changes under certain data perturbations such as significant amounts of missing data. The invariants, which are functions of the coefficients, are global descriptors and thus will be affected by such changes. In this chapter, we will first introduce a local approach to computing an approximate Euclidean distance between data sets in terms of the distance between their fitted implkicit polynomial coefficients weighted in a certain way. Unlike invariant functions, this family of measures, which we shall call *Polynomial Interpolated*

Measures (PIM), depends on both the global coefficients and the local data set. They are insensitive to global changes in the coefficients that do not produce any local changes in the polynomial around the data set. Another important distinction between these two methods is that we are limited to comparing invariant vectors of implicit polynomial curves of the same degree whereas the extension of PIMs for comparing models with different degrees is trivial as will be shown in Section 4.3. In Section 4.5, we use *maximum length invariant patches* to deal with problems of missing data and occlusion in the framework of implicit polynomial curves.

4.1 Polynomial Approximation to the Distance Transform

Assume that a set $\Omega = \left\{(x_k, y_k)\right\}_{k=1}^{m}$ of measurement points can be "well" represented by an IP curve of degree d. Let g_Ω denote the *D-Euclidean Distance Transform* [16, 8] of Ω; in other words, $g_\Omega(x, y)$ is a function taking a value at (x, y) which is the Euclidean distance from (x, y) to closest point in the data set Ω. Figure 4.1(b) is the distance transform function for the shape shown in Figure 4.1(a). For each point in Ω, generate two other points each at a distance c from Ω, i.e., on the $\pm c$ level sets of $g_\Omega(x, y)$. These level sets are shown in Figure 4.1(a). We denote these artificially generated curves by Ω_c and Ω_{-c} depending on whether they are inside or outside the curve Ω, respectively. Hence, $g_\Omega(x, y)$ takes value 0 on Ω and $-c$ and $+c$ on Ω_{-c} and Ω_c, respectively. Denote the union $\Omega_{-c} \cup \Omega \cup \Omega_c$ by Ω_{3L}. We now determine a polynomial $f(x, y)$ of degree d such that its zero set approximates Ω. We do this by choosing $f(x, y)$ to be a least squares approximation to $g_\Omega(x, y)$ on Ω_{3L}.

Specifically, $\hat{f}(x,y)$ is the d'th degree polynomial for which

$$\sum_{(x,y)\in\Omega_{3L}} \big(f(x,y) - g_{\Omega}(x,y)\big)^2 \tag{4.1}$$

is minimum. This polynomial approximation $\hat{f}(x,y)$ to the Distance Transform in the vicinity of Ω is called the $3L$ fit [45, 7], Figure 4.1(c), where $3L$ refers to a fit to three level sets. Notice that in Figure 4.1(c), $\hat{f}(x,y)$ approximates the Distance Transform only in the vicinity of Ω and differs from it globally.

4.2 Comparing Pairs of Data Sets

We propose a local shape difference measure based on both the polynomial coefficients and the data sets. Lets describe a simple image database querying scenario: given a measured object boundary data Ω^q as a query shape, check to see which stored data set $\Omega^1, \Omega^2, \ldots$ it is closest to. Assume that Ω^q is a new noisy measurement along one of the stored curves, Ω^s. The data in Ω^q may be along different subintervals of the shape than is the data in Ω^s stored for that shape in the database. Hence, the two data sets represent the same shape but do not have any points in common. If this is the case or if the measured and stored data sets are sparse or the data sets contain intervals of missing data possibly due to occlusion, it may not be physically meaningful to do data set comparison by computing the average squared distance between the two sets, i.e., for each point in one set computing the square of its distance to the nearest point in the second set and averaging these squared distances. Moreover, computing the exact average squared Euclidean distance involves storing the Distance Transform $g_{\Omega^l}(x,y)$ over a set of grid points for each object in the database, and

computing

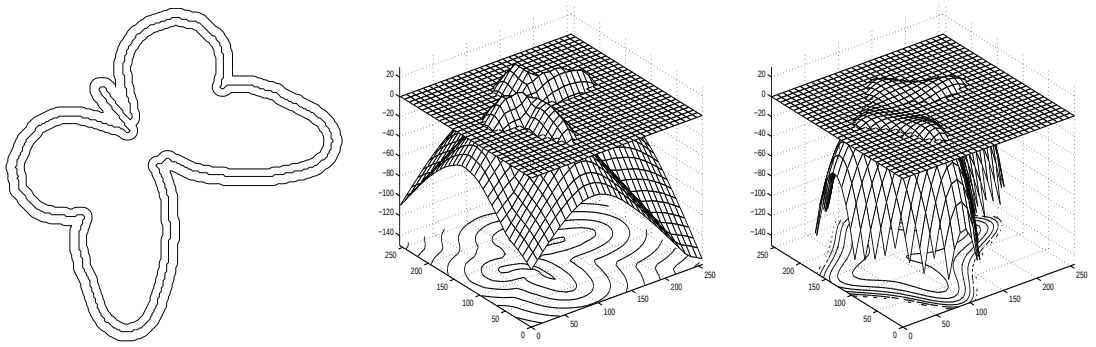
$$\frac{1}{m}\sum_{(x,y)\in\Omega^q} g^2_{\Omega^l}(x,y)$$

where m is the number of data points in the query data set. Storing $g_{\Omega^l}(x,y)$ requires considerable storage space since for good resolution, the grid points at which its value is stored have to be dense. Storing the coefficient vector $\mathbf{a}_l$ for the polynomial $\hat{f}_l(x,y)$ fitted to the data set Ω^l and computing

$$\frac{1}{m}\sum_{(x,y)\in\Omega^q} \hat{f}^2_l(x,y) \tag{4.2}$$

requires storing only a few parameters (see Section 1.1 for the exact number of parameters as a function of polynomial degree) for each shape and the computation which requires pm multiplications and pm additions is still very fast. This method has the additional advantage that if the degree of $\hat{f}_s(x,y)$ is chosen appropriately, it interpolates the data Ω^s well, and if Ω^s is sparse, then the distance from a point in Ω^q to the zero set of $\hat{f}_s(x,y)$ may be more meaningful for recognition than the distance to Ω^s as mentioned above.

Within the region bounded by the curves Ω_{-c} and Ω_{+c}, $\hat{f}(x,y)$ is generally close to the exact value $g_\Omega(x,y)$ provided that the degree of $\hat{f}$ is large enough to adequately minimize equation (4.1). As one moves away from Ω outside the object, the approximation quickly becomes larger than the exact distance; this can be observed by comparing the contour plots in Figure 4.1(b) and (c). Inside Ω, the approximation becomes smaller than the exact distance. Since the goal is computationally fast and robust comparison, a globally accurate approximation to the distance transform is not essential; certain desired properties of $\hat{f}(x,y)$ suffice for our purposes. This means that if two data sets are "close", we want



(a) (b) (c)

Figure 4.1: (a) Level-sets Ω_{-c},Ω and Ω_{+c} for a butterfly shape, (b) the Distance Transform function g_Ω and (c) the polynomial approximation to the Distance Transform around Ω.

a *dissimilarity measure* that is small, and if the two data sets are very much different, we want a dissimilarity measure that is large. Furthermore, if we wish to permit a significant amount of variability in an object shape, we need to be able to determine *relative amounts of significant dissimilarity*. Note that *gradient-one fitting* could still be used for obtaining a $\hat{f}(x, y)$ with the desired properties. The reason we choose to make use of the $3L$ fitting method in this chapter is because it makes more intuitive sense in the current context. The $3L$ and *gradient-one* fitting algorithms regularized by *ridge regression* provide $\hat{f}(x, y)$ that are very likely to meet the requirements stated above. It was shown in Section 3.4 that when the data shape is closed and the degree of the fitted polynomial is even, the implicit polynomial curve converges to the curve given by $\mathbf{a}_\infty$ which is always bounded. This implies that a choice of the ridge regression parameter κ for which the implicit polynomial curve is bounded exists. When the zero set curve is bounded and has no extra artifact pieces that do no represent the data, the polynomial surface should be approximately monotonous around the zero set curve.

4.3 IP Shape Dissimilarity Measures

In this section, we will introduce three versions of the dissimilarity measure of equation (4.2) formulated in vector notation. Let us begin by deriving a local norm for a coefficient vector $\mathbf{a}$ for a implicit polynomial curve of degree d given a data set Ω. The coefficient vector $\mathbf{a}$ is not necessarily the result of implicit polynomial curve fitting to the data set Ω; in fact they can be totally unrelated. Let $\mathbf{y_k}$ be the monomial vector associated with point $(x_k, y_k) \in \Omega$ as in Section 2.1 and let $\mathbf{M}$ be the matrix of monomials as defined in equation (2.3). Recall that $\mathbf{M}$ is of dimension $p \times m$ where p is the dimensionality of the coefficient vector $\mathbf{a}$ and m is the number of points in Ω. Define the average square norm of $\mathbf{a}$ over Ω as

$$\begin{aligned}(\mathbf{a}, \mathbf{a})_\Omega &= \frac{1}{m}\sum_{k=1}^{m}\Big(\sum_{0\le u,v;u+v\le d} a_{uv}x_k^u y_k^v\Big)^2 = \frac{1}{m}\sum_{k=1}^{m}(\mathbf{y_k}^T\mathbf{a})^2 \\ &= \frac{1}{m}\sum_{k=1}^{m}\left(\mathbf{a}^T\mathbf{y_k}\mathbf{y_k}^T\mathbf{a}\right) = \frac{1}{m}\mathbf{a}^T\left(\sum_{k=1}^{m}\mathbf{y_k}\mathbf{y_k}^T\right)\mathbf{a} \\ &= \frac{1}{m}\mathbf{a}^T\mathbf{M}\mathbf{M}^T\mathbf{a} \\ &= \frac{1}{m}\mathbf{a}^T\mathbf{S}\mathbf{a}\end{aligned} \tag{4.3}$$

where $\mathbf{S}$ is the scatter matrix of monomials as defined in equation (2.4) and m is the number of data points in Ω. $\mathbf{S}$ is symmetric and non-negative definite provided $p \le m$, that is the dimension of the coefficient vector is less than the number of data points in Ω. We can assume that this condition is met; for example, typical data sets have on the order of a few hundred points and p is 15 for a 4th degree IP. Since $\mathbf{S}$ is symmetric and non-negative definite, equation (4.3) is a squared norm. Moreover, given two coefficient vectors $\mathbf{a}$ and $\mathbf{b}$

and a data set Ω, we can define an inner product

$$(\mathbf{a}, \mathbf{b})_\Omega = \frac{1}{m}\mathbf{a}^T\mathbf{S}\mathbf{b}. \tag{4.4}$$

If the coefficient vectors $\mathbf{a}$ and $\mathbf{b}$ represent implicit polynomial curves of different degrees, they will have different dimensionalities. If the shorter coefficient vector is padded with zeros for the higher order terms and $\mathbf{S}$ is computed for the larger degree equation (4.4) still applies. Being able to compare polynomials of different degrees is a significant advantage of PIMs over invariants.

4.3.1 The Non-symmetric PIM

In the context of Section 4.2, assume for now that $\mathbf{a}_s$, which defines the approximation $\hat{f}_s(x, y)$ to the distance transform of Ω^s, has been stored in a database. Note that Ω^s has not been stored, it has been discarded after the computation of $\mathbf{a}_s$. Recall that Ω^q is the query data set which we want to match against the models stored in the database one of which is $\mathbf{a}_s$. It is useful to think of the summation in equation (4.2) as the integration of $\hat{f}_s^2(x, y)$ with respect to a discrete measure taking value $\frac{1}{m}$ at each point (x, y) in Ω^q. Using the notation developed in Section 4.3, we can rewrite this measure as

$$\mathrm{PIM}_{ns}(\Omega^s, \Omega^q) = (\mathbf{a}_s, \mathbf{a}_s)_{\Omega^q} \tag{4.5}$$

This measure compares two data sets non-symmetrically by evaluating the polynomial interpolation given by $\mathbf{a}_s$ of the data set Ω^s over the other data set, Ω^q. Hence, we call this the non-symmetric PIM. This type of PIM is useful when only one of the data sets is available as described in the database querying scenario described above. Figure 4.2 demonstrates

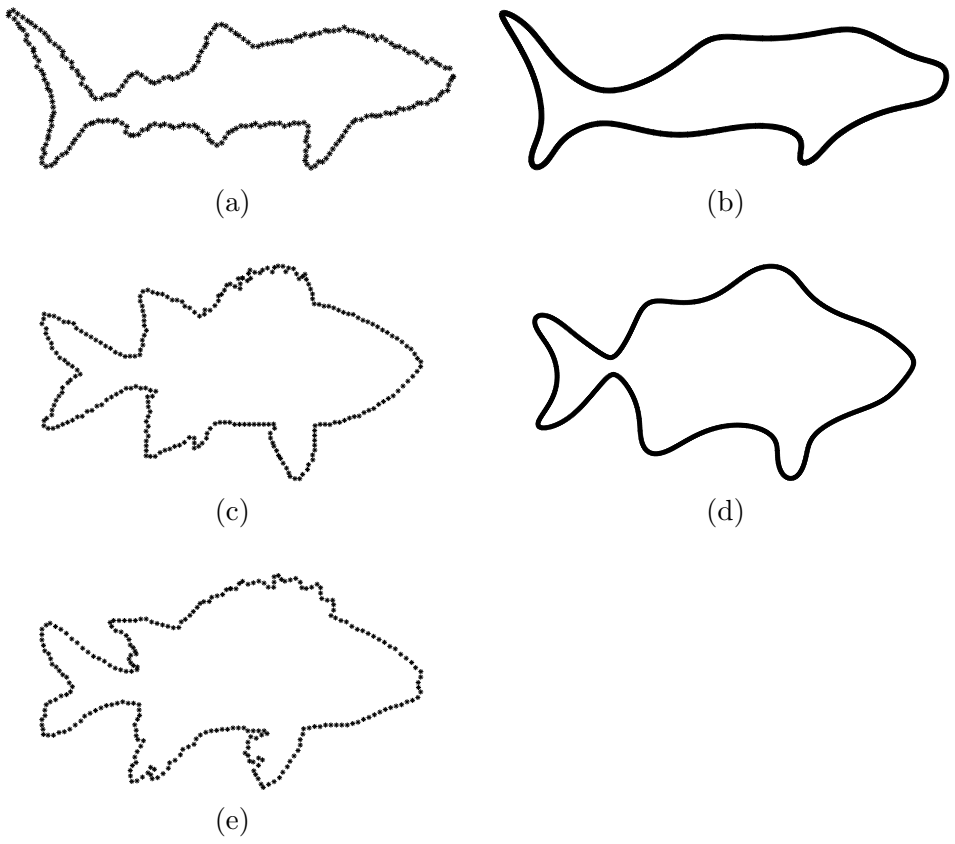


Figure 4.2: A demonstration of the use of PIM. (a) and (c) are two datasets (Ω^1 and Ω^2) for which 10th degree IP curves, (b) and (d) respectively, have been stored in a database. A query object, Ω^q, is shwon in (e). The PIM values of the query object and the database shapes in (a) and (c) are 0.2273 and 0.0064, respectively.

the use of PIM_{ns} in the database indexing scenario. In Figure 4.2(a) and (c) data sets Ω^1 and Ω^2 are shown. Coefficient vectors for the 10th degree IP models for these two shapes, $\mathbf{a}_1$ and $\mathbf{a}_2$, have been stored in a database. Figure 4.2(b) and (d) shows the zero sets of the IP models $\mathbf{a}_1$ and $\mathbf{a}_2$, respectively. Notice that with the use of high degree polynomials, it is possible to represent even complicated objects adequately with a single IP curve. Figure 4.2(e) is a query data set, Ω^q, which matches the object shown in Figure 4.2(c) much better than the object shown in Figure 4.2(a). Evaluating the PIM for the two database models gives the following result: $(\mathbf{a}_1, \mathbf{a}_1)_{\Omega^q} = 0.2273$ and $(\mathbf{a}_2, \mathbf{a}_2)_{\Omega^q} = 0.0064$. Thus from

the point of view of the PIMs, the query object is approximately 35 times closer to the shape in Figure 4.2(c) than Figure 4.2(a). This provides a reality check that equation (4.5) is working as it should.

We now digress for a moment to clarify an issue with PIMs. If $\mathbf{a}_1$ were to be multiplied by a scalar k the measure in equation (4.5) would get multiplied by k^2. This is undesirable for a shape dissimilarity measure because multiplying the coefficient vector by a scalar leaves the shape representation, namely the polynomial zero set, unaffected. Our approach is to standardize the shape by centering its center of mass at the origin and scale it by an Euclidean invariant measure of its size as described in Section 2.3. If the $3L$ fitting algorithm is being used, the level sets for each shape are consistently generated at a distance $c = 0.05$ from the dataset. When all coefficient vectors to be used in (4.5) are obtained in this way, arbitrary scalings of the measure are avoided. If the *gradient-one* fitting algorithm is used than the unit magnitude soft constraint guarantees this property.

4.3.2 The Symmetric PIM

In a somewhat different scenario both data sets may be available for consideration. This would be the case, for example, in stereo reconstruction where one is interested in computing the pose transformation between a pair of views of the same object. Implicit polynomial curves could be fitted to each shape and they can be aligned by minimizing a PIM between the two datasets, see Section 4.4. Thus, unlike the database query scenario, both data sets are immediately available and we would not be making use of half of the available information if we were to use the non-symmetric measure defined in equation (4.5). Thus,

we define another PIM:

$$\mathrm{PIM}_s(\Omega^1, \Omega^2) = \frac{(\mathbf{a}_1, \mathbf{a}_1)_{\Omega^2} + (\mathbf{a}_2, \mathbf{a}_2)_{\Omega^1}}{2} \tag{4.6}$$

Care must be taken in choosing between equation (4.5) and equation (4.6) in practice depending on the requirements of the task at hand. Figure 4.3 depicts a situation where these two measures provide two different interpretations. Two data sets: one consisting of the "o" symbols only and one that is the union of "o" and "x" symbols and the two IPs of degrees 3 and 4 fitted to these sets respectively are shown. The two implicit polynomial curves agree very well on the data set composed only of "o"s, whereas over the larger data set the they are very different. Using equation (4.5), we compute the non-symmetric PIMs of the 4th degree IP curve (fitted to the larger data set) on the smaller data set as 0.0004 and the $3rd$ degree IP curve (fitted to the smaller data set) on the larger data set as 0.2947. Equation (4.6), which is the average of these two non-symmetric measures is 0.1476. If one uses directly the symmetric measure, these two data sets will be perceived as very different. However, a closer look at the two non-symmetric components actually can reveal that one data set is actually a subset of the other. Thus, if one is interested in comparing patches to entire objects the two non-symmetric measures should be examined, but if the aim is to find fully matching shapes, then the symmetric measure should be preferred. This is also an example of comparing data shapes when the degrees of their fitted implicit polynomial curves are different.

It was stated in Section 4.2 that an important motivation in using PIMs to compare two discrete data sets instead of using an average distance between closest pairs of points of the data sets is the case when the data sets have intervals of missing data. If the dissimilarity

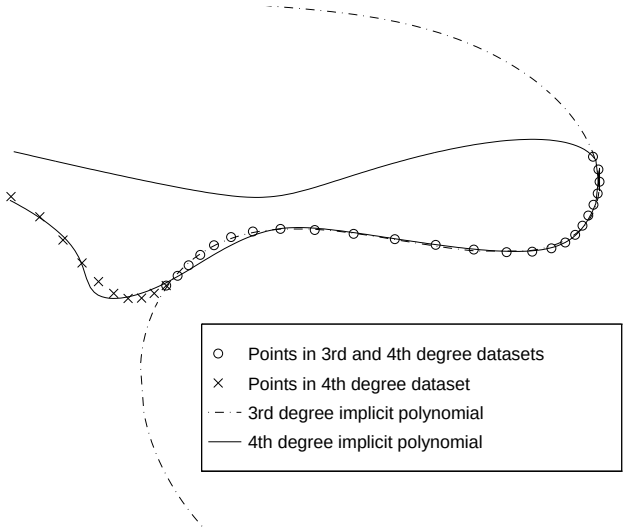
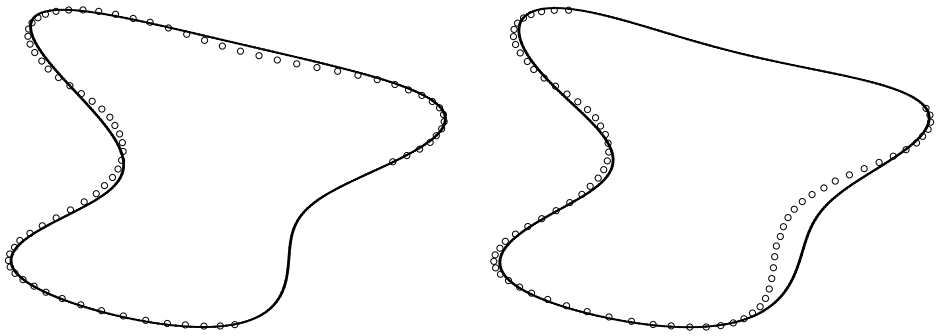


Figure 4.3: A situation where the symmetric measure defined in equation (4.6) differs from the non-symmetric measure of equation (4.5)

measure is the average distance from each point of a data set to the closest point on the other data set, meaningful results can not be obtained because for those points on the intervals of missing data, the closest points are not actually "close". Figure 4.4 demonstrates this situation. Figure 4.4(a) and (b) show two data sets that represent the same shape though with missing data along different intervals. Also shown in Figure 4.4(a) is the 4th degree implicit polynomial curve fitted to the data set in Figure 4.4(b). Similarly, Figure 4.4(b) shows the 4th degree implicit polynomial curve fitted to the data set in Figure 4.4(a). One of the strongest aspect of polynomial models is their interpolation powers which results in a certain robustness to missing data. The symmetric PIM of these two data sets is 0.0165, a value small enough for these two objects to be considered the same. Notice that when the missing data is along a high curvature interval, like in Figure 4.4(a), the fitted polynomial will be smoother than the original data along this interval, but will still be close, see Figure 4.4(b).



(a) (b)

Figure 4.4: PIM is superior to computing the exact Euclidean distance between a pair of datasets when there is missing data possibly due to occlusion. In this example, the data points are shown with circles and the fitted implicit polynomial curves with the superimposed curves. The curve shown on the left is the implicit polynomial curve fit to the data set on the right and the curve shown on the right is the implicit polynomial curve fit to the data set on the left.

4.3.3 The PIM of the Difference of 2 Coefficient Vectors

There is one more version of PIM as a dissimilarity measure that is worth considering. Assume that we have two data sets: a shape Ω^1 and an Euclidean transformed version of the same shape Ω^2, and we are trying to do pose estimation, that is to compute the transformation matrix between the two instances of the object. Obtain their respective implicit polynomial coefficient vectors, $\mathbf{a_1}$ and $\mathbf{a_2}$. Now let $\tilde{\mathbf{a_2}} = \mathbf{T}\mathbf{a_1}$, where $\mathbf{T}$ represents an Euclidean transformation. The transformation between the two objects is the Euclidean transformation matrix $\mathbf{T}$ which minimizes $(\tilde{\mathbf{a_2}}, \tilde{\mathbf{a_2}})_{\Omega^1}$. We will discuss pose estimation using PIM in Section 4.4. Notice that $(\mathbf{a_1}, \mathbf{a_1})_{\Omega^1}$ is a lower bound for $(\tilde{\mathbf{a_2}}, \tilde{\mathbf{a_2}})_{\Omega^1}$ since $\mathbf{a_1}$ is the implicit polynomial coefficient vector that best represents Ω^1. $(\mathbf{a_1}, \mathbf{a_1})_{\Omega^1}$ is the fitting error of $\mathbf{a_1}$. Let us define a PIM of the difference of these 2 coefficient vectors:

$$PIM_{diff}(\Omega^1, \Omega^2) = (\mathbf{a_1} - \mathbf{a_2}, \mathbf{a_1} - \mathbf{a_2})_{\Omega^1} \tag{4.7}$$

Minimizing this measure has the advantage that its minimum will be approximately 0 instead of some arbitrary fitting error if the two data sets are identical up to an Euclidean Transformation.

4.4 Pose Estimation and Object Recognition

In previous sections we assumed that the two IPs were aligned, but this is not always true in practice. In this section, we will use the setup of Section 4.3.3 where $\mathbf{a_1}$, $\mathbf{a_2}$, $\tilde{\mathbf{a_2}}$ and $\mathbf{T}$ were introduced. The pose estimation can be done globally, i.e., using only the the two IPs coefficients without using data sets. In [75], a technique for global pose estimation is introduced. It is computationally efficient and optimal in the sense that all the information about pose in $\mathbf{a_1}$ and $\mathbf{a_2}$ is used. The advantage of using PIM for pose estimation over other techniques is that it allows us to align the two IP curves locally in the neighborhood of point set Ω^1 by the minimization

$$\min_{\theta,t_x,t_y} \Big(\mathbf{a_1} - \mathbf{T}(\theta,t_x,t_y)\mathbf{a_2}, \mathbf{a_1} - \mathbf{T}(\theta,t_x,t_y)\mathbf{a_2}\Big)_{\Omega^1} \tag{4.8}$$

where θ is the rotation angle, and (t_x, t_y) is the translation between the two IPs.

Initially, we search for analytical solutions. When the Euclidean transformation is reduced to a rotation, all minima are the zeros of a complex polynomial. This complex polynomial is obtained by using the complex coordinates for (x, y), the complex representation for the Polynomial [75], and the complex representation of the θ rotation, in the derivative of PIM_{diff} with respect to θ. Therefore for rotation estimation, a root finder can be used for solving the PIM_{diff} minimization. But for translations or full Euclidean transformations, analytical derivations of PIM_{diff} require solving complicated systems of

high degree algebraic equations even when the complex representation is used.

An alternative is to minimize numerically using the Gauss-Newton algorithm, which is a gradient descent minimization specific for Least-Square minimization. Of course, an initialization must be provided, and thus the PIM_{diff} minimization can be seen as a refinement process. The easiest way of obtaining an initialization is to use the center of mass and second order scatter matrix (i.e geometric moments) of the two point sets for an approximative pose. This approach works fine for elongated objects, the rotation angle may be difficult to obtain when the shape is compact (or circular shapes) because of stronger rotational symmetry. For this kind of bloby objects, the advantage of using IP curve fitting and then doing the pose estimation on the IP coefficients has been shown in [75].

After an initial estimate for the pose of the query object has been obtained in the manner described above, we proceed to fine tune these estimates using the minimization in equation (4.8). This involves a simple gradient descent search algorithm as the initial estimates are already good. It is also important to note that isotropic size changes are easily included in this fine tuning stage if it is necessary to deal with query objects that are scaled versions of the objects in the database. Moreover, the initial estimates of rotation and translation obtained from the coefficients are not affected by isotropic size changes.

We have repeated the experiment of Section 4.3 with the query object rotated and translated in an arbitrary manner. The objects used are the same as those in Figure 4.2 with the exception that the query object shown in Figure 4.2(e) is rotated and translated. This scenario was repeated for three different arbitrary transformations of the query object to show that PIMs work as well with non-aligned objects as aligned objects. The results are summarized in Table 4.1. PIM values for the matching database object, i.e. Database Object #2, is always much lower than the values for Database Object #1. The small

variations in the PIMs values for Database Object #2 are due to the quantization noise introduced by rotation and due to the IP fitting algorithm. The values for the original query are larger because no pose estimation was used for the original query. Even non-matching objects can be aligned to a certain extent with pose estimation resulting in smaller PIM values.

	Database Obj. #1	Database Obj. #2
Original Query	0.2273	0.0064
Transformed Query #1	0.1307	0.0060
Transformed Query #2	0.1307	0.0061
Transformed Query #3	0.1308	0.0062

Table 4.1: Results of database queries with transformed objects.

We *Polynomial Interpolated Measures* to be useful as final stages for object recognition or shape based indexing into shape databases. In such a scenario, the number of potential matching candidates for the query object will be cut down several orders of magnitude using indexing based on invariants. This is an extremely fast operation since it requires obtaining the polynomial approximation to the query shape, computing its invariant vector and indexing into a stored database of invariant vectors. The remaining set of candidates will be examined using PIMs which allows finer recognition.

4.5 Invariant Patches and Parts

Let us examine a scenario where boundaries of objects present in a scene are found by some segmentation or edge detection/grouping routine. In many computer vision applications, data may not be available along the entire shape because of partial occlusion or missing data. Missing data may be due to the failure to detect edges at locations where edge strength is low. This problem is an important challenge for model based object recognition,

shape based indexing, pose estimation and other model based computer vision applications. Using a single shape descriptor for the entire data must be ruled out unless the missing data is a minimal percentage of the data present. If a single descriptor, $\mathbf{a}'$, is computed from the data which has missing parts, it will differ from a descriptor, $\mathbf{a}$, computed from the whole data. The magnitude of this difference will in general be an increasing function of the amount of missing data. In the case of implicit polynomial curves, up to a certain percentage of missing data $\| \mathbf{a} - \mathbf{a}' \|$ is usually a linear function of the percentage of missing data with a slope close to 0; however, as more data points are taken out, $\| \mathbf{a} - \mathbf{a}' \|$ makes a dramatic break point and quickly becomes very large. For example, in [74] Tarel et-Al. find this break point to be around %12 for pose estimation with algebraic surface models.

Occlusion is particularly troublesome because not only is a certain interval of data missing there is also clutter data from the occluding object. If a single shape descriptor were computed from the entire data in such a situation, it would differ greatly from a descriptor for the actual shape. Can occluded objects be recognized from the information present in the image? One approach to the solution of this problem is to find the physical parts of an object and to match these *parts* to the set of object parts in memory. Then from the *parts* found in the image, one can speculate what object is present in the image. If a high enough number of *parts* are unoccluded, reliable recognition will be possible. Although, this approach is compelling intuitively, finding physically meaningful parts of objects automatically is a very hard task.

There are a number of different approaches to the concept of *parts*: (a) general shape features [66]; (b) geometric primitives; (c) features pertinent to the restricted set of objects under consideration in a particular application; (d) area or volume scale; (e) complexity. Our view of patches and parts is that they are simply subsets of $2D$ curves or $3D$ surfaces

that can be found reliably in any data in which they are visible. These parts are not physically meaningful. Invariant patches are parts of objects that can be found invariantly under certain transformations, e.g., Euclidean. Invariant patches of an object generate a redundant set of overlapping parts because a physically meaningful decomposition of the object is not sought. In this section, we formulate invariant patches in terms of algebraic curves and show how they can be used for object recognition. We will call these patches *maximum length invariant IP patches*.

Choose a degree d for the implicit polynomial models. Given a data curve C, choose any point p on C. Now determine the conseccutive set of points that starts at p, transverses C clockwise, and has maximum length such that the patch can be "well fit" by a $d'th$ degree implicit polynomial curve. Now consider a Euclidean or Affine transformation of the original curve. Take the point p' on it corresponding to point p on the original curve before the transformation. Choose a maximum length patch on this curve starting at point p' and transversing C clockwise. This patch will be the Euclidean or Affine transformation, respectively, of the patch on the original curve as long as the fitting methods that are used to obtain the implicit polynomial curves and the definition of "well fit" preserve the type of invariance under consideration. Hence, this is a way of choosing patches that are invariant to Euclidean or Affine transformations. If this is done starting at every point on a shape curve, the resulting set of *maximum length invariant patches* is our set of "parts" for use in recognition or indexing or other purposes. Of course, these "parts" do not have any physical meaning. Note, the *maximum length invariant IP patch* can be found with a reasonable amount of computation because our fitting is linear least squares, so that the fitting can actually be done recursively with a small amount of computation.

To make the definition of the *maximum length invariant IP patches* more concrete, we

have to elaborate on what is meant in the previous paragraph by a data set "well fit" by a $d'th$ degree implicit polynomial curve. In [46], we had proposed to find *maximum length invariant IP patches* in the following manner: Start at point p, use $3L$ fitting and compute the average mean square fitting error, $E(d, p, l)$, as a function of increasing patch length l. Given an implicit polynomial curve and a data set, $E(d, p, l)$ is found by evaluating the total fitting error and dividing it by the number of points in the data set. This error measure will be roughly constant as long as a $d'th$ degree implicit polynomial curve fits the data well. When a $d'th$ degree implicit polynomial curve no longer can provide satisfactory fits, the error starts to increase rapidly. Thus, we imposed a threshold on E to find *maximum length invariant IP patches*.

This approach works fine for smooth curves, but with increasing amounts of noise and perturbations added to the shape curve, E also becomes increasingly noisy, especially for patches with relatively fewer points. Consequently, decisions based on thresholding E become less reliable. To solve this problem, we can modify our approach in the following manner. Implicit polynomial curves of moderate degrees have limited shape representation power and thus can not model small details. For example, if the data set is a circle of radius r with sinusoidal modulations of amplitude $<< r$, then a second degree implicit polynomial curve will fit the data "almost as well as" moderately higher degree implicit polynomial curves. This is because the fraction of the fitting error E due to the small scale details, perturbations and noise is approximately irreducible by implicit polynomial curvess of the degrees that are of interest to us. [1] The same reasoning states that given a curve "well fit" and "well constrained" by a $d'th$ degree implicit polynomial curve, a noisy version of the same curve should be fit by $d'th$ and $d + 1'th$ degree implicit polynomial curves with al-

[1] The same property will be useful again in fitting implicit polynomials to windows of image intensity in Chapter 5.

most the same accuracy. By "well constrained" we mean that there are enough data points present to justify using a degree d polynomial; this usually translates to using more points than the number of polynomial coefficients $\frac{(d+1)(d+2)}{2}$. This condition is necessary for the error due to perturbations and noise to be approximately irreducible in the fit. Define a new error function

$$F(d, p, l) = E(d, p, l) - E(d + 1, p, l)$$

which is the difference in the average mean square errors in fitting a patch of length l starting at point p with $d'th$ and $d + 1'th$ degree implicit polynomial curves. The coefficient space of a $d'th$ degree implicit polynomial curve is a subset of that of a $d + 1'th$ degree implicit polynomial curve. Thus, the global minimum of E that is achieved by fitting a degree d implicit polynomial can also be achieved by fitting a degree $d + 1'th$ implicit polynomial curve. Since linear least squares fitting is an exact global minimization, $E(d + 1, p, l) \leq E(d, p, l)$ is always true and thus $F(d, p, l) \geq 0$. F will be very small for small perturbations of data sets "well fit" by a $d'th$ degree IP curve; this follows from the above arguments. Once the length of the patch reaches the point where $d'th$ degree is no longer sufficient, F will increase sharply. We find *maximum length invariant IP patches* by imposing a threshold on F, see Figure 4.5. This threshold should be chosen as a function of shape size. For shapes standardized as in Chapter 2, 0.5 is an appropriate threshold. Figure 4.5(a) and (b) show *maximum length invariant IP patches* found in a data set and its perturbed version. The patches shown start at matching points on the two shapes. Figure 4.5(c) and (d) are the graphs of the error as a function of number of points in the patch. Observe that from Figure 4.5 that thresholding F is more robust than thresholding E under data perturbations

Degree	MXLP % length variation
2	2.39
3	3.12
4	3.22
6	3.44

Table 4.2: Stability of *maximum length invariant patches*. Average *maximum length invariant patche* length variation under %10 colored noise is given as a percentage of the number of data points in the input shape.

since F is less sensitive to perturbations. For a given shape, the set of *maximum length invariant patches* are found by choosing every point in the data set as a starting point for growing a *maximum length invariant IP patch* as described above.

4.5.1 Stability of Maximum Length Invariant IP Patches

We have used the same set of 27 objects used in the object recognition experiments in Section 3.6 (Figure 3.12) to test the stability of *maximum length invariant IP patches*. We generated 100 perturbed versions of every data set under %10 colored noise. We then computed the average absolute difference between the number of points in each patch and its corresponding patch in the unperturbed data set (the patch that starts at the same point in the original data set). Table 4.2 presents the results as percentages of the number of points in the data set. The variations in the *maximum length invariant IP patches* under perturbations are small enough to be useful in object recognition.

4.5.2 Object Recognition with Maximum Length Invariant Patches

Invariant vectors were computed for the patches found above. [75] The distributions of vectors of algebraic invariants for *maximum length invariant IP patches* have the following important properties that differ from vectors of invariants computed from single descriptors fitted to entire shapes:

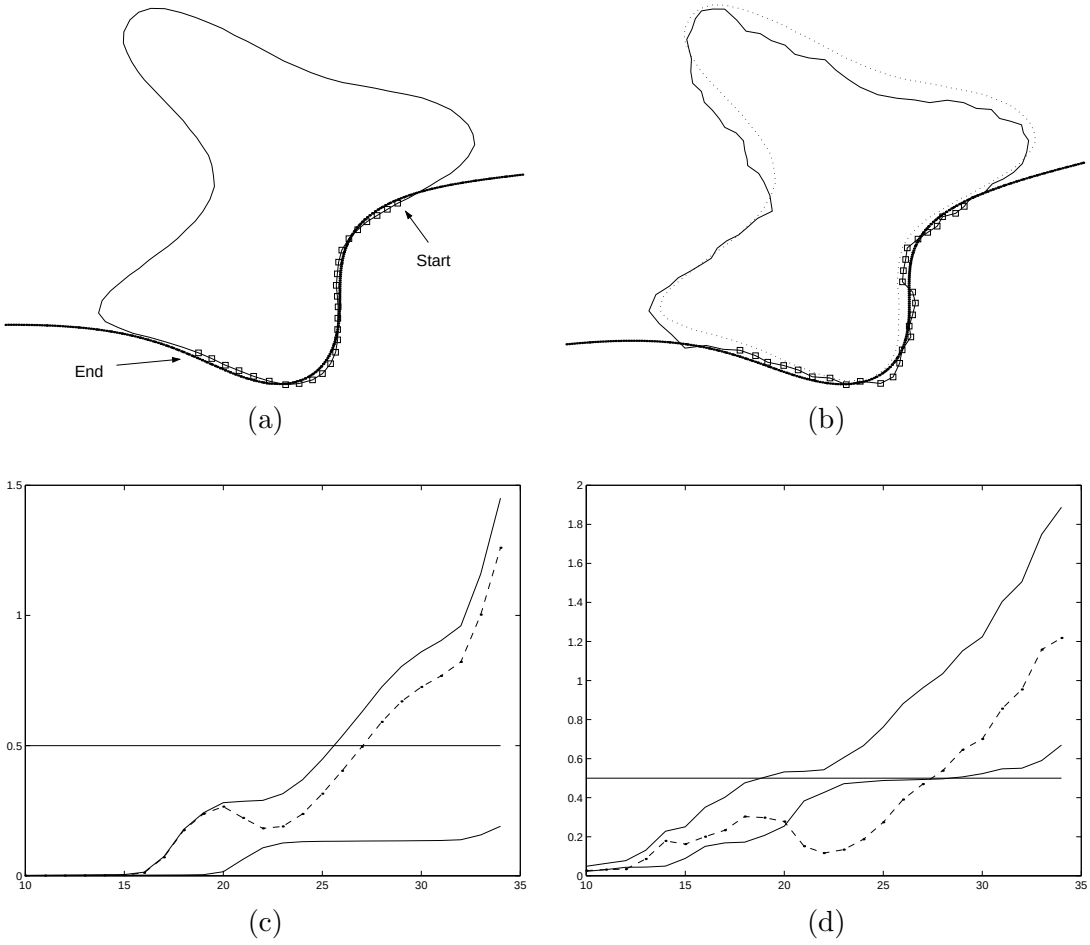


Figure 4.5: (a) A shape curve, (b) same shape with perturbations and the original shape is shown with the dotted curve for comparison. *3'rd* degree *maximum length invariant IP patches* found for one starting point, p, are shown with the squares and the implicit polynomial curve fitted to these data sets is shown with the heavier curve. (c) - (d): $E(3,p,l)$, $E(4,p,l)$ and $F(3,p,l)$ as a function of the number of points in the patch, l. The dashed curves are F, the lower solid curve is $E(4,p,l)$ and the higher solid curve is $E(3,p,l)$. The threshold is shown with a horizontal line. The *maximum length invariant IP patches* are located at the first intersection of this line with the F curve. Notice that F changes much less than E between the two shapes.

IP degree:	4	6
Experiment 1	%97	%99
Experiment 2	%94	%97
Experiment 3	%85	%90

Table 4.3: Average recognition rates. Experiment 1: Colored noise with standard deviation %5 of shape size and %20 missing data, Experiment 2: Colored noise standard deviation %5 and %40 missing data, and Experiment 3: Colored noise standard deviation %10 and %40 missing data.

1. The invariants of the *maximum length invariant IP patches* extracted from an object do not form a compact cloud in the invariant space. This is a consequence of the fact that a complex object contains patches that are significantly different from each other in shape and thus in the values of their invariants.

2. The values of a vector of invariants from different objects can be similar since it is possible for these different objects to contain patches that are similar.

Consequently, one reasonable way to do indexing/recognition is to use a Nearest Neighbor Classifier where an invariant vector obtained from a single patch of the object to be classified is compared against all invariant vectors in the database. The object then can be assigned to that class which has an invariant vector that is "closest" to the observed invariant vector. The recognition test we performed was designed as follows:

1. *Maximum length invariant patches* were computed for the 27 shapes. Invariant vectors were computed and stored for all the patches.

2. Variances for each dimension of the invariant space for the entire pool of invariant vectors is computed.

3. Given a perturbation model, 100 perturbed versions of each original shape was generated. *Maximum length invariant IP patches* for each perturbed shape are found and

invariant vectors computed. Each invariant vector is assigned to belong to the object that has the closest stored invariant vector. The distance used is a Mahalonobis distance where the squared difference in each dimension is weighted by the reciprocal of the variance found in step 2 for that dimension. The perturbed shape is then classified as that shape that obtains the most number of patch assignments.

The recognition scheme described above is not ideal since it does not make use of the joint geometry of the patches. Results *4'th* and *6'th* degree IP maximum length invariant patches are shown in Table 4.3 for a few perturbation models. Notice that *6'th* degree implicit polynomial curves give better results. In comparison with the results obtained in Section 3.6 using single implicit polynomials, patches perform better when the amount of missing data is large. Another point to note is that *ridge regression* with $\kappa = 10^{-3}$ is used in all implicit polynomial fits. If *ridge regression* is omitted, results are significantly worse: a recognition rate of %72 was obtained in experiment 1 with no *ridge regression*. See Table 4.3 caption for the description of the experiments.

The above procedure works well for shape recognition; however, it requires that all patches from all objects be stored in the database and that observed invariant vectors be compared with all of the stored vectors. The storage requirements and very large computational burden makes it infeasible for indexing into large shape databases. One solution for indexing could be to use a binary tree to narrow down the search space in a very fast manner. The space of invariant vectors can be partitioned into subspaces and a binary search can be used to find approximate closest vectors to each observed vector. A good approach to doing nearest neighbor search in high dimensional spaces is given in [52]. Once the search has been narrowed down to a more manageable number of possible matches, more accurate methods that incorporate joint geometry together with implicit polynomial

invariants can be used. An interesting approach to shape recognition with joint geometry of binary features was introduced in [3]. Generalization of this approach to non-binary features such as vectors of invariants is an interesting topic for future research. *Polynomial Interpolated Measures* can be the final stage of an indexing algorithm applied on a much smaller set of shapes to order the levels of similarity between the query shape and shapes in the database that are found to be similar by the above methods.

Chapter 5

BOUNDARY CURVELET DETECTION WITH IMPLICIT POLYNOMIAL CURVES

In Chapters 2 and 3 implicit polynomial curve fitting and *ridge regression* regularization techniques were developed for data sets that are collections of points which describe the two-dimensional silhouettes of objects. [1] Such data sets can be obtained easily from pre-segmented binary images by boundary tracing; however, segmentation is not a trivial problem and assuming that pre-segmented images will be available in a general application is not realistic. Thus, a way of finding partial or complete boundary curves from images without relying on a segmentation step is needed. In this chapter and Chapter 6, we explore finding pieces of boundaries that are salient shape features for object recognition and shape based indexing. By employing implicit polynomial curves in this task we attempt to circumvent some of the usual problems that plague the fields of edge detection and edge

[1] The generalization of these methods to $3D$ surfaces is trivial.

grouping/salient contour detection in computer vision.

The contents of a digital image code a great deal of information about the physical world from which it originates. Natural properties of images are the spatial patterns the intensity values or the color vectors form at different scales. An arguably more interesting feature for use in object recognition, $3D$ surface reconstruction and other high level computer vision tasks is abrupt changes in these image properties between adjacent locations of an image. At large enough scales, these are usually the manifestation of boundaries between different objects or different parts of the same object or they are boundaries caused by illumination effects such as shadows and will be called *boundary curves* in the rest of this thesis.

Definition 1 (Boundary Curve) *A curve in the image plane that separates two distinct regions in the image with respect to some image property.*

There are other sources of abrupt changes in image properties such as line drawings. Together with boundary curves we will call these *edge curves*.

Definition 2 (Edge Curve) *A curve in the image plane that is an ordered collection of edge elements constrained by tangential continuity.*

According to the definitions above, the set of boundary curves is a subset of the set of edge curves. Boundary curves are *salient and robust shape features* for object recognition and shape based image indexing. Although boundary curves do not provide a complete description of the contents of an image, they certainly provide very important clues about the physical world from which the image originated.

Computer vision researchers have studied this object-boundary based view of images extensively. This effort has produced numerous approaches to edge detection [49, 9, 34, 69], edge grouping and detection of salient contours based on perceptual organization [83,

65, 55, 50, 28, 51, 2, 27, 88, 87], line and curve detection using the once-popular Hough transform and its generalized variants [32, 61, 5], boundary finding with active contour models [36, 11, 86], boundary finding with parametric models [70, 90], stochastic boundary estimation [14, 92], superquadrics and implicit polynomial curve and surface representations [47, 38, 79, 7, 77] and segmentation [67, 62, 53, 54, 24], to name a few examples.

Boundary curves separat two adjacent regions with different attributes with respect to an image feature: color(chromacity and saturation), intensity, texture, etc. In one extreme, a physical boundary can be undetectable if all image features attain identical values in two adjacent regions. In the other extreme, all image properties may differ and observing any single one could be enough to detect the physical boundary. Although, it is true that the more image properties considered the better are the chances of detection of boundaries, the relative merits of different image features is a topic of debate. Most approaches in computer vision have focused on using a single feature; e.g., texture segmentation [54, 10], color based segmentation [21], and intensity based edge detectors. In this thesis we concern ourselves with intensity images, the most widely used image type. A boundary curve in intensity will exist when there is a difference in the amount of light reaching the camera from two regions. Differences in the reflectance properties of objects, differences in surface orientations and differences in the amount of light reaching the two regions (such as shadow boundaries) can all give rise to an intensity boundary curve.

Color can provide very useful information in some situations. Edge detection for color images [15] and use of color clustering in segmentation [56, 21] are two examples. The extension of the proposed approach to color images is on our list of future research. We will comment on possible ways to extend the approach to make use of color information at various steps in the algorithm.

5.1 Motivation for Implicit Polynomial Boundary Curvelet Detection

Existing approaches to edge detection/grouping and segmentation are successfully used in many scenarios to find boundary curves; however, (i) these approaches are not specifically designed to detect the boundary curves (see definition 1) that we are looking for; for example, an edge detector will most likely pick out texture details as well as boundary curves , and (ii) most existing algorithms usually are highly dependent on the choice of parameters such as the extent of smoothing applied to the image and intensity contrast thresholds. We need a totally automatic procedure that will perform well in the majority of images since we propose to use it for detecting boundary curves as features for object recognition in very large image databases. These reasons have lead us to develop our own boundary curve detection algorithm based on implicit polynomial curve models. Using implicit polynomial curves for boundary curve detection result in curves that are in the group of salient and robust shape features. The algorithm developed is totally automatic and performs very well in a very broad range of images. In this chapter, we introduce an approach based on implicit polynomial curve models that detects boundary curvelets in image windows. In Chapter 6, we discuss how to merge the curvelets detected in windows to form longer curves and the advantages of this merging over edge grouping based on saliency optimization.

Edge detection algorithms, as their name implies detect single edge elements rather than edge curves which are ordered collections of edge elements. Detecting single edge elements requires the use of local operators and decisions based on the outputs of these operators. In a cartoon-like image where regions have homogeneous intensity and no noise or texture, this approach could work perfectly based on the simplest local difference operators. However,

existence of complex intensity patterns due to texture, illumination and noise in images necessitates the use of operators with larger spatial extent in an attempt to filter out these effects while detecting edges accurately. Marr and Hildreth [49] have used the Laplacian of the Gaussian to detect edges. They argued that directional operators were not needed and a symmetric operator like the Laplacian of a Gaussian could successfully be used for edge detection. They showed this to be true under the assumption that intensity varies as a linear function along the edge. Marr and Hildreth also pointed out the need for observing the image at different scales which they did by varying the standard deviation of the smoothing Gaussian; they suggested combining the outputs of these different channels into a unified primal sketch. However, the assumption of linear variation along an edge does not hold universally; it is actually violated very frequently in images. Canny [9] used three criteria: detection, localization and single response to an edge to derive optimal edge detectors for different classes of edge profiles. He showed that the optimal step edge detector is very well approximated by the first derivative of a Gaussian. He also pointed out the tradeoff in the choice of the spatial extents of the smoothing filters: larger kernels are better for detection but they degrade localization accuracy. The direction of the derivative operator should be aligned to be perpendicular to the direction of the edge element. Canny also argued for the need for using directional operators. A large extent of smoothing along an edge is useful whereas the differencing across an edge should be confined to a smaller interval. This can't be achieved by taking the directional derivative of a symmetric $2D$ Gaussian operator. It is clear that omni-directional local smoothing with any averaging kernel that spans across the boundary curve will weaken intensity discontinuity at edges and affect their localization while filtering out noise. Filtering out texture will generally require a spatially larger averaging kernel resulting in excessive blurring of boundaries. Directional operators

alone cannot solve this problem either. Directional operators have a linear smoothing axis and an orthogonal differencing axis. Consequently, valid tangential smoothing extents of directional operators are limited by the curvature of the edge curves. If the edge curve is an infinite line then directional operators will work fine at any scale; however, for a curve as simple as a circle the amount of valid tangential smoothing is limited by its curvature. Other problems arise at corners and junctions of edge curves. There is no single edge orientation valid at such locations making linear operators almost useless. Iverson and Zucker [34] have shown that significant improvements over Canny's edge detector are possible by logical checks on the validity of linear operations before performing them. This reduces false alarms and remedies the problems with previous edge detectors around junctions. However, local smoothing is still a fundamental part of their method. Convolution of an image by a Gaussian operator is equivalent to solving the partial differential equation known as the heat equation with the image as its initial condition. The heat equation formulates a linear diffusion process using second order spatial derivatives and a first order time derivative. An approach to smoothing images without blurring boundaries are the non-linear diffusion methods [42, 41, 82, 57] derived from the heat equation.

The boundary curvelet detection method proposed in this chapter is particularly good at detecting low contrast boundary curves and boundary curves embedded in noise/texture. No local smoothing is done on the image prior to the implicit polynomial model fitting in Section 5.3. The input image is processed in overlapping 8×8 windows. The choice of window size is explained in Section 5.7. The amount of overlap is such that every pixel is processed in four different windows. In each window the task is to find boundary curvelets. This task is complicated by a few factors: multiple boundaries may exist in a single window, some of these curves may intersect to form junctions and some boundaries may be of very

low contrast or may be embedded in noise/texture. We fit an implicit polynomial model to the gradient of the data in each window as explained in Section 5.3. The fitting is linear least squares thus fast and non-iterative. Due to the nature of the fitting method, if a boundary is present in the window, it will manifest itself as a level set curve of the fitted polynomial model. Thus, after fitting a polynomial model to the data we test to see if the average intensity difference between two sides of any level curve of the model is significant, Section 5.5. The degree of the implicit polynomial model is also chosen automatically, Section 5.6.

The boundary curvelet detection for every window, Figure 5.1, involves five stages: (A) computation of gradient vectors, Section 5.2, (B) fitting an implicit polynomial of degree d to the gradient vectors, Section 5.3, (C) computation of the level set curves of the polynomial, Section 5.4, (D) testing of each level curve independently to see which level set curves might correspond to edge curvelets, Section 5.5, and (E) automatic choice of appropriate degree for the implicit polynomial curve for the given image window, Section 5.6. Section 5.7 explains the details of how an entire image is broken up into windows and the issues in the selection of window size and seed area size.

5.2 Computation of Gradient Vectors

The implicit polynomial curve fitting algorithm proposed next is based only on intensity image gradient information. This calls for the computation of an approximation to the gradient from a discrete image. Let $I(x, y)$ be a continuous intensity image and $I(i, j)$ a discrete sampling of it at pixel locations on a rectangular grid. The gradient of the continuous intensity image is a continuous vector valued function $\nabla I(x, y) = \left(\frac{\partial I}{\partial x}(x, y)\ \frac{\partial I}{\partial y}(x, y)\right)^T$. Being unable to observe $I(x, y)$ directly we have to compute an approximate discrete vec-