2008

# Model-based recognition of curves and surfaces using tactile data

Rinat Ibrayev
*Iowa State University*

**Model-based recognition of curves and surfaces using tactile data**

by

Rinat Ibrayev

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Program of Study Committee:
Yan-Bin Jia, Major Professor
David Fernandez-Baca
Dirk Reiners
Greg R. Luecke
James H. Oliver

Iowa State University

Ames, Iowa

2008

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# ACKNOWLEDGEMENTS

I would like to thank my advisor, Yan-Bin Jia, for his guidance, help and support throughout this research. Also, I would like to thank my committee members David Fernandez-Baca, Dirk Reiners, Greg R. Luecke and James H. Oliver for their help in completing this work.

I would like to thank my labmates Liangchuan Mi and Jiang Tian for providing the tactile data used in the experiments. Thanks to Derrick Parkhurst for letting me use the range scanner in his lab.

I thank my parents for their support throughout my graduate studies. Thanks to my wife for her understanding and encouragement.

Support for this research has been provided in part by Iowa State University, and in part by the National Science Foundation through a CAREER award IIS-0133681.

# ABSTRACT

Model-based object recognition has mostly been studied over inputs including images and range data. Though such data are global, cameras and range sensors are subject to occlusions and clutters, which often make recognition difficult and computationally expensive. In contrast, touch by a robot hand is free of occlusion and clutter issues, and recognition over tactile data can be more efficient.

In this thesis, we investigate model-based recognition of two and three dimensional curved objects from tactile data. The recognition of 2D objects is an invariant-based approach. We have derived differential and semi-differential invariants for quadratic curves and special cubic curves that are found in applications. These invariants, independent of translation and rotation, can be computed from local geometry of a curve. Invariants for quadratic curves are the functions in terms of the curvature and its derivative with respect to arc length. For cubic curves, the derived invariants also involve a slope in their expressions. Recognition of a curve reduces to invariant verification with its canonical parametric form determined along the way. In addition, the contact locations with the robot hand are found on the curve, thereby localizing it relative to the touch sensor. We have verified the correctness of all invariants by simulations. We have also shown that the shape parameters of the recognized curve can be recovered with small errors. The byproduct is a procedure that reliably estimates curvature and its derivative from real tactile data. The presented work distinguishes itself from traditional model-based recognition in its ability to simultaneously recognize and localize a shape from one of several classes, each consisting of a continuum of shapes, by the use of local data.

The recognition of 3D objects is based on registration and consists of two steps. First, a robotic hand with touch sensors samples data points on the object's surface along three

concurrent curves. The two principal curvatures at the curve intersection point are estimated and then used in a table lookup to find surface points that have similar local geometries. Next, starting at each such point, a local search is conducted to superpose the tactile data onto the surface model. Recognition of the model is based on the quality of this registration. The presented method can recognize algebraic as well as free-form surfaces, as demonstrated via simulations and robot experiments. One difference in the recognition of these two sets of shapes lies in the principal curvature estimation, which are calculated from the close forms and estimated through fitting, respectively. The other difference lies in data registration, which is carried out by nonlinear optimization and a greedy algorithm, respectively.

# CHAPTER 1. INTRODUCTION

A human being can easily recognize objects by his eyes. However, this ability depends on the presence of light and on the appearance of the object in sight. When neither condition is satisfied, the human being cannot see the object but can still feel its shape by his hands. The hands are complementary to the eyes in object recognition tasks. In order to build a powerful recognition system, touch sensing should be employed in addition to vision.

Object recognition has traditionally been the subject of computer vision. The availability of high accuracy laser range scanners has enabled vision researchers to develop recognition strategies for free-form objects. However, occlusion and clutter are still obstacles to object recognition from range data. For instance, in an in-hand object manipulation task, most of the object might be occluded by the robot hand. Equipped with tactile sensors, the hand is able to record the contact positions on the object surface. Such tactile data could be more effective than range data in recognizing the object, especially occlusion and clutter are no longer the issues.

Tactile data do not require expensive preprocessing, like scene image segmentation in the case of range data. Since they are local and sparse, the computation with tactile data is more efficient than with dense range data. Past work on object recognition from tactile data has been limited to certain classes of surfaces including polyhedra [18, 19], convex objects [2], quadrics [30], and superquadrics [1].

In this thesis, we study object recognition using tactile data. Our focus is on the objects with *curved* boundaries and surfaces. Recognition of such objects is harder than that of polygons and polyhedra for the following reasons. The normal stays the same at every point on a polygon edge or a polyhedron facet. Also, polygons and polyhedra are well defined. So,

it is easy to distinguish between different polygons and polyhedra using angle, distance and model constraints [18, 19]. In the case of curves and surfaces, the normal is different at every point and they may not always have a closed form, which makes the recognition task very challenging.

This thesis presents two different recognitions methods. The first one is invariant-based and applicable to 2D objects. The second one is registration-based and applicable to 3D objects.

## 1.1 Invariant-based Recognition

Figure 1.1 illustrates a robotic hand with two tactile fingers touching a 2D object. Suppose through local movements the fingers are able to estimate information such as the curvatures at several points of contact. Also, suppose the shape is known to be from a finite number of families of parametric curves. We would like to recognize the shape as well as determine the finger placement.

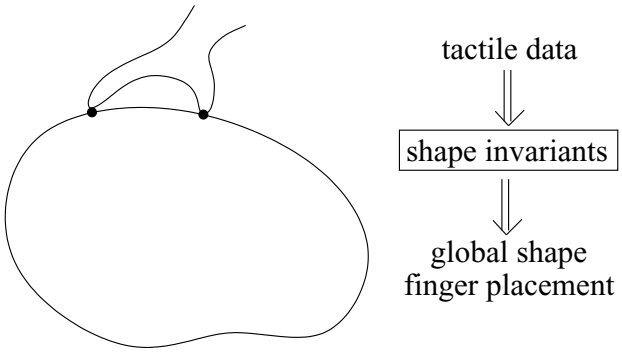



Figure 1.1 A robotic hand touching an object to recognize its shape.

This problem draws several distinctions from traditional model-based recognition. First, every model here is not a real shape but rather a *continuum* of shapes parameterized in the same form. Second, we would like to keep the use of sensor data to the minimum. This is because a touch sensor, unlike a vision system, does not generate global shape data. Third, we hope to determine where the tactile data were obtained on the shape, so as to localize the hand to it.

The characteristics of our problem naturally suggest an approach based on differential and

semi-differential invariants. Such invariants of a shape are independent of its position and orientation, the computation of which is often a burden. In addition, we are interested in invariants that are also independent of point locations on a shape at which they are evaluated. Given the local nature of touch sensing, such shape descriptors should be computable from measurements at just a few points. Our investigation will be focused on quadratic and cubic spline curves.

## 1.2 Registration-based Recognition

We present a novel method that recognizes 3D objects from tactile data. Based on fast registration, the method is capable of recognizing closed-form as well as free-form objects. Here, the model of an object to be recognized is from a known set.

Our approach is to acquire data points along three concurrent curves residing on the surface of the object. Then, we find the best superposition of these data points onto every model surface in the given set. To achieve this, we make use of the estimated local geometry at the curve intersection point $p$, and combine table lookup with local search.

In the first phase, we estimate two principal curvatures $\kappa_1$ and $\kappa_2$ at $p$ using only data points in its neighborhood as described in [27]. Meanwhile, a table has been constructed in advance for every model in the database to store the principal curvatures evaluated at discretization points. We look up the table with the pair of curvature estimates $(\kappa_1, \kappa_2)$ to find a set of locations on the model that have similar local geometry to that of $p$. Starting with each estimated location $p_i$ on the model, a local search will walk along a path to reach some point $p_i^*$ n the neighborhood that induces the best superposition of the data points onto the model. This is illustrated in Figure 1.2.

Matching between the data points and the surface, at some point $q$ on the model, is done by coinciding $p$ with $q$ and aligning the two corresponding tangent planes, and rotating the first one (along with the data curves) to yield the smallest total distance from data points to the surface. The *quality of match* is defined as the minimum aggregated distances from all local searches. The model that yields the best match against the data is then recognized.

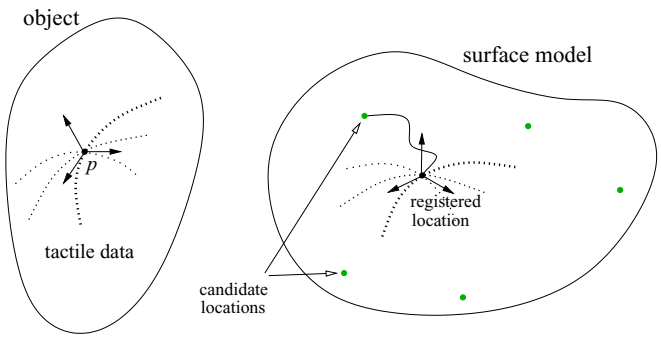



Figure 1.2 Registering three data curves on a surface. The grey (green) dots represent candidate locations of the curve intersection point $p$ on the surface found in a table lookup. A search path leads from one of these locations to where the curves are "best" superposed onto the surface.

The rest of the paper is organized as follows. In Chapter 2, we overview the related work on invariants, curvature estimation, surface registration, and object recognition. In Chapter 3, we describe the recognition of quadratic and cubic curves based on differential invariants. In Chapter 4, we present the recognition of closed-form and free-form objects based on registration of curve segments sampled on their surfaces. In Chapter 5, we summarize all the results.

# CHAPTER 2. RELATED WORK

In this chapter, we give an overview of previous work on invariants, curvature estimation, surface registration, and object recognition.

## 2.1 Invariants

Algebraic invariants are expressions of the coefficients of polynomial equations describing curved shapes. The foundation was due to Cayley, Sylvester, Young, and among others, Hilbert [22], who offered a procedure that constructs all independent algebraic invariants for a given curve or surface. In real applications, polynomials are fit to image data and their coefficients are extracted for invariant evaluation. Keren [29] and Forsyth *et al.* [15] presented efficient methods for finding algebraic invariants and demonstrated on recognition of real objects. Civi *et al.* [10] also conducted object recognition experiments with algebraic invariants of Euclidean, affine, and projective groups.

One drawback of algebraic invariants is the requirement of global shape data. This is almost impossible to provide by a touch sensor, or by a vision system in case of occlusion.

Differential invariants depend on local data and deal with situations like occlusion well. They are functions of curvature and torsion and their derivatives. Up till now, vision- and invariant-based recognition has focused on differential invariants that are independent of various transformation groups but not of point locations on a shape. For recognition, multiple invariants are computed along the shape boundary and plotted against each other to serve as the shape's signature.

Calabi *et al.* [5] introduced the "signature curve" that is invariant to Euclidean or affine transformation. Rivlin and Weiss [41] derived differential invariants for a shape by applying

to its quartic fit the same transformation that turns an osculating curve (a cubic) into the canonical form, and extracted the resulting coefficients to plot a signature curve for recognition.

Semi-differential invariants combine global constraints and local information to ease the correspondence issue faced by non-invariant-based methods and also relieve the burden on estimating higher order derivatives for differential invariants. The theoretical foundation for this type of invariants was presented by Moons *et al.* [36]. Pajdla and Van Gool [37] used simple semi-differential invariants to match curves extracted from range data in the presence of partial occlusion.

## 2.2 Curvature Estimation

Rusinkiewicz [42] generalizes a method of estimating a normal at a vertex, which takes an average of the normals of incident faces, to estimate principal curvatures at the vertices of triangular meshes. Curvature estimates are computed from an average of the accumulated curvature tensor at a vertex, which is calculated using finite differences of estimated normals. This method can also be generalized to estimate higher-order derivatives of a curvature.

Taubin [49] constructs a matrix, expressed as an integral, at a point on a surface. The eigenvectors of the matrix form a Darboux frame at the point. The eigenvalue corresponding to the normal is always zero. Two principal curvatures are the linear combinations of eigenvalues corresponding to two principal directions, respectively. Chen and Schmitt [7] estimate normal curvatures at a point on a surface in three or more tangent directions. By Euler's theorem, this gives a system of equations in terms of two principal curvatures and an angle between the tangent and one of the principal directions. This system has a direct solution in the case of three equations or can be solved as least-squares when more than three equations exist. These two methods have later been modified to deal with real noisy range data [20].

In analytical approaches [14, 46, 32, 12], a local patch is fitted over a point on a surface and its geometric neighbors. Usually, quadratic or cubic surfaces are used in a fit. Principal curvatures are analytically calculated from the fitted patch.

An angle deficit [46, 32] is the summation of all angles around a vertex subtracted from

$2\pi$. At the vertex, the ratio of the angle deficit to one-third of the sum of areas of all incident triangles approximates the Gaussian curvature. Since no information about a mean curvature is provided, the angle deficit method cannot be used to estimate principal curvatures.

In [48], several methods, including Taubin's [49], Chen and Schmitt's [7] and angle deficit [46, 32], to estimate Gaussian and mean curvatures on triangular meshes were compared. According to the results, angle deficit is the best and paraboloid fitting is the second best methods to approximate the Gaussian curvature. For mean curvature estimation, paraboloid fitting is the best approach. So, paraboloid fitting is the best method to estimate two principal curvatures and will be used in the paper.

## 2.3 Surface Registration

In 3D registration, one of the most important tasks is to compute an optimal rigid transformation from sensed shape to model shape which involves a minimization over 6 degrees of freedom. If the correspondences between sensed and model data points are known, the optimal transformation has a closed-form least-squares solution which uses quaternions to represent rotations [13, 23, 3, 44].

Besl and McKay [3] have developed an iterative closest point (ICP) algorithm for 3D shape registration. ICP works for different data representations: point sets, polylines, implicit and parametric curves and surfaces, and triangular meshes. The algorithm iteratively computes an optimal transformation from sensed data to model data. ICP always converges to a local minimum, but the correct registration between model and data is not guaranteed.

Chua and Jarvis [8] select three dispersed data points on a sensed surface. Principal curvature, Darboux frame and distance constraints are used to find the three corresponding model points. Many possible 3-tuples on the model are found. A heuristic search is used to single out the optimal transformation from the sensed 3-tuple to the model 3-tuple in low order time.

## 2.4 Object Recognition

A representation is a key in object recognition from range data. The representation should have the following qualities: invariance to rigid transformations, ability to represent free-form objects, ability to handle occlusion and clutter, robustness to noise, and efficiency in computation [34, 45].

A number of surface representation schemes were developed in computer vision. They include splashes [45], spherical attribute images [21], COSMOS [11], point signatures [9], spin images [28], surface signatures [50], point fingerprints [47], regional point descriptors [16], and salient geometric features [17]. A survey of recent free-form object representation and recognition techniques is available in [6].

Object recognition methods based on hash tables [31, 35] consist of two phases: offline and online. During an offline phase, a hash table is constructed from certain features derived from all models in the database. During an online recognition phase, the similar features are extracted from a scene and used to index the hash table casting a vote for a model in the record. The models with the most votes are hypothesized to be present in the scene. The hypothesis is verified by transforming these models and the scene into a common coordinate frame and selecting the best matching model.

In touch sensing, the domain of object recognition have so far been limited to polyhedra, convex objects, quadrics, and superquadrics.

Gaston and Lozano-Perez [18] recognize a polyhedra on a plane (having 3 degrees of freedom) from tactile data. They use an *interpretation tree* which shows the range of possible pairings of contact points and object's facets. Distance, angle and model constraints are used to prune the tree removing a significant number of interpretations that are inconsistent with input data. In case of ambiguities after applying all the constraints, additional contact points are obtained to recognize the object from among a set of known objects. Grimson and Lozano-Perez [19] have later generalized this method to recognize polyhedra with 6 degrees of freedom. The exponential size of the interpretation tree makes the method computationally expensive for polyhedra with very large number of facets. Therefore, this approach is practically inapplicable

to free-form objects represented as triangular meshes.

In [2], internal and external volumetric approximations of a convex object are built from sparse tactile data. Internal volume is approximated by the convex hull of contact points. External volume is initially a workspace box. With each contact, the external volume is recursively reduced by removing the external semispace of the contact plane. As the number of contact points increases, the internal volume grows and the external volume shrinks, eventually approximating the volume of the object. The main disadvantage of this method is its applicability to convex objects only. Also, in order to get good volumetric approximations, the contact points need to be gathered from all over the surface, which is very hard to accomplish with tactile sensors. In contrast, our method presented in this paper is applicable to free-form objects and needs only local tactile data.

Allen and Roberts [1] fit a superquadric over sparse tactile data. The recovered parameters of the superquadric are matched against a set of model parameters. Keren *et. al.* [30] derives differential invariants for curves lying on surfaces of spheres, cylinders, cones, and tori. The invariants are the expression in terms of curvature, torsion and their higher order derivatives which need to be estimated from tactile data. Differential invariants are usually shape specific and not applicable for the recognition of general objects.

To the best of our knowledge, there is no previous work on recognizing *free-form* objects from tactile data.

# CHAPTER 3. RECOGNITION OF ALGEBRAIC CURVES

In this chapter, we describe the invariant-based recognition of algebraic curves. First, we discuss the basics of curve geometry in Section 3.1. Then, we derive differential and semi-differential invariants for quadratic and cubic curves in Section 3.2. Next, we describe how to recognize the curves with invariants and locate the contact points on them in Section 3.3. Finally, we demonstrate the invariant-based recognition with simulations and experiments in Section 3.4.

## 3.1 Curve Geometry

The touch sensor in contact with a 2D object can "feel" its local geometry, which is described by the curvature. At the contact point, denote by $\phi$ the tangential angle formed by the tangent with the $x$-axis. The *curvature* $\kappa$ is the rate of change of $\phi$ with respect to arc length $s$, that is,

$$\kappa = \frac{d\phi}{ds}.$$

Curvature is independent of parametrization, rotation, and translation. For the parametric curves, in the form $\alpha(t) = (x(t), y(t))$ the curvature is given by [43, p. 136]

$$\kappa(t) = \frac{x'y'' - x''y'}{(x'^2 + y'^2)^{3/2}}. \tag{3.1}$$

The touch sensor estimates the change of geometry with respect to arc length only. For this reason, we are interested in the derivative of the curvature with respect to arc length:

$$\kappa_s = \frac{d\kappa}{dt}\frac{dt}{ds} = \frac{\kappa'(t)}{(x'^2 + y'^2)^{1/2}}. \tag{3.2}$$

Section 3.4.2 will look at how the curvature and its derivative can be reliably estimated from real data. Until then, we just assume that these two quantities are measurable.

### 3.1.1 Signature Curve

The *Euclidean signature curve* of a curve $\alpha(t)$ is the set of all points $(\kappa(t), \kappa_s(t))$ evaluated along the curve. An example is shown in Figure 3.1. The following result is well known [5]:


(a) (b)

Figure 3.1 (a) A cubical parabola $y = 0.6x^3 + 0.4x$; (b) its signature curve.

**Theorem 1** *Two smooth curves are equivalent up to an Euclidean transformation if and only if their signature curves are identical.*

The above result has led to the development of shape recognition methods [41, 38, 5] based on matching signature curves. Construction of the signature curve, nevertheless, requires global shape information, which the touch sensor does not provide. So, our aim is to make use of the local geometry at a small number of points to perform the recognition task.

## 3.2 Differential and Semi-differential Invariants

We will derive differential and semi-differential invariants for algebraic curves. These invariants are the expressions in terms of the curvature and its derivative with respect to arc length at one or two points. Some invariants depend on the slope as well. Our focus will be on quadratic and cubic curves.

### 3.2.1 Quadratics

A quadratic curve has the general form:[1]

$$\alpha x^2 + 2\beta xy + \gamma y^2 + 2\delta x + 2\epsilon y + \zeta \;=\; 0. \tag{3.3}$$

It is known that through proper rotation and translation equation (3.3) can be transformed into one of the following canonical forms:

$$\begin{aligned} \frac{x^2}{a^2} + \frac{y^2}{b^2} &= 1, && \text{if } \alpha\gamma - \beta^2 > 0; \\ \frac{x^2}{a^2} - \frac{y^2}{b^2} &= 1, && \text{if } \alpha\gamma - \beta^2 < 0; \\ y^2 &= 4ax, && \text{if } \alpha\gamma - \beta^2 = 0. \end{aligned}$$

All quadratic curves are thus classified into three classes: ellipses, hyperbolas, and parabolas. Together they are referred to as the *conics*.

A curve can be in any position and orientation. Since our recognition strategy is based on local information, which is independent of rotation and translation, we can always transform a quadratic curve into its canonical form.

To recognize a conic, we first identify which class it belongs to and then recover the shape parameters $a$ and $b$ in the equation that describes the curve. For convenience, we will start with some parametrization instead and derive an expression that is independent of the parametrization. This expression is based on local geometry at two points, namely, their curvatures and derivatives with respect to arc length. However, its value is independent of specific points.

#### 3.2.1.1 Parabola

Parabolas are identified with all the curves parametrized by quadratic polynomials:

$$x = a_2t^2 + a_1t + a_0 \quad \text{and} \quad y = b_2t^2 + b_1t + b_0. \tag{3.4}$$

[1]The determinant

$$\Delta = \det \begin{pmatrix} \alpha & \beta & \delta \\ \beta & \gamma & \epsilon \\ \delta & \epsilon & \zeta \end{pmatrix} \neq 0$$

but $\Delta/(\alpha+\gamma) < 0$ when $\alpha\gamma - \beta^2 > 0$.

To verify that the above two equations describe a parabola, we treat $x$ and $y$ as constants and rewrite these equations as polynomials in $t$:

$$\begin{aligned} a_2t^2 + a_1t + a_0 - x &= 0, \\ b_2t^2 + b_1t + b_0 - y &= 0. \end{aligned}$$

Taking the resultant of two polynomials will give us an implicit quadratic equation in the form of (3.3). The coefficients $\alpha$, $2\beta$, and $\gamma$ of terms $x^2$, $xy$, and $y^2$ are $b_2^2$, $-2a_2b_2$, and $a_2^2$ respectively. From $\alpha\gamma - \beta^2 = b_2^2a_2^2 - (-a_2b_2)^2 = 0$, we know that the curve is indeed a parabola. This also shows us that polynomial parametric curves are just a proper subset of polynomial algebraic curves. As for quadratic curves, ellipses and hyperbolas can not be parametrized using polynomials.

One curve can be parametrized in many different ways. The general parametrization of parabola in (3.4) involves six shape parameters, namely, $a_2$, $a_1$, $a_0$, $b_2$, $b_1$, and $b_0$. To recognize a parabola in this form, we have to solve for all six shape parameters. This can be very difficult. To simplify our job, we will try to reduce the number of parameters while not changing the geometry of curve. Since we are interested in the recovery of the shape of the curve, we are not bounded by one particular parametrization. Moreover, the method we use to recognize the curve does not assume particular position and orientation. So, we have the freedom to translate, rotate, and reparametrize the curve in order to get its simplest parametric form.

To obtain simpler form for the parabola, we first rotate the curve by $\theta$:

$$\begin{aligned} x &= \Big(a_2t^2 + a_1t + a_0\Big)\cos\theta + \Big(b_2t^2 + b_1t + b_0\Big)\sin\theta, \\ y &= -\Big(a_2t^2 + a_1t + a_0\Big)\sin\theta + \Big(b_2t^2 + b_1t + b_0\Big)\cos\theta. \end{aligned}$$

To cancel the leading term of $y$, we choose rotational angle $\theta$ such that $-a_2\sin\theta + b_2\cos\theta = 0$, or $\tan\theta = b_2/a_2$. Then, after some reparametrization and translation the equation of the parabola reduces to its simplest form:

$$\begin{aligned} x &= at^2, \\ y &= 2at, \end{aligned}$$

where

$$a = \frac{(-a_1 \sin\theta + b_1 \cos\theta)^2}{(a_2 \cos\theta + b_2 \sin\theta)}.$$

The above parametric form has only one shape parameter, and it satisfies the canonical implicit equation of the parabola, which is $y^2 = 4ax$. From now on, we will use this simplest parametric form to derive an invariant for the parabola.

From equation (3.1) the curvature for the parabola will be:

$$\kappa = -\frac{1}{2a(t^2+1)^{3/2}}. \tag{3.5}$$

The speed of the curve is $v(t) = \sqrt{(x'^2 + y'^2)} = 2a\sqrt{(t^2+1)}$, so the derivative of curvature with respect to arc length is:

$$\kappa_s = \frac{\kappa'}{v(t)} = \frac{3t}{4a^2(t^2+1)^3}. \tag{3.6}$$

From equation (3.5) we can obtain the expression for $t^2$:

$$t^2 = \frac{1}{(2a\kappa)^{2/3}} - 1. \tag{3.7}$$

Taking the square of left and right hand sides of (3.6) will give us:

$${\kappa_s}^2 = \frac{9t^2}{16a^4\,(t^2+1)^6}. \tag{3.8}$$

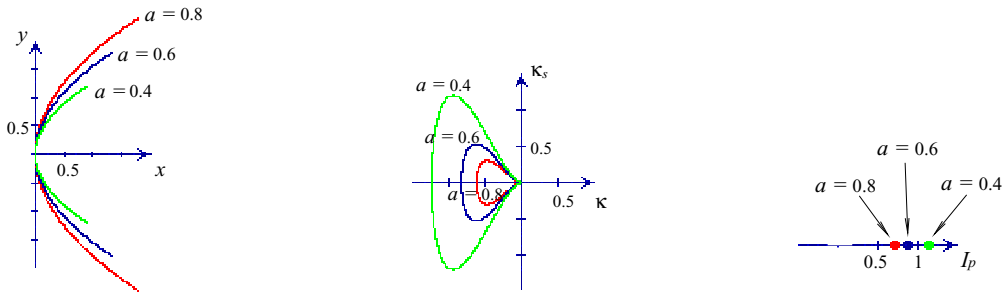
We can eliminate $t$ by substituting $t^2$ in (3.7) into (3.8), then we will get an equation that describes the signature curve of the parabola:

$$\kappa^{2/3}\left(\frac{\kappa_s^2}{9\kappa^4} + 1\right) = \frac{1}{(2a)^{2/3}}. \tag{3.9}$$

Denote the left hand side of (3.9) by $I_{\mathrm{p}}$, we get our invariant for the parabola:

$$I_{\mathrm{p}}(\kappa, \kappa_s) \equiv \kappa^{2/3}\left(\frac{\kappa_s^2}{9\kappa^4} + 1\right). \tag{3.10}$$

The expression $I_{\mathrm{p}}(\kappa, \kappa_s)$ has value independent of $t$. It is an invariant which has a one-to-one correspondence to the shape of the parabola. Figure 3.2 illustrates three parabolas distinguished by $I_{\mathrm{p}}$. Since $\kappa$ and $\kappa_s$ are measurable, from (3.9) we can determine the shape parameter $a$.




(a) (b) (c)

Figure 3.2 (a) Three parabolas in the form $y^2 = 4ax$; (b) their signature curves $\{(\kappa, \kappa_s)\}$; (c) corresponding values of the invariant $I_p$. The invariant is evaluated using any point on a signature curve.

#### 3.2.1.2 Ellipse

Let us start with the canonical parametrization:

$$\begin{aligned} x &= a\cos(t), \\ y &= b\sin(t), \end{aligned}$$

which has the speed $v(t) = \sqrt{a^2\sin^2(t) + b^2\cos^2(t)}$. From equations (3.1) and (3.2) the curvature and its derivative with respect to arc length are

$$\kappa = \frac{ab}{\left(a^2\sin^2(t) + b^2\cos^2(t)\right)^{3/2}}, \tag{3.11}$$

$$\kappa_s = \frac{-3ab\left(a^2 - b^2\right)\sin(t)\cos(t)}{\left(a^2\sin^2(t) + b^2\cos^2(t)\right)^{3}}. \tag{3.12}$$

Next, we describe how to derive an invariant for the ellipse. By squaring both sides of (3.12) we get:

$${\kappa_s}^2 = \frac{9a^2b^2\left(a^2 - b^2\right)^2\sin^2(t)\cos^2(t)}{\left(a^2\sin^2(t) + b^2\cos^2(t)\right)^{6}}. \tag{3.13}$$

First, we eliminate $\cos(t)$ by substituting $\cos^2(t) = 1 - \sin^2(t)$ into (3.11) and (3.13), and get the following:

$$\kappa = \frac{ab}{\left(\left(a^2 - b^2\right)\sin^2(t) + b^2\right)^{3/2}}, \tag{3.14}$$

$${\kappa_s}^2 = \frac{9a^2b^2\left(a^2 - b^2\right)^2\sin^2(t)\left(1 - \sin^2(t)\right)}{\left(\left(a^2 - b^2\right)\sin^2(t) + b^2\right)^{6}}. \tag{3.15}$$

From equation (3.14) $\sin^2(t)$ will be:

$$\sin^2(t) = \frac{\left(\frac{ab}{\kappa}\right)^{2/3} - b^2}{a^2 - b^2}. \tag{3.16}$$

Then, we eliminate $\sin(t)$ by substituting $\sin^2(t)$ in (3.16) into (3.15), and after few more steps we obtain an equation in terms of $\kappa$, $\kappa_s$, $a$, and $b$, that describes the signature curve (see Figure 3.3(b)):

$$\kappa^{2/3}\left(\frac{\kappa_s^2}{9\kappa^4} + 1\right) = \frac{a^2 + b^2}{(ab)^{4/3}} - \frac{1}{(ab\kappa)^{2/3}}. \tag{3.17}$$

In equation (3.17), the values of $\kappa$ and $\kappa_s$ are measurable, so we have only two unknown quantities $a$ and $b$. This means that two points on the ellipse are enough to solve for the shape parameters $a$ and $b$.

First, we derive an invariant for the ellipse using two points. Let $\kappa_i$ and $\kappa_{si}$, $i = 1, 2$, be the curvature and its derivative at the $i$th point. We end up with two equations in the form of (3.17). Subtracting one of them from the other yields the following (assuming $\kappa_1 \neq \kappa_2$):

$$\kappa_1^{2/3}\left(\frac{\kappa_{s1}^2}{9\kappa_1^4} + 1\right) - \kappa_2^{2/3}\left(\frac{\kappa_{s2}^2}{9\kappa_2^4} + 1\right) = \frac{1}{(ab\kappa_2)^{2/3}} - \frac{1}{(ab\kappa_1)^{2/3}}.$$

Taking $1/(ab)^{2/3}$ out of parenthesis gives:

$$\kappa_1^{2/3}\left(\frac{\kappa_{s1}^2}{9\kappa_1^4} + 1\right) - \kappa_2^{2/3}\left(\frac{\kappa_{s2}^2}{9\kappa_2^4} + 1\right) = \frac{1}{(ab)^{2/3}}\frac{\kappa_1^{2/3} - \kappa_2^{2/3}}{(\kappa_1\kappa_2)^{2/3}}.$$

Finally, we get the following:

$$\frac{(\kappa_1\kappa_2)^{2/3}}{\kappa_1^{2/3} - \kappa_2^{2/3}}\left(\kappa_1^{2/3}\left(\frac{\kappa_{s1}^2}{9\kappa_1^4} + 1\right) - \kappa_2^{2/3}\left(\frac{\kappa_{s2}^2}{9\kappa_2^4} + 1\right)\right) = \frac{1}{(ab)^{2/3}}. \tag{3.18}$$

Write the left hand side of (3.18) as

$$I_{c1}(\kappa_1, \kappa_2, \kappa_{s1}, \kappa_{s2}) \equiv \frac{(\kappa_1\kappa_2)^{2/3}}{\kappa_1^{2/3} - \kappa_2^{2/3}}\left(\kappa_1^{2/3}\left(\frac{\kappa_{s1}^2}{9\kappa_1^4} + 1\right) - \kappa_2^{2/3}\left(\frac{\kappa_{s2}^2}{9\kappa_2^4} + 1\right)\right). \tag{3.19}$$

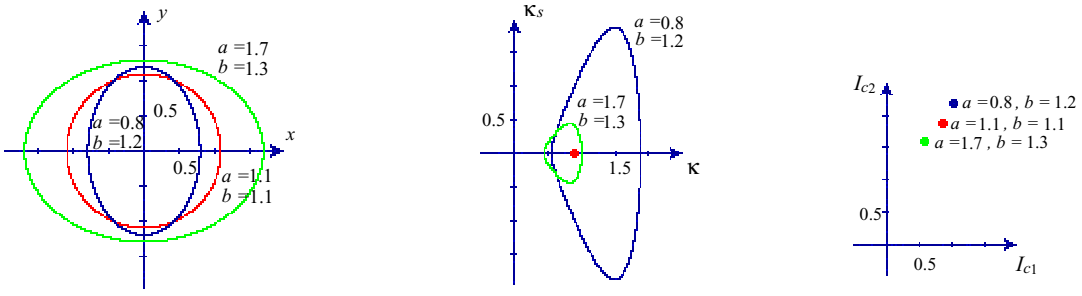
The expression $I_{c1}$ is a semi-differential invariant, since it involves the geometry at more than one point. Its value $1/(ab)^{2/3}$ is independent of the two points that are used, and it is always positive.

The invariant $I_{c1}$ alone cannot distinguish ellipses with the same product $ab$, or equivalently, with the same area. So, we find a second invariant by substituting $I_{c1}$ for $1/(ab)^{2/3}$ into equation (3.17):

$$\kappa_1^{2/3}\left(\frac{\kappa_{s1}^2}{9\kappa_1^4+1}\right)+\frac{I_{c1}(\kappa_1,\kappa_2,\kappa_{s1},\kappa_{s2})}{\kappa_1^{2/3}}=\frac{a^2+b^2}{(ab)^{4/3}}$$

From left hand side of the above equation we get second invariant:

$$I_{c2}(\kappa_1,\kappa_2,\kappa_{s1},\kappa_{s2})\equiv\frac{1}{\kappa_1^{2/3}-\kappa_2^{2/3}}\left(\kappa_1^{4/3}\left(\frac{\kappa_{s1}^2}{9\kappa_1^4}+1\right)-\kappa_2^{4/3}\left(\frac{\kappa_{s2}^2}{9\kappa_2^4}+1\right)\right). \tag{3.20}$$




(a) (b) (c)

Figure 3.3 (a) Three ellipses in the form $x^2/a^2+y^2/b^2=1$; (b) their signature curves (the one for the circle with $a=b=1.1$ degenerates into a point $(1/1.1,0)$); (c) corresponding values of the invariant pair $(I_{c1},I_{c2})$. The invariants are evaluated using any two points on the same signature curve.

A one-to-one correspondence exists between the tuples $(I_{c1},I_{c2})$ and $(a,b)$. Figure 3.3 compares two ellipses and a circle distinguished by the invariants $I_{c1}$ and $I_{c2}$. To recover the shape parameters $a$ and $b$, we let $\mu_1=ab$. This product can be calculated from $I_{c1}$. Let $\mu_2=a^2+b^2$, which can be calculated from $I_{c2}$. From $\mu_1$ and $\mu_2$ it is easy to obtain the following:

$$\begin{aligned} a &= \sqrt{\frac{\mu_2+\sqrt{\mu_2^2-4\mu_1^2}}{2}},\\ b &= \sqrt{\frac{\mu_2-\sqrt{\mu_2^2-4\mu_1^2}}{2}}. \end{aligned}$$

Hence the ellipse is completely determined.

#### 3.2.1.3 Hyperbola

A hyperbola has the canonical parametric form

$$\begin{aligned} x &= a\cosh(t) = a\,\frac{e^t + e^{-t}}{2}, \\ y &= b\sinh(t) = b\,\frac{e^t - e^{-t}}{2}. \end{aligned}$$

The curvature and its derivative with respect to arc length are

$$\kappa = \frac{-ab}{(a^2\sinh^2(t) + b^2\cosh^2(t))^{3/2}}, \tag{3.21}$$

$$\kappa_s = \frac{3ab(a^2+b^2)\sinh(t)\cosh(t)}{(a^2\sinh^2(t) + b^2\cosh^2(t))^{3}}. \tag{3.22}$$

Using equations (3.21), (3.22), and $\cosh^2(t) = \sinh^2(t) + 1$, we eliminate $\cosh(t)$ and $\sinh(t)$ and obtain the following:

$$\kappa^{2/3}\left(\frac{\kappa_s^2}{9\kappa^4} + 1\right) = \frac{a^2 - b^2}{(ab)^{4/3}} + \frac{1}{(ab\kappa)^{2/3}}, \tag{3.23}$$

which is defined for one point on the hyperbola.

Taking the curvatures and derivatives at two points on the hyperbola, from the two copies of equation (3.23) we derive

$$I_{c1}(\kappa_1,\kappa_2,\kappa_{s1},\kappa_{s2}) \equiv \frac{(\kappa_1\kappa_2)^{2/3}}{\kappa_1^{2/3} - \kappa_2^{2/3}}\left(\kappa_1^{2/3}\left(\frac{\kappa_{s1}^2}{9\kappa_1^4} + 1\right) - \kappa_2^{2/3}\left(\frac{\kappa_{s2}^2}{9\kappa_2^4} + 1\right)\right) = -\frac{1}{(ab)^{2/3}}. \tag{3.24}$$

The invariant $I_{c1}$ for the hyperbola is the same expression as for the ellipse, but it's values is in different expression of $a$ and $b$. From (3.24) we can see that $I_{c1}$ is always negative for the hyperbola. The second invariant $I_{c2}$ is also the same expression as for ellipse, but with different values in terms of $a$ and $b$:

$$I_{c2}(\kappa_1,\kappa_2,\kappa_{s1},\kappa_{s2}) \equiv \frac{1}{\kappa_1^{2/3} - \kappa_2^{2/3}}\left(\kappa_1^{4/3}\left(\frac{\kappa_{s1}^2}{9\kappa_1^4} + 1\right) - \kappa_2^{4/3}\left(\frac{\kappa_{s2}^2}{9\kappa_2^4} + 1\right)\right) = \frac{a^2 - b^2}{(ab)^{4/3}}.$$

The invariants $I_{c1}$ and $I_{c2}$ completely determine the hyperbola. To solve for shape parameters $a$ and $b$, let $\mu_1 = ab$ and $\mu_2 = a^2 - b^2$. The values of $\mu_1$ and $\mu_2$ can be calculated from $I_{c1}$ and $I_{c2}$, respectively. Then we can find $a$ and $b$ as

$$\begin{aligned} a &= \sqrt{\frac{\sqrt{\mu_2^2 + 4\mu_1^2} + \mu_2}{2}}, \\ b &= \sqrt{\frac{\sqrt{\mu_2^2 + 4\mu_1^2} - \mu_2}{2}}. \end{aligned}$$

#### 3.2.1.4 Invariants for Conics

Both $I_{c1}$ and $I_{c2}$ are also invariants for a parabola. Since $I_p$ defined in (3.10) is an invariant for parabola from (3.9) we have:

$$\kappa_1^{2/3}\left(\frac{\kappa_{s1}^2}{9\kappa_1^4}+1\right)=\kappa_2^{2/3}\left(\frac{\kappa_{s2}^2}{9\kappa_2^4}+1\right)=\frac{1}{(2a)^{2/3}}.$$

Then the values of $I_{c1}$ and $I_{c2}$ for the parabola will be:

$$\begin{aligned} I_{c1} &= \frac{(\kappa_1\kappa_2)^{2/3}}{\kappa_1^{2/3}-\kappa_2^{2/3}}\left(\frac{1}{(2a)^{2/3}}-\frac{1}{(2a)^{2/3}}\right)=0, \\ I_{c2} &= \frac{1}{\kappa_1^{2/3}-\kappa_2^{2/3}}\left(\kappa_1^{2/3}\frac{1}{(2a)^{2/3}}-\kappa_2^{2/3}\frac{1}{(2a)^{2/3}}\right)=\frac{1}{(2a)^{2/3}}. \end{aligned}$$

As we have seen three conic curves share the same invariants. So, how do we recognize a given conic curve then? It is the sign of invariant $I_{c1}$ that discriminates one conic from another. When the $I_{c1}$ is positive the curve is an ellipse, when it is negative the curve is a hyperbola, and when it is zero the curve is a parabola.

We estimate $\kappa$ and $\kappa_s$ at two points on the curve, and plug these values into $I_p$. If the value of $I_p$ is the same for both points, we know that the curve is a parabola.[2] If not, we estimate $\kappa$ and $\kappa_s$ at third point and test invariants $I_{c1}$ and $I_{c2}$ using three resulting pairs of $\kappa$ and $\kappa_s$. If the values of $I_{c1}$ and $I_{c2}$ stay constant, then the curve is either ellipse or hyperbola. The sign of $I_{c1}$ will tell us which curve it is. Finally, if the test on $I_{c1}$ and $I_{c2}$ fails, we conclude that the curve is not quadratic.

### 3.2.2 Cubics

There is no classification of all cubic curves. So, it seems very difficult to construct invariants for all of them. However, we are interested in recognizing cubic splines, whose continuity in curvature enables them to model curved shapes in graphics and geometric modeling. Every segment of a cubic spline has the general form:

$$\begin{aligned} x &= a_3t^3+a_2t^2+a_1t+a_0, \\ y &= b_3t^3+b_2t^2+b_1t+b_0. \end{aligned} \tag{3.25}$$

[2]Except for a degenerate case where the two points are on a non-parabolic curve and assume the same value for the invariant.

This section starts with two subclasses of cubic spline polynomials: cubical and semi-cubical parabolas, before move on to general cubic splines.

#### 3.2.2.1 Cubical Parabola

This class of curves has the parametric form:

$$\begin{aligned} x &= a_3t^3 + a_1t + a_0, \\ y &= b_3t^3 + b_1t + b_0, \qquad \text{where } a_3b_1 - a_1b_3 \neq 0. \end{aligned}$$

It is a subclass of cubic spline curves without quadratic terms in both $x$ and $y$ coordinates. First, we derive a simpler but equivalent form. Since either $a_3 \neq 0$ or $b_3 \neq 0$, without loss of generality we assume $b_3 \neq 0$. Rotate the curve by $\theta = \arctan(-\frac{a_3}{b_3})$. Then, after some reparametrization and translation the curve becomes:

$$\begin{aligned} x &= t, \\ y &= at^3 + bt, \end{aligned}$$

where

$$\begin{aligned} a &= \frac{-a_3\sin\theta + b_3\cos\theta}{(a_1\cos\theta + b_1\sin\theta)^3}, \\ b &= \frac{-a_1\sin\theta + b_1\cos\theta}{a_1\cos\theta + b_1\sin\theta}. \end{aligned}$$

Now, the curve has only two shape parameters $a$ and $b$ instead of six.

The curvature and its derivative with respect to arc length are given by

$$\begin{aligned} \kappa &= \frac{6at}{\left(1 + (3at^2 + b)^2\right)^{3/2}}, \\ \kappa_s &= \frac{6a\left(1 + (3at^2 + b)^2\right) - 108a^2t^2\left(3at^2 + b\right)}{\left(1 + (3at^2 + b)^2\right)^3}. \end{aligned}$$

Figure 3.1 plots an example of cubical parabola and its signature curve.

The parameter value $t$ is not a measurable quantity. So, we have to eliminate it from the equations. In the case of quadratic curves we were able to do so, and obtained an invariant.

But this is not easy for cubic curves. To derive some invariants, we first reparametrize the curve using the slope:

$$\lambda = \frac{y'}{x'} = 3at^2 + b. \tag{3.26}$$

The curvature and its derivative are rewritten in terms of the slope:

$$\begin{aligned} \kappa^2 &= \frac{12a(\lambda - b)}{(1+\lambda^2)^3} \\ \kappa_s &= \frac{6a(1+\lambda^2) - 36a\lambda(\lambda - b)}{(1+\lambda^2)^3}. \end{aligned}$$

We can obtain $a$ and $b$ using $\lambda$, $\kappa$, and $\kappa_s$:

$$a = \frac{(\kappa_s + 3\lambda\kappa^2)(1+\lambda^2)^2}{6} \equiv I_{\text{cp1}}(\lambda, \kappa, \kappa_s), \tag{3.27}$$

$$b = \lambda - \frac{\kappa^2(1+\lambda^2)}{2(\kappa_s + 3\lambda\kappa^2)} \equiv I_{\text{cp2}}(\lambda, \kappa, \kappa_s). \tag{3.28}$$

The expressions $I_{\text{cp1}}$ and $I_{\text{cp2}}$ are invariants of the cubical parabola provided that the slope $\lambda$ can be determined.

Next, we describe how we can obtain the slope. Suppose a straight jaw is mounted on the robot and it is able to make contact everywhere on the curve. The tangent rotation from one point to another is accurately measured as the jaw rotation by the robot. Thus if we know the slope at one point we can then get the slope at any point. Let $\phi_1$ and $\phi_2 = \phi_1 + \Delta\phi_{12}$ be the tangential angles at two different points on the curve, where $\Delta\phi_{12}$ is measured as the jaw rotation. Writing $\lambda_i = \tan\phi_i$, $i = 1, 2$, and $\delta_{12} = \tan(\Delta\phi_{12})$, we have the following equation relating the two slopes:

$$\lambda_2 = \frac{\lambda_1 + \delta_{12}}{1 - \lambda_1\delta_{12}}. \tag{3.29}$$

Measure the curvatures $\kappa_i$ and derivatives $\kappa_{si}$, $i = 1, 2$, at two points. From invariant $I_{\text{cp2}}$ we have:

$$\lambda_1 - \frac{\kappa_1^2(1+\lambda_1^2)}{2(\kappa_{s1} + 3\lambda_1\kappa_1^2)} = \lambda_2 - \frac{\kappa_2^2(1+\lambda_2^2)}{2(\kappa_{s2} + 3\lambda_2\kappa_2^2)} \tag{3.30}$$

Elimination of $\lambda_2$ from (3.29) and (3.30) results in a quartic polynomial:

$$d_4\lambda_1^4 + d_3\lambda_1^3 + d_2\lambda_1^2 + d_1\lambda_1 + d_0 = 0, \tag{3.31}$$

with coefficients:

$$\begin{aligned}
d_0 &= \kappa_{s1}\left(\kappa_2^2\left(5\delta_{12}^2-1\right)+2\kappa_{s2}\delta_{12}\right)+\kappa_1^2\left(3\kappa_2^2\delta_{12}+\kappa_{s2}\right),\\
d_1 &= 2\delta_{12}\left(\kappa_{s1}\left(3\kappa_2^2-\kappa_{s2}\delta_{12}\right)+2\kappa_1^2\left(3\kappa_2^2\delta_{12}+\kappa_{s2}\right)\right),\\
d_2 &= \kappa_{s1}\left(\kappa_2^2\left(5\delta_{12}^2-1\right)+2\kappa_{s2}\delta_{12}\right)+\kappa_1^2\left(18\kappa_2^2\delta_{12}-\kappa_{s2}\left(5\delta_{12}^2-1\right)\right),\\
d_3 &= 2\delta_{12}\left(\kappa_{s1}\left(3\kappa_2^2-\kappa_{s2}\delta_{12}\right)+2\kappa_1^2\left(3\kappa_2^2\delta_{12}+\kappa_{s2}\right)\right),\\
d_4 &= 5\kappa_1^2\delta_{12}\left(3\kappa_2^2-\kappa_{s2}\delta_{12}\right).
\end{aligned}$$

The values of $\kappa_1$, $\kappa_{s1}$, $\kappa_2$, $\kappa_{s2}$, and $\delta_{12}$ can all be estimated. So, by solving the quartic polynomial in (3.31), we find the value of $\lambda_1$. Then, using equations (3.27) and (3.28), we calculate the shape parameters $a$ and $b$ of the curve. The invariants $I_{\text{cp1}}$ and $I_{\text{cp2}}$ of the cubical parabola are different from the invariants for quadratic curves, their evaluation requires the solution of the slope ($\lambda_1$) at one point.

#### 3.2.2.2 Semi-Cubical Parabola

This class of curves is described by the equations:

$$\begin{aligned}
x &= a_3t^3+a_2t^2+a_0,\\
y &= b_3t^3+b_2t^2+b_0, \qquad \text{where } a_3b_2-a_2b_3\neq 0.
\end{aligned}$$

The equivalent canonical parametrization involves only two shape parameters:

$$\begin{aligned}
x &= t^2,\\
y &= at^3+bt^2, \qquad a\neq 0.
\end{aligned}$$

The slope is $\lambda = y'/x' = 3at/2+b$. So, this time we reparametrize the curve using

$$t=\frac{2(\lambda-b)}{3a},$$

and derive the curvature and its derivative as:

$$\begin{aligned}
\kappa &= \frac{9a^2}{8(\lambda-b)(1+\lambda^2)^{3/2}},\\
\kappa_s &= -\frac{81a^4\Big(1+\lambda^2+3\lambda(\lambda-b)\Big)}{64(\lambda-b)^3(1+\lambda^2)^3}.
\end{aligned}$$

From the above two equations we determine $a$ and $b$ in terms of $\lambda$, $\kappa$, and $\kappa_s$:

$$a = \sqrt{-\frac{8\kappa^3(1+\lambda^2)^{5/2}}{9(\kappa_s+3\lambda\kappa^2)}} \equiv I_{\text{scp1}}(\lambda,\kappa,\kappa_s), \tag{3.32}$$

$$b = \lambda + \frac{\kappa^2(1+\lambda^2)}{\kappa_s+3\lambda\kappa^2} \equiv I_{\text{scp2}}(\lambda,\kappa,\kappa_s). \tag{3.33}$$

The expressions $I_{\text{scp1}}$ and $I_{\text{scp2}}$ are the invariants for semi-cubical parabola. Using two points, from invariant $I_{\text{scp2}}$ we can set up an equation:

$$\lambda_1 + \frac{\kappa_1^2(1+\lambda_1^2)}{\kappa_{s1}+3\lambda_1\kappa_1^2} = \lambda_2 + \frac{\kappa_2^2(1+\lambda_2^2)}{\kappa_{s2}+3\lambda_2\kappa_2^2}. \tag{3.34}$$

After eliminating $\lambda_2$ from (3.34) and (3.29), we will again get a quartic polynomial

$$d_4\lambda_1^4 + d_3\lambda_1^3 + d_2\lambda_1^2 + d_1\lambda_1 + d_0 = 0,$$

whose coefficients are:

$$\begin{aligned}
d_0 &= \kappa_1^2\left(\kappa_{s2}+3\kappa_2^2\delta_{12}\right) - \kappa_{s1}\left(\kappa_{s2}\delta_{12}+\kappa_2^2\left(1+4\delta_{12}^2\right)\right),\\
d_1 &= \delta_{12}\left(\kappa_{s1}\left(\kappa_{s2}\delta_{12}-3\kappa_2^2\right) - 5\kappa_1^2\left(\kappa_{s2}+3\kappa_2^2\delta_{12}\right)\right),\\
d_2 &= \kappa_1^2\left(\kappa_{s2}\left(1+4\delta_{12}^2\right) - 9\kappa_2^2\delta_{12}\right) - \kappa_{s1}\left(\kappa_{s2}\delta_{12}+\kappa_2^2\left(1+4\delta_{12}^2\right)\right),\\
d_3 &= \delta_{12}\left(\kappa_{s1}\left(\kappa_{s2}\delta_{12}-3\kappa_2^2\right) - 5\kappa_1^2\left(\kappa_{s2}+3\kappa_2^2\delta_{12}\right)\right),\\
d_4 &= 4\kappa_1^2\delta_{12}\left(\kappa_{s2}\delta_{12}-3\kappa_2^2\right).
\end{aligned}$$

Solving the above quartic polynomial will give us $\lambda_1$, and subsequently $\lambda_2$. Then, from equations (3.32) and (3.33), we will obtain the values of shape parameters $a$ and $b$, respectively.

#### 3.2.2.3 Cubic Spline

The general parametric form of cubic spline given by (3.25). By proper rotation, translation, and reparametrization it can be simplified into the following form:

$$\begin{aligned}
x &= t^2,\\
y &= at^3 + bt^2 + ct,
\end{aligned} \tag{3.35}$$

where $a$, $b$, and $c$ are the shape parameters.

The slope expression for cubic spline is:

$$\lambda = \frac{3at^2 + 2bt + c}{2t}. \tag{3.36}$$

This time, we can not completely reparametrize this curve using slope. So, the expression for the curvature involves both $t$ and $\lambda$:

$$\kappa = \frac{3at^2 - c}{4t^3\,(1+\lambda^2)^{3/2}}. \tag{3.37}$$

The derivative of the curvature in terms of the $t$, $\lambda$, $\kappa$, and shape parameters will be:

$$\kappa_s = \frac{\kappa^2\,(1+\lambda^2)\,(3(\lambda - b) - 6at)}{(\lambda - b)^2 - 3ac} - 3\kappa^2\lambda. \tag{3.38}$$

Rewrite equations (3.36), (3.37) and (3.38) as polynomials in $t$:

$$\begin{aligned} 3at^2 + 2(b-\lambda)t + c &= 0, & (3.39)\\ 2Lt^3 - 3at^2 + c &= 0, & (3.40)\\ 6at + M\Big((b-\lambda)^2 - 3ac\Big) + 3(b-\lambda) &= 0, & (3.41) \end{aligned}$$

where

$$\begin{aligned} L &= 2(1+\lambda^2)^{3/2}\kappa,\\ M &= \frac{\kappa_s + 3\kappa^2\lambda}{(1+\lambda^2)\kappa^2}. \end{aligned}$$

Subtract (3.39) from (3.40) we will get:

$$Lt^2 - 3at - (b-\lambda) = 0. \tag{3.42}$$

We substitute $c$ in (3.41) with (3.39):

$$9a^2Mt^2 + 6a\Big(1 + M(b-\lambda)\Big)t + M(b-\lambda)^2 + 3(b-\lambda) = 0. \tag{3.43}$$

Next, the resultant of equations (3.42) and (3.43) is computed to eliminate $t$:

$$81Ma^4 + 18L\Big(1 + 3M(b-\lambda)\Big)a^2 + L^2(b-\lambda)\Big(M(b-\lambda) + 3\Big)^2 = 0. \tag{3.44}$$

Since $M$ can get very large when $\kappa$ is small, we divide the left hand side of (3.44) by $81Ma^4$ and denote the resulting expression as the function $g(a, b, \lambda)$.

With curvatures and derivatives estimated at $l \geq 3$ points, the shape parameters $a$, $b$, and the slope $\lambda_1$ at the first point can be estimated through a least-squares optimization:

$$\min_{a,b,\lambda_1} \sum_{i=1}^{l} g(a,b,\lambda_i)^2,$$

where $\lambda_i$ depends on $\lambda_1$ according to (3.29). To determine the third parameter $c$, we first eliminate the $t^2$ terms from (3.39) and (3.42) and then eliminate $t$ from the resulting equation and (3.43).

$$c = \frac{\Big(\Big(2(b-\lambda)L + 9a^2\Big)\Big(M(b-\lambda)+3\Big) - 18a^2\Big)(b-\lambda)}{3a\Big(2L + 2M(b-\lambda)L + 9a^2M\Big)}.$$

## 3.3 Curve Localization and Recognition

Using the invariants introduced in Sections 3.2.1 and 3.2.2, we can recognize the given curve. Since the canonical parametric form of the curve can be recovered, we also can localize the curve relative to touch sensor. Next, we describe how to determine the locations of the contacts on the curve.

### 3.3.1 Locating Contact

The parameter value $t$ determines the contact location on the curve with the touch sensor. Since the tangent at the contact is measurable, $t$ also determines the relative pose of the shape to the hand. The value of $t$ can be easily determined for the quadratic and cubic curves discussed in Sections 3.2.1 and 3.2.2. Below, we simply give its expressions:

$$t = \begin{cases} \frac{\kappa_s}{3\kappa^2}, & \text{if parabola;} \\ \sin^{-1}\left(\sqrt{\frac{\left(\frac{ab}{\kappa}\right)^{2/3} - b^2}{a^2 - b^2}}\right), & \text{if ellipse;} \\ \sinh^{-1}\left(\sqrt{\frac{\left(\frac{ab}{\kappa}\right)^{2/3} - b^2}{a^2 + b^2}}\right), & \text{if hyperbola;} \\ \pm\sqrt{\frac{\lambda - b}{3a}}, & \text{if cubical parabola;} \\ \frac{2(\lambda - b)}{3a}, & \text{if semi-cubical parabola;} \\ -\frac{M\big((b-\lambda)^2 - 3ac\big) + 3(b-\lambda)}{6a}, & \text{if cubic spline.} \end{cases}$$

In the case of a cubical parabola, the sign is determined based on the relative configuration of the two data points.

### 3.3.2 Recognition Tree

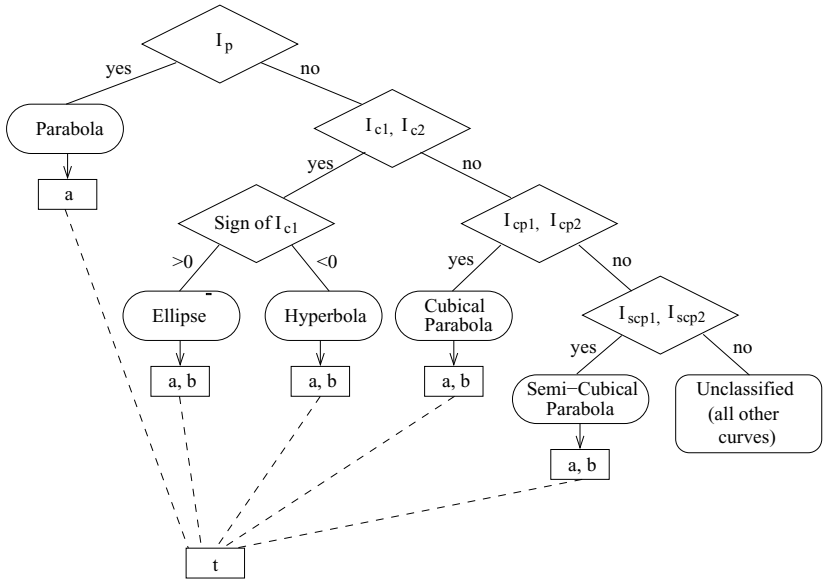



Figure 3.4 Recognition tree for quadratic and special cubic curves.

A general recognition strategy is illustrated using the tree in Figure 3.4. We estimate the values of $\kappa$ and $\kappa_s$ at as few as three points on the curve. Then, we test the invariants down the tree to identify the curve type or determine that it is unclassified. Next, we recover the shape parameters of the curve. Finally, we compute the parameter value $t$, which determines the contact on the curve.

To determine whether a curve is quadratic, we may measure $\kappa$ and $\kappa_s$ at three different points on the curve. Then evaluate the two invariants $I_{c1}$ and $I_{c2}$ according to (3.19) and (3.20) for each of the three resulting pairs. If the values of the two invariants do not change, there is a strong indication that the curve is a conic. For some curve, to satisfy the invariants, is the necessary condition to be from that curve family.

For example, consider the ellipse in Figure 3.5(a). The values of $\kappa$ and $\kappa_s$ are estimated at $t_1 = 0.36$, $t_2 = 1.86$, and $t_3 = 4.23$. Invariant $I_p$ has values 0.8971 and 0.4030 at the first

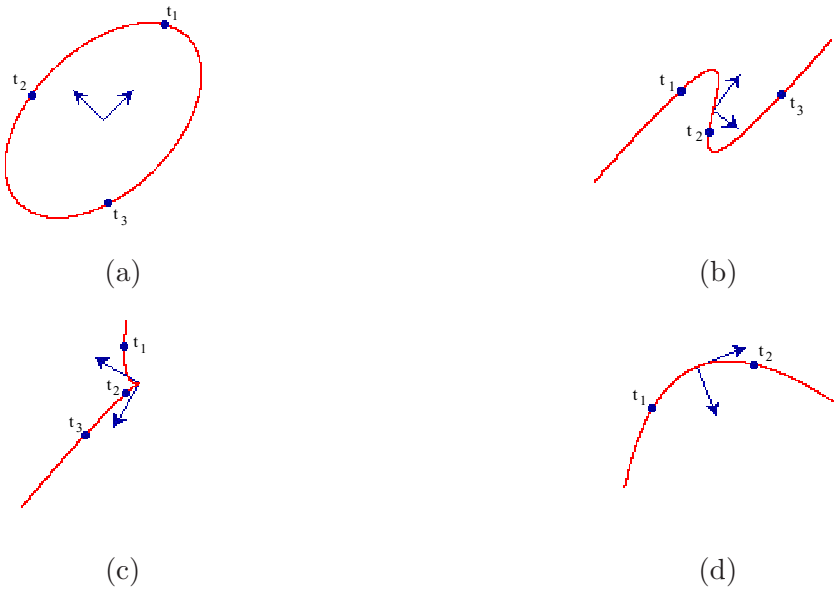



Figure 3.5 Recognition of four shapes based on local geometry at two or three points. (a) An ellipse with $a = 2.8605$ and $b = 1.7263$; (b) a cubical parabola with $a = 3.2543$ and $b = -2.3215$; (c) a semi-cubical parabola with $a = 2.5683$ and $b = 1.4102$; and (d) a parabola with $a = 0.8685$.

two points, so the curve is not a parabola. Invariant $I_{c1}$ yields values 0.3447, 0.3446, and 0.3449 at the three resulting pair of points, from which we infer that the curve is an ellipse. The recovered coefficients (from $I_{c1}$ and $I_{c2}$) are $a \approx 2.8609$, and $b \approx 1.7275$. The computed parameter values $t_1 \approx 0.36$, $t_2 \approx 1.86$, and $t_3 \approx 4.23$ are correct.

On the cubical parabola in Figure 3.5(b), estimates are taken at $t_1 = -0.86$, $t_2 = 0.19$, and $t_3 = 1.03$. Tests on both invariants $I_p$ and $I_{c1}$ have failed, so we know that the curve is not quadratic. Hypothesizing cubical parabola, we solve for $\lambda_1$ from (3.31). Subsequent tests on invariant $I_{cp1}$ (for shape parameter $a$) yield values 3.2244, 3.2536, and 3.1872, and on invariant $I_{cp2}$ (for $b$) yield values $-2.3237$, $-2.3237$, and $-2.3972$. The parameter values are estimated as $t_1 \approx -0.86$, $t_2 \approx 0.20$, and $t_3 \approx 1.04$.

Similarly, we have successfully recognized a semi-cubical parabola and a parabola as shown in Figure 3.5 (c) and (d), respectively.

## 3.4 Simulations and Experiments

In simulations, we approximate the curvature and its derivative by finite difference quotients:

$$\kappa \approx \frac{\phi(s+\Delta s)-\phi(s-\Delta s)}{2\Delta s} \quad \text{and} \quad \kappa_s \approx \frac{\phi(s+\Delta s)-2\phi(s)+\phi(s-\Delta s)}{(\Delta s)^2},$$

where $s$ and $\phi$ are arc length and tangential angle, respectively. The arc length between two points on the curve, close to each other, is approximated by their Euclidean distance. The rotation of the tangent from one point to another uses the exact value since it can be measured quite accurately in practice. [3]

### 3.4.1 Verification of Invariants

The first group of simulations were conducted to verify the invariants of all curve classes presented in this chapter. One curve out of each class was chosen, and for each curve, its invariants were evaluated 100 times using randomly selected points. The results are summarized in Table 3.1. Estimation errors of $\kappa$ and $\kappa_s$ were due to linear approximation. They showed up

Table 3.1 Invariant verification on five specific shapes. Each invariant is labeled with the equation where it is defined. The first row shows the real values of the invariants. The next three rows display the min, max, and mean of 100 values.

| inv. | $I_p$ | $I_{c1}$ (3.19) | | $I_{c2}$ (3.20) | | $I_{cp1}$ | $I_{cp2}$ | $I_{scp1}$ | $I_{scp2}$ |
|---|---|---|---|---|---|---|---|---|---|
| | (3.10) | ellipse | hyperbola | ellipse | hyperbola | (3.27) | (3.28) | (3.32) | (3.33) |
| real | 0.2198 | 0.1857 | −0.2678 | 1.2055 | 0.3222 | 6.9963 | 2.6127 | 1.3730 | 6.5107 |
| min | 0.2168 | 0.1801 | −0.2729 | 1.1749 | 0.2937 | 6.7687 | 1.7312 | 1.4111 | 6.3945 |
| max | 0.2230 | 0.1863 | −0.2655 | 1.2083 | 0.3615 | 7.0289 | 3.1684 | 1.4447 | 6.5834 |
| mean | 0.2198 | 0.1852 | −0.2675 | 1.2035 | 0.3210 | 6.9355 | 2.5022 | 1.4220 | 6.5154 |

in Table 3.1 as the discrepancies between actual invariant values and their estimates. Although three points on the curve are enough to recognize it, it would be more reliable to calculate the

[3] A method introduced in [5] approximates the osculating circle with one that passes through three local points. This curvature estimation scheme was extended in [4]. These methods are able to generate slightly better estimates than finite differencing but the simulation outcomes would not have been altered.

invariant at more points and take the mean value. As we can see from Table 3.1, the mean values of the invariants are close to real ones.

Having verified the correctness of invariants, we empirically demonstrate that the invariant of one curve class would not hold for another. This is necessary for the recognition strategy to work. Since all quadratic curves share the invariant $I_{c1}$, there are only three curve classes. We tested the invariants of one curve class against the data from another. The results are summarized in Table 3.2. From Table 3.2, we see that when an invariant is applied to curves

Table 3.2 Evaluating some invariants on data from curves of different classes. Each cell displays the summary over 100 values.

| \ data<br>inv.\ | conic<br>(ellipse) | cubical<br>parabola | semi-cub.<br>parabola | cubic<br>spline |
|---|---|---|---|---|
| $I_{c1}$ | ____ | −6.38(min)<br>−0.04(max)<br>−0.73(mean)<br>1.22(stdev) | −22.84<br>28.37<br>3.37<br>6.76 | −45.24<br>−4.94<br>−16.50<br>10.86 |
| $I_{cp2}$ | −11.97<br>15.46<br>−0.04<br>2.53 | ____ | 8.54<br>19.03<br>13.76<br>3.07 | 11.66<br>1721.04<br>55.52<br>217.34 |
| $I_{scp2}$ | −265.80<br>5.83<br>−3.22<br>26.75 | 7.80<br>65.22<br>29.17<br>17.19 | ____ | −150.68<br>1715.73<br>38.97<br>182.24 |

outside the curve class it was derived for, it has different values for different points. So, each invariant only holds for its own curve class.

Finally, we looked into how well a given curve can be recognized. In other words, we examined how much the recovered parameters $\bar{a}$, $\bar{b}$ and $\bar{c}$ would differ from the real ones $a$, $b$ and $c$. For measurement, we calculated the relative errors of recovered parameters with respect to real ones as $\sqrt{\left(\frac{a-\bar{a}}{a}\right)^2 + \left(\frac{b-\bar{b}}{b}\right)^2 + \left(\frac{c-\bar{c}}{c}\right)^2}$. The calculations used 100 different shapes from each family. For each recovered shape the relative error was calculated. The results are summarized in Table 3.3. From the table we see that on the average the relative errors are around 1%, except for cubic splines. These errors are likely due to finite differencing used for

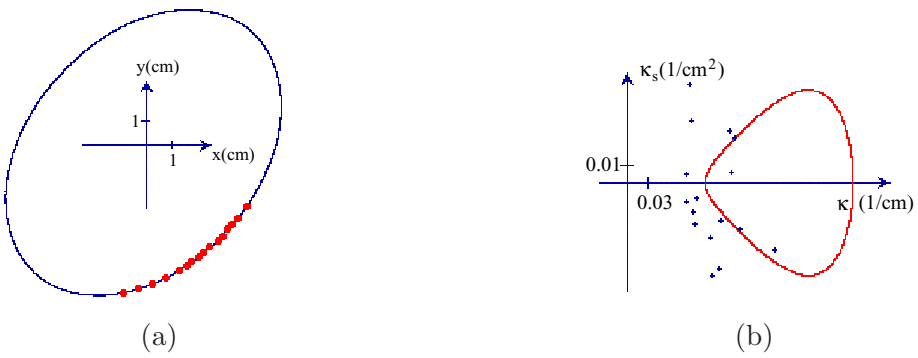
Table 3.3 Relative errors on estimating shape parameters. Summary over 100 different curves from each class (only 25 curves from the cubic spline class).

| | ellipse | hyperbola | parabola | cubical parabola | semi-cubical parabola | cubic spline |
|---|---|---|---|---|---|---|
| min | 0.02% | 0.10% | 0.01% | 0.02% | 0.04% | 0.61% |
| max | 7.99% | 9.71% | 3.35% | 7.49% | 8.09% | 29.23% |
| mean | 0.40% | 1.15% | 0.36% | 0.83% | 1.23% | 11.27% |

the estimation of the curvature and its derivative.

### 3.4.2 Experiments

We have assumed that the curvature and its derivative with respect to arc length can be estimated at any point on the curve. In simulations, finite differencing was used to estimate $\kappa$ and $\kappa_s$. We also tried to use finite differencing with real data. The curvature was estimated by the inverse of the radius of osculating circle as introduced in [5]. The derivative of the curvature was estimated by finitely differentiating the estimated curvature values as described in [4]. The results of such estimation are illustrated in Figure 3.6. From Figure 3.6, we see




(a) (b)

Figure 3.6 Estimating $\kappa$ and derivative $\kappa_s$ from tactile data using finite differencing: (a) an ellipse and 16 sample points; (b) estimates $(\hat{\kappa}, \hat{\kappa}_s)$ at these points plotted against the signature curve.

that the estimates of $\kappa$ and $\kappa_s$ are not accurate. The estimated values differ a lot from real ones. Since the curvature and its derivative are the second and the third order derivatives of the curve, respectively, they are very sensitive to noisy data. That's why the finite differencing

method failed in estimating $\kappa$ and $\kappa_s$. So, we used different method to reliably estimate $\kappa$ and $\kappa_s$.

Tactile data used in our experiments were generated by a joystick sensor mounted on an Adept Cobra 600 robot [33]. The joystick sensor gives a point location $(x, y)$ on the boundary of the shape, in the world coordinates. So, using such limited information, we have to come up with a method that reliably estimates $\kappa$ and $\kappa_s$. To estimate $\kappa$ and $\kappa_s$ at some point, we perform local fitting twice. The first fitting is done to get estimates of curvatures, and the second fitting is done to estimate derivatives.

Let $(x_0, y_0), (x_1, y_1), \ldots, (x_p, y_p)$ be a sequence of $p$ neighboring points, generated by joystick sensor on the boundary of the shape. Fit a quadratic curve over the first $n$ points in the sequence. [4] The resulting quadratic curve has the form:

$$y = a_2 x^2 + a_1 x + a_0.$$

We use the curvature

$$\kappa = \frac{2a_2}{\left(1 + (2a_2 x + a_1)^2\right)^{3/2}},$$

of this quadratic curve as the approximation to the curvature of the middle point in the sequence $(x_0, y_0), (x_1, y_1), \ldots, (x_{n-1}, y_{n-1})$. So, we get our first curvature value. Next, we shift the sequence of points by $l$, and fit a new quadratic curve over $(x_l, y_l), (x_{l+1}, y_{l+1}), \ldots, (x_{n-1+l}, y_{n-1+l})$. This will give us a second value of the curvature at the middle point of the new sequence. The arc length between two points, at which curvatures were estimated, can be found as

$$s = \int_{x_{\frac{n-1}{2}}}^{x_{\frac{n-1}{2}+l}} \sqrt{1 + (2a_2 x + a_1)^2}\, dx.$$

Since the function $\sqrt{1 + (2a_2 x + a_1)^2}$ is not integrable, we use Simpson's method [39, p. 117] to numerically estimate the arc length.

Now, we have two pairs $(0, \hat{\kappa}_0), (\hat{s}_1, \hat{\kappa}_1)$ of estimated arc length and curvature values. Repeating this process $m$ times, will generate a sequence of pairs $(0, \hat{\kappa}_0), (\hat{s}_1, \hat{\kappa}_1), \ldots, (\hat{s}_{m-1}, \hat{\kappa}_{m-1})$.

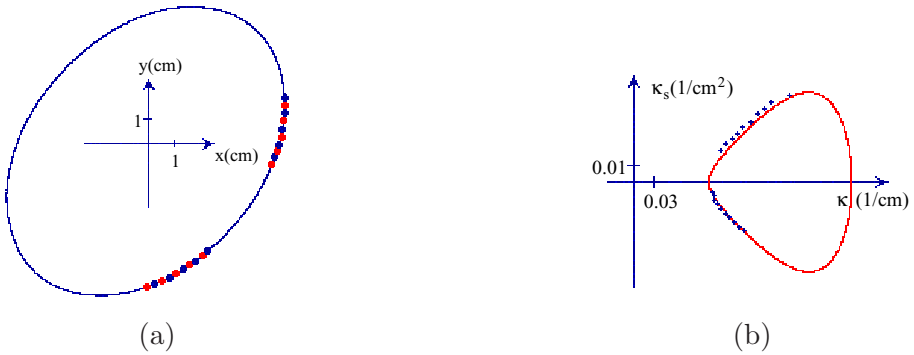
[4] A quadratic curve is chosen because it has the same degree as the osculating circle but does a better job at approximating a short curve segment.

By fitting a quadratic curve over this sequence we will get a curvature as a function of arc length

$$\kappa = b_2 s^2 + b_1 s + b_0.$$

Differentiating the above function, gives us curvature derivative estimates at $0, \hat{s}_1, \ldots, \hat{s}_{m-1}$. Finally, we end up with $m$ estimated pairs of $\kappa$ and $\kappa_s$ values, $(\hat{\kappa}_0, \hat{\kappa}_{s_0}), (\hat{\kappa}_1, \hat{\kappa}_{s_1}), \ldots, (\hat{\kappa}_{m-1}, \hat{\kappa}_{s_{m-1}})$.

The first experiment was done on the ellipse shown in Figure 3.7(a). Curvatures and derivatives were estimated at 20 sample points in two different parts of the ellipse. These $(\hat{\kappa}, \hat{\kappa}_s)$ estimates are plotted against the signature curve of the ellipse as shown in Figure 3.7(b). In ideal case, these $(\hat{\kappa}, \hat{\kappa}_s)$ estimates would lie on the signature curve. From the figure we can see that the estimated $(\hat{\kappa}, \hat{\kappa}_s)$ values are very close to real ones. Table 3.4 displays the summary




(a) (b)

Figure 3.7 Estimating $\kappa$ and derivative $\kappa_s$ from tactile data generated by a joystick sensor: (a) an ellipse and 20 sample points; (b) estimates $(\hat{\kappa}, \hat{\kappa}_s)$ at these points plotted against the signature curve.

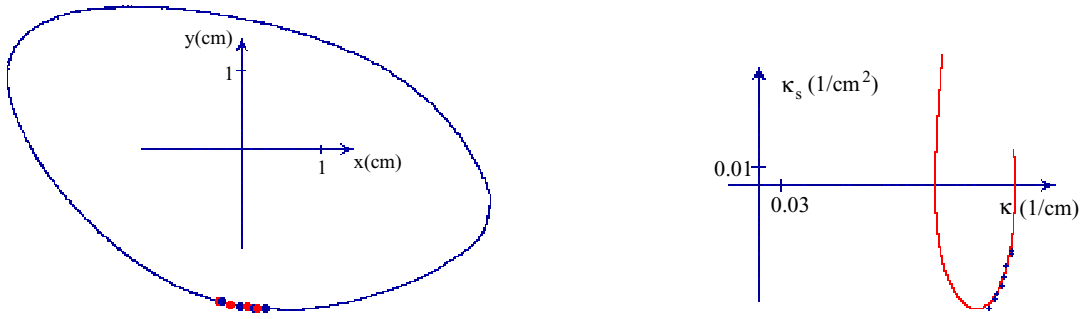
of evaluating the invariants $I_{c1}$ and $I_{c2}$ of the ellipse with the estimated $(\hat{\kappa}, \hat{\kappa}_s)$ values.

Table 3.4 Invariants $I_{c1}$, $I_{c2}$ and the recovered shape parameters $a$ and $b$ of the ellipse in Figure 3.7(a). Summary over 80 values computed from different pairs of estimated $(\hat{\kappa}, \hat{\kappa}_s)$ values.

| | $I_{c1}$ | $I_{c2}$ | $a$ | $b$ |
|---|---|---|---|---|
| real | 0.373836 | 1.30145 | 2.5 | 1.75 |
| min | 0.350559 | 1.26074 | 2.38636 | 1.62636 |
| max | 0.404903 | 1.36736 | 2.67234 | 1.83549 |
| mean | 0.377728 | 1.31825 | 2.51127 | 1.71959 |

The second experiment was performed on the closed cubic spline in Figure 3.8(a), which

consists of five cubic spline segments. Curvature and derivative estimates were taken at 8 sample points on one of the spline's segments. The estimated $(\hat{\kappa}, \hat{\kappa}_s)$ values are displayed in Figure 3.8(b) along with the signature curve of that segment.




(a) (b)

Figure 3.8 Estimating $\kappa$ and $\kappa_s$ from tactile data: (a) a closed cubic spline with 8 sample points from one of its segment; (b) estimates $(\hat{\kappa}, \hat{\kappa}_s)$ and the signature curve of the segment.

We used two groups of three estimates $(\hat{\kappa}, \hat{\kappa}_s)$ out of the eight from a cubic spline segment in Figure 3.8(b). Due to inefficiency in nonlinear optimization, only two tests were performed. The results are summarized in Table 3.5. Larger errors in the shape parameter estimation

Table 3.5 Recovering the cubic spline segment in Figure 3.8(a) from three of the eight data points. The segment is of the form (3.35).

| | $a$ | $b$ | $c$ |
|---|---|---|---|
| real | 1.10734 | 1.67996 | −0.401898 |
| test 1 | 1.07246 | 1.57648 | −0.39233 |
| test 2 | 1.06594 | 1.55496 | −0.44884 |

were observed, compared to the case of the elliptic part.

# CHAPTER 4. RECOGNITION OF CLOSED-FORM AND FREE-FORM SURFACES

In this chapter, we present registration-based recognition method. The method is described in Section 4.1 and it is applied to closed-form and free-form surfaces in Sections 4.2 and 4.3, respectively. Experimental results with various objects will follow in Section 4.4.

## 4.1 Registration and Recognition Scheme

To recognize the shape of an object, a robot acquires a number of points on its surface. Then, these data points are registered onto the all models in a database. The model yielding the best match is selected as the shape of the object.

To facilitate the matching process, the data points are acquired in a specific configuration which is described below.

### 4.1.1 Configuration of Data Points

The intersection of the object's surface with a plane is a space curve. Data points are sampled along three such space curves which intersect at one point $p$. Figure 4.1 displays an object with three sequences of data points on its surface. These point sequences are called the *data curves* and denoted by $\alpha$, $\beta$, and $\gamma$, respectively. We refer to the planes of the three data curves as *sampling planes*.

To register three data curves onto a model, which may or may not be for the object, we use a surface normal and two principal curvatures at the intersection point $p$. They are estimated in the following way.

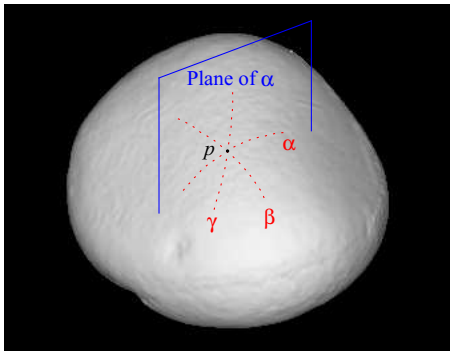



Figure 4.1 Three data curves $\alpha$, $\beta$, and $\gamma$ intersect at one point $p$ on a surface. Each data curve is sampled along the intersection of the surface and a plane. The sampling plane of $\alpha$ is displayed.

#### 4.1.1.1 Estimation of Surface Normal

In the sampling plane of each data curve, we fit a 2D quadratic parabola, $y = ax^2 + bx + c$, over a few data points around $p$. Figure 4.2 shows a quadratic fit for the data curve $\alpha$. The

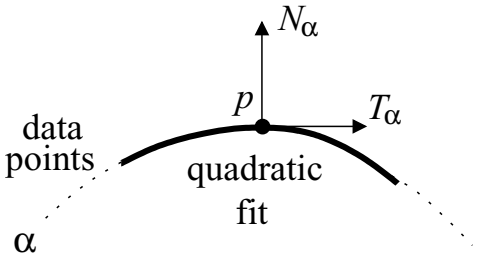



Figure 4.2 A quadratic curve fit over the neighborhood of $p$ in the sampling plane of $\alpha$.

normal $N_\alpha$, the tangent $T_\alpha$, and the curvature $\kappa_\alpha$ of $\alpha$ at $p$ are estimated by differentiating the fitted parabola. Similarly, we can estimate the normals $N_\beta$ and $N_\gamma$, the tangents $T_\beta$ and $T_\gamma$, and the curvatures $\kappa_\beta$ and $\kappa_\gamma$ of the data curves $\beta$ and $\gamma$ at $p$, respectively.

The surface normal $N$ at the intersection point $p$ is orthogonal to the three tangents $T_\alpha$, $T_\beta$, and $T_\gamma$. The cross product of any two of these tangents should give us the normal $N$. To get a more reliable estimate, we solve the following least-squares formulation [27],

$$\min_{||N||=1} (N \cdot T_\alpha)^2 + (N \cdot T_\beta)^2 + (N \cdot T_\gamma)^2. \tag{4.1}$$

#### 4.1.1.2 Estimation of Principal Curvatures

We estimate the principal curvatures at the curve intersection point $p$ using the method described in [27].

In case the curve normal $N_\alpha$ coincides with the surface normal $N$ at $p$, the curvature of $\alpha$ at $p$ is the normal curvature. If there is an angle between $N$ and $N_\alpha$, the normal curvature in the direction of $T_\alpha$ is given by [40, p. 128], $\kappa_n^{(\alpha)} = \kappa_\alpha \left(N_\alpha \cdot N\right)$. Similarly, the normal curvatures in the directions of $T_\beta$ and $T_\gamma$ are $\kappa_n^{(\beta)} = \kappa_\beta \left(N_\beta \cdot N\right)$ and $\kappa_n^{(\gamma)} = \kappa_\gamma \left(N_\gamma \cdot N\right)$, respectively.

By Euler's theorem [40, p. 137], the normal curvature can be written as a linear combination of principal curvatures $\kappa_1$ and $\kappa_2$,

$$\kappa_n^{(\alpha)} = \kappa_1 \cos^2\theta + \kappa_2 \sin^2\theta, \tag{4.2}$$

$$\kappa_n^{(\beta)} = \kappa_1 \cos^2(\theta+\theta_1) + \kappa_2 \sin^2(\theta+\theta_1), \tag{4.3}$$

$$\kappa_n^{(\gamma)} = \kappa_1 \cos^2(\theta+\theta_2) + \kappa_2 \sin^2(\theta+\theta_2), \tag{4.4}$$

where,

$$\theta_1 = \cos^{-1}\left(T_\alpha \cdot T_\beta\right),$$

$$\theta_2 = \cos^{-1}\left(T_\alpha \cdot T_\gamma\right),$$

and $\theta$ is the angle between $T_\alpha$ and principal direction corresponding to $\kappa_1$. We have three unknowns $\kappa_1$, $\kappa_2$, and $\theta$ in three equations (4.2), (4.3), and (4.4). First, the angle $\theta$ is determined as,

$$\theta = \frac{\tan^{-1}\left(\dfrac{\frac{\sin(\theta_1-\theta_2)}{\left(\kappa_n^{(\alpha)}-\kappa_n^{(\beta)}\right)\sin\theta_2}}{\frac{\left(\kappa_n^{(\alpha)}-\kappa_n^{(\gamma)}\right)\sin\theta_1}{}-\cos(\theta_1-\theta_2)}\right) - \theta_2}{2}.$$

Then, principal curvatures $\kappa_1$ and $\kappa_2$ at the intersection point $p$ of three data curves obtained as,

$$\kappa_1 = \frac{\kappa_n^{(\alpha)} \sin^2(\theta+\theta_1) - \kappa_n^{(\beta)} \sin^2\theta}{\cos^2\theta \sin^2(\theta+\theta_1) - \cos^2(\theta+\theta_1)\sin^2\theta}, \tag{4.5}$$

$$\kappa_2 = \frac{\kappa_n^{(\beta)} \cos^2\theta - \kappa_n^{(\alpha)} \cos^2(\theta+\theta_1)}{\cos^2\theta \sin^2(\theta+\theta_1) - \cos^2(\theta+\theta_1)\sin^2\theta}. \tag{4.6}$$

### 4.1.2 Registration of Data Curves

Let $M$ be a model to be matched against the data curves. We calculate principal curvatures at a set of points on the surface of $M$. This calculation depends on $M$ and will be described later for specific models. The values of the principal curvatures are stored in a table.

#### 4.1.2.1 Table Lookup

We look up the table to find the points with local geometries similar to that of the intersection point $p$ of the three data curves $\alpha$, $\beta$, and $\gamma$. Let $\kappa_{1i}$ and $\kappa_{2i}$ be principal curvatures at the $i$th point in the table. We select points which satisfy the following condition,

$$\sqrt{(\kappa_1 - \kappa_{1i})^2 + (\kappa_2 - \kappa_{2i})^2} < \delta, \tag{4.7}$$

where $\delta$ is a tolerance constant. Let $p_1, p_2, \ldots, p_m$ be the points selected. We call them the *candidate points*. The next step is to conduct local search, starting at each point $p_i$, for the best superposition of $\alpha$, $\beta$, and $\gamma$ onto $M$.

#### 4.1.2.2 Local Search

We superpose the data curves $\alpha$, $\beta$, and $\gamma$ onto the model $M$ by coinciding their intersection point $p$ with a candidate point $p_j$ and aligning the data normal $N$ at $p$ with the model surface normal at $p_j$. Let $q_i$, $1 \leq i \leq n$, be all the data points along $\alpha$, $\beta$, and $\gamma$. Denote by $d(q_i, M)$ the distance from $q_i$ to $M$. The *superposition error* $E$ is defined as the average distance from the data points to $M$:

$$E = \frac{1}{n} \sum_{i=1}^{n} d(q_i, M), \tag{4.8}$$

for the data curves in their current positions and orientations.

We minimize the superposition error by moving $p$ on the model surface, aligning the data normal $N$ with the corresponding model normal, and rotating $\alpha$, $\beta$, and $\gamma$ together about it. This minimization depends on $M$ and will be discussed later for closed-form and free-form models separately.

The *registration error* $\mathcal{E}(M)$ is the minimum of $E$ over all possible locations of $p$ and all orientations of $\alpha$, $\beta$, and $\gamma$ about the surface normal at each location. To obtain the registration error, we minimize the superposition error in the neighborhood of $p$ for each candidate point and select the minimum over the results.

### 4.1.3 Recognition

Let $M_1, M_2, \ldots, M_k$ be a database of models. We compute the registration error $\mathcal{E}$ on each model $M_i$. The model that yields the minimum registration error, which is

$$\mathcal{E}_{\min} = \min\{\mathcal{E}(M_1), \mathcal{E}(M_2), \ldots, \mathcal{E}(M_k)\}, \tag{4.9}$$

is recognized as the shape of the object.

## 4.2 Closed-Form Surfaces

We have discussed the recognition-by-registration scheme in general. This section describes how it applies to surfaces with closed-form descriptions. Specifically, we consider a model $M$ bounded by a surface with an implicit form $f(x, y, z) = 0$ and a parameterization $\boldsymbol{\sigma}(u, v)$. The parametric form is needed for domain discretization and principal curvature computation. The implicit form is needed for using equation (4.11) to estimate point distance to the surface.

### 4.2.1 Lookup Table for a Closed-Form Model

Let $\boldsymbol{\sigma}_u$ and $\boldsymbol{\sigma}_v$ denote the first order partial derivatives of $\boldsymbol{\sigma}(u, v)$ with respect to $u$ and $v$, respectively. The surface normal is:

$$\boldsymbol{n} = \frac{\boldsymbol{\sigma}_u \times \boldsymbol{\sigma}_v}{||\boldsymbol{\sigma}_u \times \boldsymbol{\sigma}_v||}. \tag{4.10}$$

We use two matrices [40, p. 132] which consist of the coefficients of the first and second fundamental forms, respectively:

$$\mathcal{F}_I = \begin{pmatrix} \boldsymbol{\sigma}_u \cdot \boldsymbol{\sigma}_u & \boldsymbol{\sigma}_u \cdot \boldsymbol{\sigma}_v \\ \boldsymbol{\sigma}_u \cdot \boldsymbol{\sigma}_v & \boldsymbol{\sigma}_v \cdot \boldsymbol{\sigma}_v \end{pmatrix}$$

$$\mathcal{F}_{II} = \begin{pmatrix} \boldsymbol{\sigma}_{uu} \cdot \boldsymbol{n} & \boldsymbol{\sigma}_{uv} \cdot \boldsymbol{n} \\ \boldsymbol{\sigma}_{uv} \cdot \boldsymbol{n} & \boldsymbol{\sigma}_{vv} \cdot \boldsymbol{n} \end{pmatrix},$$

where $\boldsymbol{\sigma}_{uu}$, $\boldsymbol{\sigma}_{uv}$ and $\boldsymbol{\sigma}_{vv}$ are the second order partial derivatives of $\boldsymbol{\sigma}$. The eigenvalues of $\mathcal{F}_I^{-1}\mathcal{F}_{II}$ are the two principal curvatures.

We uniformly discretize the domain of $\boldsymbol{\sigma}(u, v)$. The principal curvatures are calculated at all discretization points and their values are stored in a lookup table.

### 4.2.2 Registration on a Closed-Form Model

The normal $N$ at the intersection point $p$ of $\alpha$, $\beta$, and $\gamma$ is estimated according to (4.1). The tangent plane at $p$ is orthogonal to $N$. We arbitrarily pick two orthogonal tangent vectors $\boldsymbol{t}_1$ and $\boldsymbol{t}_2$ at $p$. So, $N$, $\boldsymbol{t}_1$, and $\boldsymbol{t}_2$ form a local frame at $p$. All data points $(x_i, y_i, z_i)$, $1 \leq i \leq n$, along $\alpha$, $\beta$, and $\gamma$, are converted into the local coordinates at $p$,

$$\begin{pmatrix} x_i' \\ y_i' \\ z_i' \end{pmatrix} = (\boldsymbol{t}_1\ \boldsymbol{t}_2\ N)^T \left( \begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} - p \right).$$

Suppose the intersection point $p$ of $\alpha$, $\beta$, and $\gamma$ coincides with a point $q$ on the model $M$ in its body frame. The normal $\boldsymbol{n}$ at $q$ is given by (4.10). We align the two normals. Let $\boldsymbol{d}_1$ and $\boldsymbol{d}_2$ be two arbitrarily selected tangent vectors at $q$ such that $\boldsymbol{d}_1 \times \boldsymbol{d}_2 = \boldsymbol{n}$. Let $\phi$ be the angle of rotation from $\boldsymbol{d}_1$ to $\boldsymbol{t}_1$. Every data point $(x_i', y_i', z_i')$ in the local coordinates at $p$ is transformed into some point $(x_i'', y_i'', z_i'')$ in the coordinate system of $M$:

$$\begin{pmatrix} x_i'' \\ y_i'' \\ z_i'' \end{pmatrix} = (\boldsymbol{d}_1\ \boldsymbol{d}_2\ \boldsymbol{n}) \begin{pmatrix} x_i' \cos\phi - y_i' \sin\phi \\ x_i' \sin\phi + y_i' \cos\phi \\ z_i' \end{pmatrix} + q.$$

This transformation depends on the parameters $u$ and $v$, which locate the point $q$ on $M$, as well as on the angle $\phi$.

The distance from the transformed data point $(x_i'', y_i'', z_i'')$, $1 \leq i \leq n$, to $M$ has a first order approximation $\frac{|f(x_i'', y_i'', z_i'')|}{||\nabla f(x_i'', y_i'', z_i'')||}$. The superposition error (4.8) has the form:

$$E(u, v, \phi) = \frac{1}{n} \sum_{i=1}^{n} \frac{|f(x_i'', y_i'', z_i'')|}{||\nabla f(x_i'', y_i'', z_i'')||}. \tag{4.11}$$

#### 4.2.2.1 Error Minimization

Starting at every candidate point $p_j = \boldsymbol{\sigma}(u_j, v_j)$, we search for a local minimum of the function $E(u, v, \phi)$ using the steepest descent method [39, p. 318]. The gradient $\nabla E(u, v, \phi) = (E_u, E_v, E_\phi)$ is obtained by differentiating (4.11) with respect to $u$, $v$, and $\phi$, respectively.

The value $(u_j, v_j)$ obtained from the table lookup is used as initial estimate of $(u, v)$. Let $p_j^{(0)} = p_j$, such that $u_j^{(0)} = u_j$ and $v_j^{(0)} = v_j$. Since $u_j^{(0)}$ and $v_j^{(0)}$ are known, the error $E$ depends on $\phi$ only. Figure 4.3 shows how it changes with $\phi$ when three (synthetic) data curves are superposed onto an ellipsoid. We find $\phi_j^{(0)}$ that minimizes the registration error and use

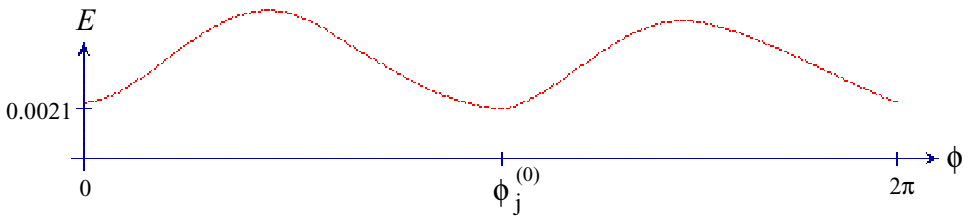



Figure 4.3 Error (4.11) of superposing three data curves onto an ellipsoid. The curves, each consisting of 61 points, were obtained from the ellipsoid where they intersected at $(u, v) = (1.02, 0.69)$. They are now placed at the point $(0.31, 0.63)$ and rotated about its surface normal through an angle $\phi$.

it as the initial value of $\phi_j$.

To compute the initial estimate $\phi_j^{(0)}$, we discretize the domain $[0, 2\pi)$ of $\phi$. All local minima of $E$ are bracketed. Within each bracket, bisection is performed to find a local minimum. Then, $\phi_j^{(0)}$ is the angle yielding the smallest of these minima.

#### 4.2.2.2 The Registration Algorithm

The input to the algorithm includes data points along $\alpha$, $\beta$ and $\gamma$ and their intersection point $p$. It also includes a model $M$ with a lookup table recording precomputed principal curvatures at discretization points. The pseudocode of the algorithm is given below.

1 estimate the normal $N$ of data at $p$

2 estimate two principal curvatures $\kappa_1$ and $\kappa_2$ at $p$

3 convert all data points into the local coordinates at $p$

4 use the lookup table to find candidate points on $M$

5 **for** each candidate point $p_j^{(0)} = \boldsymbol{\sigma}\left(u_j^{(0)}, v_j^{(0)}\right)$

6 $\phi_j^{(0)} \leftarrow$ initial estimate of $\phi$

7 $k \leftarrow 0$

8 **repeat**

9 $p_j^{(k+1)} \leftarrow$ steepest descent from $p_j^{(k)}$on $M$
along $-\nabla E\left(u_j^{(k)}, v_j^{(k)}, \phi_j^{(k)}\right)$

10 $k \leftarrow k+1$

11 **until** no further minimization is possible

12 $p_j^* \leftarrow p_j^{(k-1)}$

Each candidate point $p_j$ will converge to some point $p_j^*$ at which the superposition error (4.11) achieves a local minimum $E_j^*$, as illustrated in Figure 4.4. Let $p^*$ be the point among

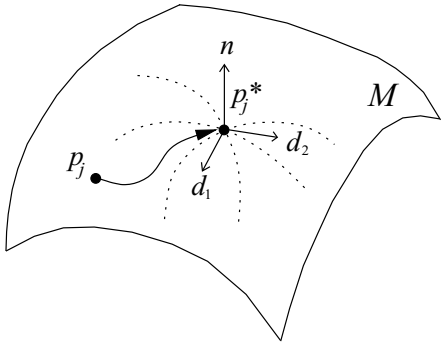



Figure 4.4 Translating and rotating three concurrent data curves (as dotted lines) on the model $M$ to find the best superposition. The point $p_j$ is the initial estimate of the location of the curve intersection while the point $p_j^*$ is the location found through optimization.

the resulting points $p_1^*, p_2^*, \ldots, p_m^*$ that yields the registration error, which is

$$\mathcal{E}(M) = \min\{E_1^*, E_2^*, \ldots, E_m^*\}. \tag{4.12}$$

Then $p^*$ is the estimated location of $p$ on the model $M$.

### 4.2.3 Simulation

Table 4.1 lists four families of surfaces, in both implicit and parametric forms, which were used in our simulation.

Table 4.1 Four surface families used in the simulation.

| **Implicit form** | **Parametric form** |
|---|---|
| $\frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} = 1$ (ellipsoid) | $(a\cos u \sin v, b \sin u \sin v, c \cos v)$ $(u, v) \in [0, 2\pi] \times [0, \pi]$ |
| $\frac{x^2}{a^2} + \frac{y^2}{b^2} = z$ (elliptic paraboloid) | $(av \cos u, bv \sin u, v^2)$ $(u, v) \in [0, 2\pi] \times [-1, 1]$ |
| $x^3 - 3xy^2 = z$ (monkey saddle) | $(u, v, u^3 - 3uv^2)$ $(u, v) \in [-1, 1] \times [-1, 1]$ |
| $x^2y^2 = z$ (crossed trough) | $(u, v, u^2v^2)$ $(u, v) \in [-1, 1] \times [-1, 1]$ |

#### 4.2.3.1 Curve Registration Results

Consider the elliptic paraboloid displayed in Figure 4.5. We select a point $p$, say, with

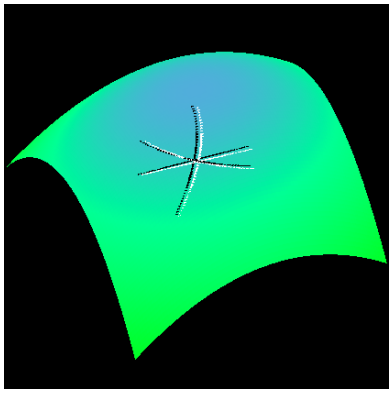


Figure 4.5 Three data curves (in black color) and their registered locations (in white color) on the surface of an elliptic paraboloid given by $z = \frac{x^2}{1.5^2} + \frac{y^2}{1.1^2}$.

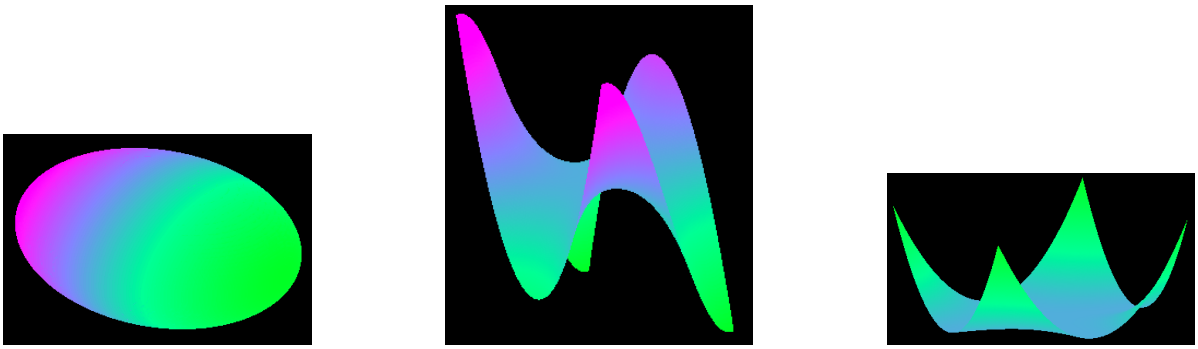
parameter values $(u, v) = (1.21, 0.43)$. Intersect the elliptic paraboloid with three arbitrary

planes through $p$, and generate 61 data points along each of the three intersection curves. Random noises within the range of $\pm 0.001$ are added to the generated data points.

The normal and the two principal curvatures at $p$ are estimated as $\tilde{N} = (0.1611, 0.5764, 0.8011)$, $\tilde{\kappa}_1 = 0.9548$, and $\tilde{\kappa}_2 = 0.6370$.[1] A table lookup finds three candidate points: $p_1 = (1.26, 0.4)$, $p_2 = (1.41, 0.4)$, and $p_3 = (0.63, 0.5)$, all in parameter values. The superposition errors (4.11) at these points are respectively $E_1 = 0.000888$, $E_2 = 0.000961$, and $E_3 = 0.001654$.

Local searches at the candidate points yields locations $p_1^* = (1.26, 0.44)$, $p_2^* = (1.27, 0.44)$, and $p_3^* = (1.17, 0.44)$. The corresponding superposition errors are $E_1^* = 0.000662$, $E_2^* = 0.000666$, and $E_3^* = 0.000686$. The location of $p$ is thus estimated to be $p^* = (1.26, 0.44)$, which is very close to its real location $(1.21, 0.43)$. The three curves are well registered onto the elliptic paraboloid, as shown in Fig. 4.5.

More registration tests are conducted on the same elliptic paraboloid as well as on three other shapes including an ellipsoid, the monkey saddle, and the crossed trough (see Figure 4.6). The results are displayed in Figure 4.7.



(a) (b) (c)

Figure 4.6 Three surfaces used in the simulation in addition to the one shown in Figure 4.5: (a) an ellipsoid with $a = 1$, $b = 0.8$, and $c = 0.5$; (b) the monkey saddle; and (c) the crossed trough.

On each of the four surfaces, ten registration instances are performed with randomly generated intersection points $p$ and three sampling planes through each. The real intersection points $p$ of three data curves are drawn as circular dots and their estimated locations $p^*$ as crosses. In

[1] To verify, we also compute their values using the surface equation: $N = (0.1612, 0.5827, 0.7966)$, $\kappa_1 = 0.9074$, and $\kappa_2 = 0.6519$, respectively.

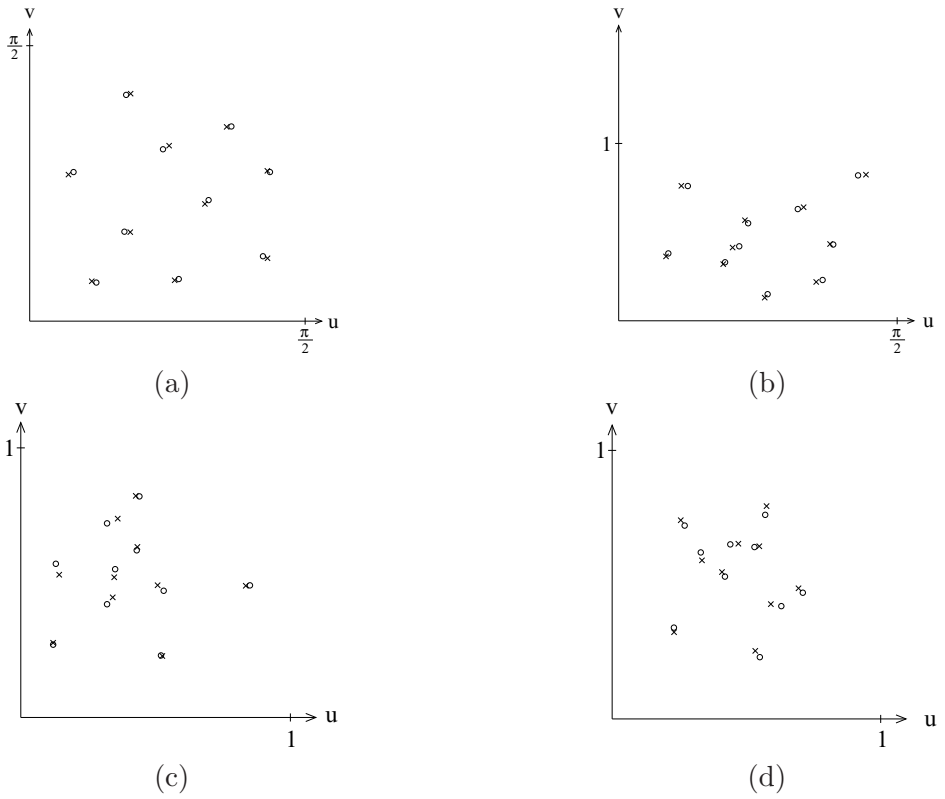



Figure 4.7 Instances of curve registration on the four shapes displayed in Figures 4.5 and 4.6: (a) an ellipsoid; (b) an elliptic paraboloid; (c) the monkey saddle; (d) the crossed trough. In each instance, the intersection point $p$ of three data curves is represented by a circular dot and its estimated location $p^*$ by the closest cross.

the figure every original location $p$ and its estimate $p^*$ lie the closest to each other. Table 4.2 summarizes the Euclidean distances $\|p - p^*\|$ for the registration instances in Figure 4.7.

#### 4.2.3.2 Recognition Tests

As a recognition example we generate three data curves through a randomly selected intersection point on the crossed trough, and register them onto the four surfaces. The registration error (4.12) on the crossed trough itself is 0.000393. Using the estimated principal curvatures values, we find no candidate curve intersection points on the ellipsoid and elliptic paraboloid after table lookups. The error on the monkey saddle is 0.001020, 159% higher than that on the crossed trough.

Table 4.2 Minimum, maximum and average Euclidean distance $\|p - p^*\|$ between the real and estimated locations of the curve intersection calculated over the 40 registration instances in Figure 4.7.

| | ellipsoid | elliptic paraboloid | monkey saddle | crossed trough |
|---|---|---|---|---|
| min | 0.0169 | 0.0148 | 0.0053 | 0.0171 |
| max | 0.0408 | 0.0448 | 0.0433 | 0.0396 |
| avg | 0.0288 | 0.0297 | 0.0234 | 0.0265 |

For each surfaces in Figures 4.5 and 4.6, ten recognition instances, each with a different curve intersection, were carried out. The results are displayed in Table 4.3. In total, 12 out

Table 4.3 Summary of recognition tests, ten on each shape.

| successes | ellipsoid | elliptic paraboloid | monkey saddle | crossed trough |
|---|---|---|---|---|
| table lookup | 2 | 1 | 9 | 0 |
| local search | 8 | 9 | 1 | 10 |

of 40 tests succeeded after table lookups had yielded no candidate points on the wrong models. The other 28 successes involved both table lookups and local optimizations. In these tests, the registration errors on the wrong models exceeded those on the right models by an average of 180%.

## 4.3 Free-Form Surfaces

Unlike closed-form shapes, free-form surfaces usually do not have implicit or parametric forms. More specifically, the surface of a free-form object cannot be easily described by a single function. Since most of the real world objects are free-form, it is important for them to be recognized by a robot.

The general recognition scheme in Section 4.1 will be applicable. However, compared to the recognition of shapes with closed-form description, the difference lie in the lookup table construction and the registration of data curves onto a model. Let us begin with a brief introduction to the models of free-form objects.

### 4.3.1 Triangular Mesh Models

Free-form objects are often modeled as triangular meshes. Let a triangular mesh model $M = (V, F)$ consists of a set $V$ of vertices, and a set $F$ of faces. A vertex $v \in V$ is described by its 3D coordinates. A face $f \in F$ is represented by the indices of the three vertices of $f$. $M$ can be obtained by scanning a free-form object with a laser range sensor.

The connectivity information of neighboring vertices and faces on $M$ is not explicit. Finding a face incident to a vertex or adjacent to another face takes $O(|F|)$ time. Such operations are frequently performed during the lookup table construction and curve registration processes. In order to perform these operations efficiently, for each vertex and face we need to store the information about their neighbors.

#### 4.3.1.1 Triangular Mesh Connectivity

Each vertex $v \in V$ store its 3D coordinates and a pointer to one of its incident faces. Each face $f \in F$ has pointers to its three vertices and to the three adjacent faces, as shown in Figure 4.8. Since all faces in $F$ are triangles, we will refer to them as such from now on.


(a) (b)

Figure 4.8 Connectivity of a triangular mesh: (a) each vertex has a pointer to one of its incident triangles; (b) each triangle has pointers to its three vertices and to the three adjacent triangles.

A *k-ring* neighborhood of a vertex $v$ is a set of all vertices that lie at most $k$ edges away from $v$. Figure 4.8(a) shows an example of the 1-ring neighborhood of $v$. Using the connectivity information illustrated in Figure 4.8, the $k$-ring neighborhood can be efficiently computed for any vertex of $M$. Next, we describe how to construct a lookup table.

### 4.3.2 Lookup Table for a Triangular Mesh Model

In contrast to a closed-form surface, the principal curvatures on a triangular mesh do not have closed forms. They are estimated through fitting.

A typical triangular mesh model of an object consists of a large number of vertices. Estimating principal curvatures at every vertex is inefficient and often unnecessary for the purpose of object recognition. Instead, computation is carried out only at some sampled vertices on the model.

#### 4.3.2.1 Sampling a Triangular Mesh

The triangular mesh is sampled densely in high curvature regions and sparsely in low curvature regions. Since the principal curvatures measure the rate of geometric change of a surface, the sampling density should be adjusted according to the change rate of the surface normal.

The surface normal $\boldsymbol{n}$ at $v$ is estimated as,

$$\boldsymbol{n} = \frac{\sum_1^k \boldsymbol{n}_i}{\| \sum_1^k \boldsymbol{n}_i \|}, \tag{4.13}$$

where $k$ is the total number of triangles in the 1-ring neighborhood of $v$, and $\boldsymbol{n}_i$ is the normal of the $i$th such triangle.

We randomly select an *unmarked* vertex $v \in V$ and compute its 6-ring neighborhood $\mathcal{N}$. Then, the angles between the normal at $v$ and the normals of all triangles in $\mathcal{N}$ are calculated. If any of these angles is greater than some specified value $\theta$, we reduce the ring size of $\mathcal{N}$ by one. The size reduction is repeated until all angles are less than $\theta$, or $\mathcal{N}$ becomes a 2-ring neighborhood of $v$. Then, if all the vertices in $\mathcal{N}$ are *unmarked*, we add $v$ to a list $\mathcal{L}$ and *mark* all vertices. The pseudocode of the sampling procedure is given below.

1 $\mathcal{L} \leftarrow \emptyset$

2 **repeat**

3 randomly select an *unmarked* vertex $v \in V$

4 $\mathcal{N} \leftarrow$ 6-ring neighborhood of $v$

5 $\boldsymbol{n} \leftarrow$ normal at $v$

5 **repeat**

6 $\phi \leftarrow$ maximum angle between $\boldsymbol{n}$ and the normals of all triangles in $\mathcal{N}$

7 **if** $\phi > \theta$

8 ring($\mathcal{N}$) $\leftarrow$ ring($\mathcal{N}$) $- 1$

9 **until** $\phi \leq \theta$ **or** ring($\mathcal{N}$) $= 2$

10 **if** all vertices in $\mathcal{N}$ are *unmarked*

11 add $v$ to the list $\mathcal{L}$

12 *mark* all vertices in $\mathcal{N}$

13 **until** half of all vertices in $V$ are *marked*

The procedure exits when half of all vertices in $V$ are marked, for two reasons. Firstly, we want to make sure that the procedure halts. Secondly, the surface of the model is well covered by $\mathcal{L}$ when half of all vertices in $V$ are marked.

The list $\mathcal{L}$ is the sample set of vertices at which the principal curvatures will be estimated. We have empirically found the optimal value of $\theta$ to be $\frac{\pi}{8}$.

Fig. 4.9 shows a model of a cat object with 64821 vertices and 129638 faces, generated by NextEngine's desktop 3D scanner. The sampling procedure selects 550 vertices (displayed as black dots).

#### 4.3.2.2 Estimation of Principal Curvatures

The product and half of the sum of two principal curvatures are Gaussian and mean curvatures, respectively. A number of proposed approaches to estimate Gaussian and mean curvatures on triangular meshes were compared in [48] where paraboloid fitting was found to be the best estimation method. This method fits paraboloids in the form of $z = ax^2 + bxy + cy^2$ in the local neighborhoods of vertices on synthetic triangular mesh models that represent the

(a) (b)

Figure 4.9 Sampling a cat model: front (a) and back (b) views of the model with a total of 550 sample vertices displayed as black dots on its surface.

tesselations of NURBS surfaces. Since the models used in our experiments were generated from range data, we obtained more robust curvature estimation by fitting a paraboloid with linear and constant terms:

$$z = ax^2 + bxy + cy^2 + dx + ey + f. \tag{4.14}$$

The normal $\boldsymbol{n}$ at a vertex $v$ is estimated according to (4.13). All vertices in the 6-ring neighborhood $\mathcal{N}$ of $v$ are transformed into the local frame formed by $\boldsymbol{n}$ and two orthogonal tangents. The vertices in $\mathcal{N}$ are sorted by their Euclidean distances to $v$. We fit the paraboloid in (4.14) over five local neighborhoods with 30, 40, 50, 60, and 70 geometrically closest vertices to $v$, respectively. Each resulting paraboloid yields a pair of principal curvatures via closed-form evaluation. The median values are selected as the estimates of principal curvatures at $v$. The selection of two principal curvatures are independent of each other, meaning that they might have been obtained from different paraboloids.

Principal curvatures are estimated for all vertices in $\mathcal{L}$ and stored in a lookup table used for recognition.

### 4.3.3 Registration on a Triangular Mesh Model

Now, we describe how the three data curves $\alpha$, $\beta$, and $\gamma$ are superposed onto a surface of a triangular mesh model.

#### 4.3.3.1 Superposing Data Curves onto a Model

Let $v$ be a vertex on a triangular mesh model $M$. The normal $\boldsymbol{n}$ at $v$ is estimated according to (4.13) and the normal $N$ at the intersection point $p$ of $\alpha$, $\beta$ and $\gamma$ is obtained from (4.1). To superpose $\alpha$, $\beta$, and $\gamma$ onto $M$ at $v$, we coincide $p$ with $v$ and align $N$ with $\boldsymbol{n}$. Consequently, all data points along $\alpha$, $\beta$, and $\gamma$ are converted into the coordinate system of $M$. The alignment eliminates all degrees of freedom except the rotation $\phi$ of $\alpha$, $\beta$, and $\gamma$ about $\boldsymbol{n}$.

For a value of $\phi$, we find the corresponding points $(x_i', y_i', z_i')^T$ on $M$ to the data points $(x_i, y_i, z_i)^T$ along $\alpha$, $\beta$, and $\gamma$. To do so, we first fit three planes to $\alpha$, $\beta$, and $\gamma$, respectively. Then, we sample points along the intersections of the three planes and $M$ with the same separations between points as in $\alpha$, $\beta$, and $\gamma$, respectively. Note that the points $(x_i', y_i', z_i')^T$ on $M$ are not necessarily vertices. They can be points on the boundaries or in the interiors of triangles.

From (4.8), the superposition error is

$$E(v) = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(x_i - x_i')^2 + (y_i - y_i')^2 + (z_i - z_i')^2}. \tag{4.15}$$

To minimize $E$ at $v$ over the orientation $\phi$, we can no longer apply the bisection method. Instead, we discretize the domain $[0, 2\pi)$ of $\phi$ and calculate $E$ at every value of $\phi \in \{0, \frac{\pi}{18}, \frac{\pi}{9}, \ldots, \frac{35\pi}{18}\}$. The minimum value is chosen.

#### 4.3.3.2 Local Search

The superposition error $E$ is further minimized in the neighborhood of a candidate vertex $v_j$ using a *greedy* approach. We calculate $E$ at each *unvisited* vertex of a 1-ring neighborhood $\mathcal{N}$ of $v_j$. Then, we mark all unvisited vertices in $\mathcal{N}$ as *visited* and move from $v_j$ to a vertex in $\mathcal{N}$ with the smallest value of $E$. This movement is repeated until $E$ achieve a local minimum.

Let $v_1, v_2, \ldots, v_m$ be the candidate vertices found by the table lookup discussed in Section 4.1.2.1. Perform a local search starting at every candidate vertex $v_i$ and let $v_i'$ be the local minimum vertex. The registration error on the model $M$ is

$$\mathcal{E}(M) = \min\{E(v_1'), E(v_2'), \ldots, E(v_m')\}. \tag{4.16}$$

The vertex $v'_k$ which yields the registration error $\mathcal{E}$ is the estimated location of the intersection point $p$ of $\alpha$, $\beta$, and $\gamma$ on $M$.

### 4.3.4 Simulations

In simulations, we use synthetic data to test the registration algorithm on triangular meshes. Experiments with real tactile data will be presented in Section 4.4. We scanned 10 objects using NextEngine's 3D desktop scanner. The images of their models are displayed in Figure 4.10. The description of each model is given in Table 4.4.

On each model, we randomly select 10 vertices such that no selected vertex is in the neighborhood of any other. This is to make sure that these vertices are spread all over the surface of a model. There are 10 models in the database, so we have a total of 100 selected vertices which are used as curve intersections for data generation.

To generate three concurrent data curves intersecting at the selected vertex $v$ on a model $M$, we intersect three planes passing through $v$ with $M$. On each intersection curve, data points are sampled with interval $d$ such that $v$ is the middle point. The data points are not necessarily the vertices. They can be points in the interiors or on the boundaries of triangles.

With data generated at 100 selected vertices, we can perform 100 instances of recognition. We look at how the time and success rate of recognition depend on the following parameters:
1) tolerance $\delta$ defined in (4.7),
2) length $l$ of a data curve,
3) angle $\phi$ between sampling planes.
The length of data curve of $n$ points is $l = (n-1)d$, where $d$ is the distance between two neighbor points. All three sampling planes are perpendicular to a tangent plane at $v$. The angle between sampling planes is $\phi$.

Figure 4.11 plots recognition success rate and recognition time versus tolerance. We use $n = 81$, $d = 0.3$mm, $l = 24$mm, and $\phi = 60°$. The larger the tolerance the more candidate vertices are selected in a table lookup. With more candidate vertices, we are more likely to recognize an object. The recognition time increases with the number of candidate vertices.

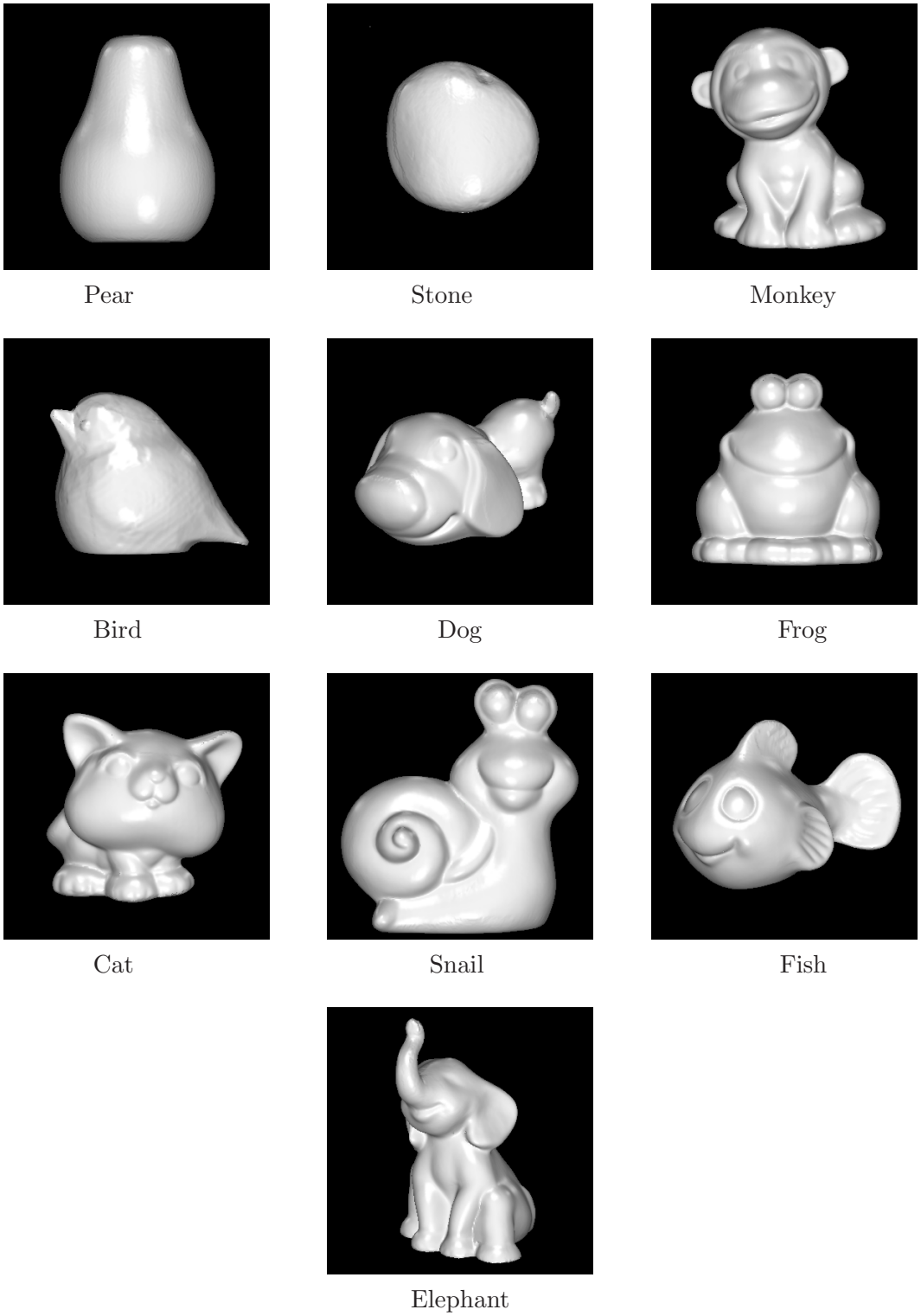



Figure 4.10 Images of 10 models in the database.

Table 4.4 Description of each model in the database.

| Model | Number of vertices | Number of triangles | Dimension (mm) | Surface area ($mm^2$) | Sample vertices | Sampling time (sec) | Lookup table construction time (sec) |
|---|---|---|---|---|---|---|---|
| Pear | 84656 | 169308 | $83.9 \times 62.6 \times 62.4$ | 15169 | 400 | 4.857 | 5.017 |
| Stone | 42777 | 85550 | $62.7 \times 56.3 \times 43.5$ | 9131 | 250 | 4.446 | 3.195 |
| Monkey | 75443 | 150882 | $80.0 \times 71.9 \times 51.5$ | 14488 | 500 | 2.654 | 6.389 |
| Bird | 63323 | 126642 | $78.6 \times 61.4 \times 55.9$ | 12258 | 400 | 9.234 | 5.077 |
| Dog | 67101 | 134198 | $90.3 \times 62.5 \times 43.2$ | 13654 | 450 | 1.593 | 5.748 |
| Frog | 91970 | 183936 | $79.4 \times 78.9 \times 58.5$ | 17023 | 600 | 4.657 | 7.511 |
| Cat | 64821 | 129638 | $71.9 \times 67.2 \times 53.5$ | 12681 | 550 | 3.415 | 7.030 |
| Snail | 101268 | 202532 | $94.8 \times 89.6 \times 52.2$ | 18257 | 650 | 5.709 | 8.271 |
| Fish | 71490 | 142976 | $90.5 \times 71.5 \times 61.4$ | 14103 | 570 | 4.216 | 7.341 |
| Elephant | 72665 | 145326 | $92.0 \times 62.1 \times 61.5$ | 13308 | 650 | 5.398 | 8.462 |

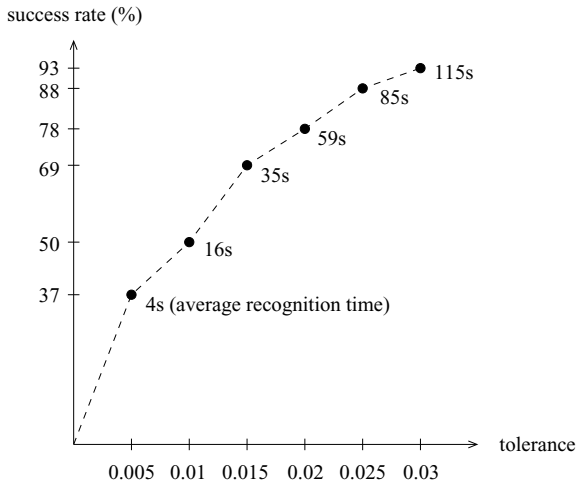



Figure 4.11 Recognition success rate and time versus tolerance. For all instances of recognition, the length of data curves was $l = 24$mm and the angle between sampling planes was $\phi = 60°$.

Figure 4.12 plots success rate and recognition time versus the length of a data curve. The tolerance and the angle between sampling planes are $\delta = 0.03$ and $\phi = 60°$, respectively. Long data curves provide more information about the shape of the object than short ones. In other words, it is easier to embed the shorter data curves onto the object than the longer ones. Long data curves can only be embedded onto the object they were originally obtained from. So, the recognition success rate is higher for longer data curves. Since long data curves contain more data points that short ones, the recognition time increases with the length of the data curves.

Figure 4.13 shows how the angle between sampling planes affects the recognition success rate and time. For all values of $\phi$, the tolerance is $\delta = 0.03$ and the length of the data curves is $l = 24$mm.

From Figure 4.13, we see that the success rate increases when the angle between sampling planes increases. It is intuitive because data curves span more area on the object's surface when they more spread away from each other. Consequently, they possess more information about the shape of the object and thus result in higher recognition rate. The angle between sampling planes does not have much impact on the recognition time. Note that the angle between first and second sampling planes does not have to be the same as the angle between

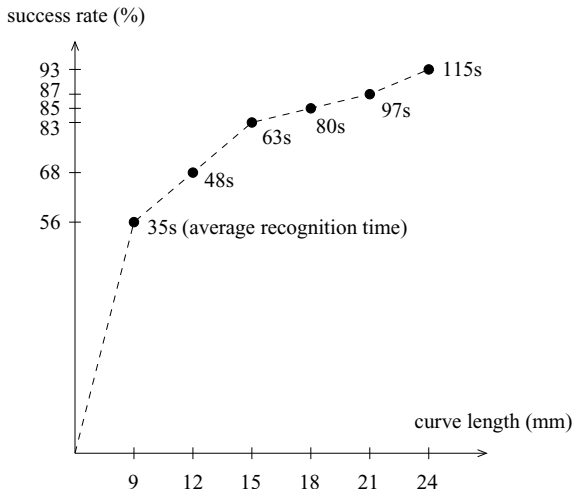



Figure 4.12 Recognition success rate and time versus the length of data curves. For all instances of recognition, we used tolerance $\delta = 0.03$ and angle between sampling planes $\phi = 60°$.

second and third sampling planes. We used the same angles for the simplicity of presentation.

## 4.4 Experiments

In this section, we present experiments with objects bounded by algebraic and free-form surfaces.

### 4.4.1 Data Acquisition

Data points were obtained using a joystick sensor driven by an Adept robot as displayed in Figure 4.14. The detailed description of the sensor is given in [27]. By constraining the robot movement in three different planes the sensor sampled points along the corresponding intersection curves with an object. The angle between two neighboring planes was set to be $\frac{\pi}{3}$. On each curve, $n$ points including the curve intersection were sampled. The interval between two neighboring points was set to $d$. The total number of points along the three data curves was $3n - 2$.

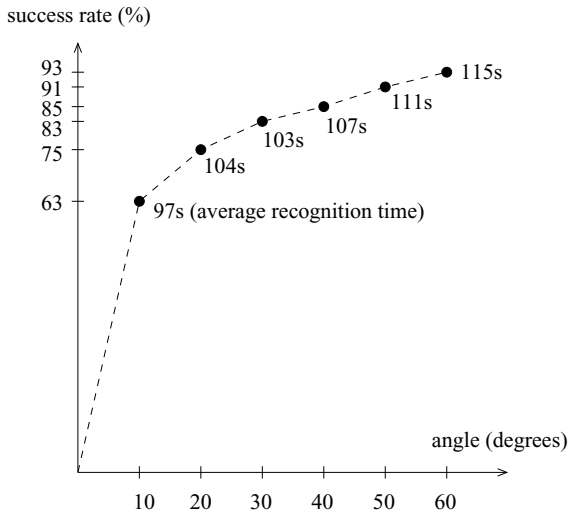



Figure 4.13 Recognition success rate and time versus the angle between sampling planes. For all instances of recognition, we used tolerance $\delta = 0.03$ and length of data curves $l = 24$mm.

### 4.4.2 Results on Closed-Form Objects

Figure 4.15 displays four algebraic objects used in our experiments. For data curve sampling we used $n = 41$ and $d = 0.05$mm. Due to symmetry, curve registration results on the sphere and cylinders were meaningless. But the minimum registration error $\mathcal{E}_{\min}$ defined by (4.9) was still useful for recognition of these shapes. This error was computed for every object.

In Table 4.5, each column records the minimum registration error (4.9) for registering three data curves acquired from the same object onto all four models. The diagonal cells

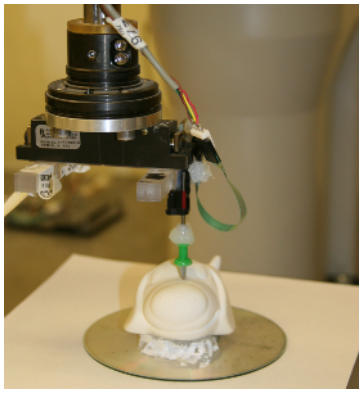


Figure 4.14 A robot is sampling data points on a dog object.

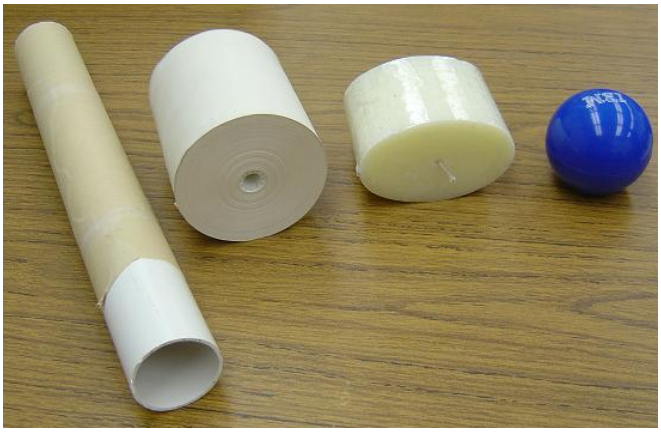


Figure 4.15 Algebraic objects used in experiments: two regular cylinders with diameters 50.4mm and 94mm, respectively, an elliptic cylinder with semimajor axis 50.8mm and semiminor axis 31.75mm, and a sphere with radius 33mm.

Table 4.5 Minimum error (in millimeters) of registering data acquired from the four objects shown in Figure 4.15 onto their models.

| model \ object | cylin. 1 | cylin. 2 | ell. cylin. | sphere |
|---|---|---|---|---|
| cylinder 1 | **0.033** | 0.155 | 0.446 | 0.296 |
| cylinder 2 | 0.182 | **0.072** | 0.195 | 1.056 |
| elliptic cylinder | 0.230 | 0.224 | **0.156** | 0.254 |
| sphere | 0.271 | 0.306 | 0.929 | **0.055** |

in the table correspond to registration onto the correct models, while the non-diagonal cells correspond to registration on incorrect models. It is seen that every diagonal cell has the smallest value in its column. As a result, all four objects were correctly recognized.

### 4.4.3 Results on Free-form Objects

Figure 4.16 shows the results from an experiment on ten free-form objects. In the figure, each model is displayed next to the corresponding object. On each object, we selected a point and sampled three data curves passing through it. This point is displayed as a white dot on the surface of each object. The number of points per data curve was 21 and the interval between points was 1.2mm. The tolerance $\delta$ was set to be 0.01.

The curve registration results are shown in Figure 4.16. The registered data curves are

displayed in red color on the surface of each model. On every object the data curves were registered at locations that were almost indistinguishable from the original ones.

Table 4.6 shows the results on recognizing ten objects based on curve registrations in Figure 4.16. Each cell $a_{ij}$ in the table records the registration error (4.16) of the data curves obtained on the object in $j$th column onto the model in $i$th row. The NC entries in the table mean that *no candidate* vertices were found on those models. In such cases, they were immediately rejected after table lookups. Within each column the minimum registration error (4.9) is found on a diagonal cell $a_{ii}$. As a result, all ten objects were correctly identified.

Table 4.7 displays the time spent on recognition in Table 4.6. Each cell $b_{ij}$ in the table records the time taken to register three data curves acquired on the object in $j$th column onto the model in $i$th row. The recognition time is the total time of registration onto all ten models.

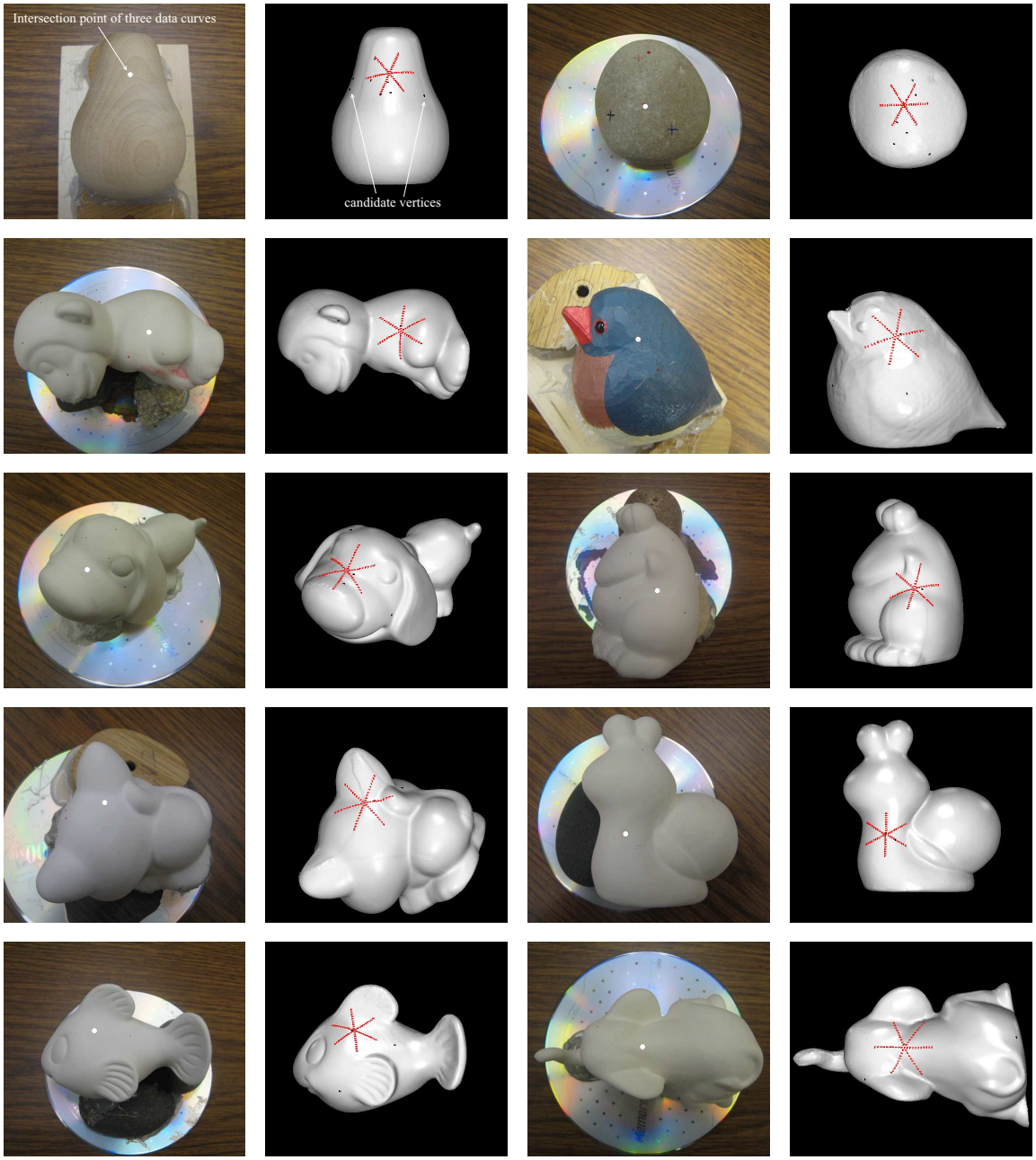



Figure 4.16 Curve registrations on ten objects. Each model is displayed next to the corresponding object. The white dots on the surface of each object indicate the locations through which the robot sampled three concurrent curves. The black dots on the surface of each model are the candidate vertices found after table lookups. All data curves displayed in red color were successfully registered on all models.

Table 4.6 Results on recognizing ten objects. Every cell $a_{ij}$ displays the registration error (in millimeters) per data point of three data curves obtained on the object in $j$th column onto the model in $i$th row. NC entries mean that no candidate vertices were found on those models.

| | | **Objects** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Pear | Stone | Monkey | Bird | Dog | Frog | Cat | Snail | Fish | Elephant |
| **M** | Pear | **0.0724** | 0.1314 | 0.8347 | 0.1411 | NC | NC | 0.3803 | 0.4146 | 0.1607 | 0.6271 |
| **o** | Stone | 0.2523 | **0.0741** | 0.5507 | 0.4308 | NC | NC | NC | 0.5157 | 0.2306 | NC |
| **d** | Monkey | 0.4520 | 0.1027 | **0.1304** | 0.2228 | 0.3403 | 0.6033 | 0.3908 | 0.2057 | 0.2816 | 0.7358 |
| **e** | Bird | 0.2856 | 0.1430 | 0.1661 | **0.0933** | 0.4341 | 0.6639 | 0.5049 | 0.4089 | 0.2617 | 0.9158 |
| **l** | Dog | 0.2940 | 0.1340 | 0.3131 | 0.2250 | **0.1420** | 0.3675 | 0.2131 | 0.3328 | 0.2283 | 0.8359 |
| **s** | Frog | 0.2026 | 0.1273 | 0.4004 | 0.1575 | 0.3011 | **0.2494** | 0.4869 | 1.2953 | 0.1766 | 0.8584 |
| | Cat | 0.3429 | 0.1863 | 0.5486 | 0.3981 | 0.3741 | 0.5384 | **0.1569** | 0.4772 | 0.2640 | 1.0529 |
| | Snail | 0.1907 | 0.1557 | 0.4128 | 0.2308 | 0.2552 | 0.6508 | 0.4575 | **0.1312** | 0.2162 | NC |
| | Fish | 0.2355 | 0.1083 | 0.3370 | 0.2562 | 0.3386 | 0.6460 | 0.4270 | 0.2692 | **0.0762** | 0.3796 |
| | Elephant | 0.3875 | 0.2998 | 0.4680 | 0.2754 | 0.7291 | 0.7552 | 0.4427 | 0.3568 | 0.2475 | **0.1844** |

Table 4.7 Registration times (seconds) corresponding to recognitions of ten objects in Table 4.6. Each cell $b_{ij}$ shows the time taken to register three data curves acquired on the object in $j$th column onto the model in $i$th row. Each cell in the last row displays the total time needed to recognize the object in the corresponding column.

| | | **Objects** | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Pear | Stone | Monkey | Bird | Dog | Frog | Cat | Snail | Fish | Elephant |
| **M** | Pear | 2.143 | 2.524 | 0.290 | 2.333 | 0.160 | 0.180 | 0.480 | 0.350 | 2.603 | 0.211 |
| | Stone | 0.801 | 2.063 | 0.210 | 0.350 | 0.191 | 0.160 | 0.130 | 0.861 | 2.113 | 0.130 |
| **o** | Monkey | 0.591 | 1.141 | 0.330 | 1.412 | 0.711 | 0.491 | 0.862 | 1.472 | 0.821 | 0.601 |
| **d** | Bird | 1.342 | 2.283 | 0.301 | 0.921 | 0.711 | 0.240 | 0.581 | 1.222 | 2.083 | 0.241 |
| **e** | Dog | 0.320 | 0.681 | 0.351 | 1.432 | 0.601 | 0.330 | 0.801 | 0.310 | 1.882 | 0.521 |
| **l** | Frog | 1.262 | 1.392 | 0.391 | 1.112 | 0.941 | 0.481 | 0.691 | 0.370 | 4.467 | 0.842 |
| **s** | Cat | 0.852 | 1.082 | 0.390 | 0.411 | 0.691 | 0.520 | 0.550 | 0.601 | 0.291 | 0.651 |
| | Snail | 2.214 | 2.403 | 0.511 | 1.372 | 0.541 | 0.311 | 0.631 | 0.961 | 2.003 | 0.130 |
| | Fish | 0.872 | 1.872 | 1.092 | 0.761 | 0.591 | 0.301 | 0.541 | 1.903 | 1.412 | 1.202 |
| | Elephant | 0.721 | 2.473 | 0.701 | 1.382 | 0.321 | 0.421 | 0.851 | 0.771 | 1.633 | 1.562 |
| | Recognition time | **11.118** | **17.914** | **4.567** | **11.486** | **5.459** | **3.435** | **6.118** | **8.821** | **19.308** | **6.091** |

# CHAPTER 5. SUMMARY AND DISCUSSION

We have introduced two new methods that recognize objects from tactile data. The first one is invariant-based, which is applicable to 2D objects. Invariants for quadratic curves and two special cubic spline curves have been derived. The derivation of the invariants consists of the following steps. First, the curve equation is reduced to its simplest canonical form through rotation, translation and reparametrization which do not alter the shape of the curve. Then, the parameter variable is eliminated from the equations of the curvature and its derivative with respect to arc length resulting in a single equation that describes the signature curve. Finally, the equation of the signature curve is modified to have the curvature terms on the left-hand side and shape parameter terms on the right-hand side.

The invariants of quadratic curves are computable from local geometry, namely the curvatures and their derivatives, at a couple of points on the curve. Since the curvature and its derivative with respect to arc length are independent of rotation, translation, and reparametrization, so are the invariants computed from them. Unlike the quadratic curves, the invariants for cubic spline curves involve not only the curvature and its derivative, but also a slope. Although the slope depends on rotation, we have shown that finding the slope is reducible to solving a quartic polynomial, which always has a closed-form solution. In addition to recognizing an object, we can also localize it relative to the robot's hand. We have demonstrated the correctness of invariants with simulations. Experimental results show that this method is applicable in real life.

One advantage of the invariant-based approach is that, it is independent of object's position and orientation, which have three degrees of freedom in 2D. Another advantage is that, it requires small amount of tactile data. The disadvantage is that, for some curves it is very

hard to derive such invariants. As for cubic splines, we can only recognize them by solving non-linear optimization problem. Also, the invariant-based recognition is not extendable to 3D objects. Therefore, we use different method to recognize 3D objects which is summarized below.

The second introduced method is registration-based, which is applicable to 3D objects. It recognizes an object with a curved surface from a set of surface models based on "one-dimensional" tactile data. More specifically, the data points are sampled along three *concurrent* curves using a touch sensor on the object's surface. The problem of recognition turns into registering these data curves on each model and choosing the one that yields the best matching result.

The special configuration of the tactile data, namely three concurrent curve segments, makes it possible to estimate the principal curvatures at the curve intersection point. Also, principal curvatures are estimated at discrete points on each model and saved in a table. We look up the table to find the locations on the models that have similar local geometries with the curve intersection point. The data curves are registered in the neighborhoods of those locations and the model yielding the best match is then recognized.

We have applied this recognition method to closed-form and free-form surfaces. For closed-form surfaces, a lookup table is constructed by uniformly discretizing the surface domain and calculating the principal curvatures at discrete points using closed-from expressions. The curve registration is converted to a nonlinear optimization problem, which is solved using steepest descent algorithm. For free-form surfaces, the lookup table is constructed by randomly sampling a triangular mesh and estimating the principal curvatures at the sampled vertices using paraboloid fitting method. The curves are registered on the triangular mesh using a greedy algorithm, which moves from one vertex to another by minimizing the total distance of all data points to the surface. Experimental results successfully recognizing four closed-form and ten free-form objects have been presented.

The dependency of the registration-based recognition method on curvatures may be removed. We can bypass the curvature estimation and table lookup, and register the data

curves starting at every selected discrete point on the model. However, this will increase the recognition time but will produce more successful results.

The length and separation of the three data curves affect the recognition success rate. Using longer data curves that are widely spread from each other provides the most information about the local shape of the object, and thus results in more robust recognition.

Although the recognition method is presented using tactile data, it is not limited to a certain sensing modality. The method can be used with any type of sensor as long as it is able to generate three concurrent data curves. Especially, in the case of range sensors, our recognition method would be robust to occlusions because it is based on local data.

# BIBLIOGRAPHY

[1] P. K. Allen and K. S. Roberts. Haptic object recognition using a multi-fingered dexterous hand. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 342–347, 1989.

[2] G. Beccari, S. Caselli, and F. Zanichelli. Pose-independent recognition of convex objects from sparse tactile data. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 3397–3402, 1997.

[3] P. J. Besl and N. D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992.

[4] M. Boutin. Numerically invariant signature curves. *International Journal of Computer Vision*, 40(3):235–248, 2000.

[5] E. Calabi, P. J. Olver, C. Shakiban, A. Tannenbaum, and S. Haker. Differential and numerically invariant signature curves applied to object recognition. *International Journal of Computer Vision*, 26(2):107–135, 1998.

[6] R. J. Campbell and P. J. Flynn. A survey of free-form object representation and recognition techniques. *Computer Vision and Image Understanding*, 81:166–210, 2001.

[7] X. Chen and F. Schmitt. Intrinsic surface properties from surface triangulation. *Proceedings of the Second European Conference on Computer Vision*, pp. 739–743, 1992.

[8] C. S. Chua and R. Jarvis. 3D free-form surface registration and object recognition. *International Journal of Computer Vision*, 17:77–99, 1996.

[9] C. S. Chua and R. Jarvis. Point Signatures: a new representation for 3D object recognition. *International Journal of Computer Vision*, 25(1):63–85, 1997.

[10] H. Civi, C. Christopher, and A. Ercil. The classical theory of invariants and object recognition using algebraic curve and surfaces. *Journal of Mathematical Imaging and Vision*, 19:237–253, 2003.

[11] C. Dorai and A. K. Jain. COSMOS — A representation scheme for 3D free-form objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(10):1115–1130, 1995.

[12] I. Douros and B. F. Buxton. Three-dimensional surface curvature estimation using quadric surface patches. *Proceedings of Scanning*, 2002.

[13] O. D. Faugeras and M. Hebert. The representation, recognition, and locating of 3-D objects. *International Journal of Robotics Research*, 5(3):27–52, 1986.

[14] P. J. Flynn and A. K. Jain. On reliable curvature estimation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 110–116, 1989.

[15] D. Forsyth, J. L. Mundy, A. Zisserman, C. Coelho, A. Heller, and C. Rothwell. Invariant descriptors for 3D object recognition and pose. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):971–991, 1991.

[16] A. Frome, D. Huber, R. Kolluri, T. Bulow, and J. Malik. Recognizing objects in range data using regional point descriptors. *Proceedings of the European Conference on Computer Vision*, 2004.

[17] R. Gal and D. Cohen-Or. Salient geometric features for partial shape matching and similarity. *ACM Transactions on Graphics*, 25(1):130–150, 2006.

[18] P. C. Gaston and T. Lozano-Pérez. Tactile recognition and localization using object models: The case of polyhedra on a plane. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(3):257–266, 1984.

[19] W. E. L. Grimson and T. Lozano-Pérez. Model-based recognition and localization from sparse range or tactile data. *International Journal of Robotics Research*, 3(3):3–35, 1984.

[20] E. Hameiri and I. Shimshoni. Estimating the principal curvatures and the Darboux frame from real 3-D range data. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 33(4):626–637, 2003.

[21] M. Hebert, K. Ikeuchi, and H. Delingette. A spherical representation for recognition of free-form surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(7):681–690, 1995.

[22] D. Hilbert. *Theory of Algebraic Invariants*. Cambridge University Press, 1993.

[23] B. K. P. Horn. Closed-form solution of absolute orientation using unit quaternions. *Journal of the Optical Society of America A*, 4(4):629–642, 1987.

[24] R. Ibrayev and Y.-B. Jia. Tactile recognition of algebraic shapes using differential invariants. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1548-1553, 2004.

[25] R. Ibrayev and Y.-B. Jia. Semi-differential invariants for tactile recognition of algebraic curves. *International Journal of Robotics Research*, 24(11):951–969, 2005.

[26] R. Ibrayev and Y.-B. Jia. Surface recognition by registering data curves from touch. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 55–60, 2006.

[27] Y.-B. Jia, L. Mi, and J. Tian. Surface patch reconstruction via curve sampling. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1371–1377, 2006.

[28] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3D scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 1999.

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

[30] D. Keren, E. Rivlin, I. Shimshoni, and I. Weiss. Recognizing 3D objects using tactile sensing and curve invariants. *Journal of Mathematical Imaging and Vision*, 12(1):5–23, 2000.

[31] Y. Lamdan and H. J. Wolfson. Geometric hashing: a general and efficient model-based recognition scheme. *Proceedings of the Second International Conference on Computer Vision*, pp. 238–249, 1988.

[32] D. S. Meek and D. J. Walton. On surface normal and Gaussian curvature approximations given data sampled from a smooth surface. *Computer Aided Geometric Design*, 17(6):521–543, 2000.

[33] L. Mi and Y.-B. Jia. High precision contour tracking with a joystick sensor. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 804–809, 2004.

[34] A. S. Mian, M. Bennamoun and R. Owens. Automated 3D model-based free-form object recognition. *Sensor Review*, 24(2):206–215, 2004.

[35] A. S. Mian, M. Bennamoun and R. A. Owens. A novel algorithm for automatic 3D model-based free-form object recognition. *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 6348–6353, 2004.

[36] T. Moons, E. J. Pauwels, L. J. van Gool, and A. Oosterlinck. Foundations of semi-differential invariants. *International Journal of Computer Vision*, 14(1):25–47, 1995.

[37] T. Pajdla and L. Van Gool. Matching of 3-d curves using semi-differential invariants. *Proceedings of the Fifth International Conference on Computer Vision*, pp. 390–395, 1995.

[38] E. J. Pauwels, T. Moons, L. J. van Gool, P. Kempenaers, and A. Oosterlinck. Recognition of planar shapes under affine distortion. *International Journal of Computer Vision*, 14(1):49–65, 1995.

[39] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T Vetterling. *Numerical Recipes in C.* Cambridge University Press, 1988.

[40] A. Pressley. *Elementary Differential Geometry.* Springer-Verlag, 2001.

[41] E. Rivlin and I. Weiss. Local invariants for recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3):226–238, 1995.

[42] S. Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes *Symposium on 3D Data Processing, Visualization, and Transmission*, pp. 486–493, 2004.

[43] J. W. Rutter. *Geometry of Curves.* Chapman & Hall/Crc, Boca Raton, Florida, 2000.

[44] Q. Shi, N. Xi, Y. Chen and W. Sheng. Registration of point clouds for 3D shape inspection. *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 235–240, 2006.

[45] F. Stein and G. Medioni. Structural indexing: efficient 3-D object recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):125–145, 1992.

[46] E. M. Stokely and S. Y. Wu. Surface parameterization and curvature measurement of arbitrary 3-D objects: five practical methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(8):833–840, 1992.

[47] Y. Sun, J. Paik, A. Koschan, D. L. Page, and M. A. Abidi. Point fingerprint: a new 3D object representation scheme. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 33(4):712–717,2003.

[48] T. Surazhsky, E. Magid, O. Soldea, G. Elber, and E. Rivlin. A comparison of gaussian and mean curvatures estimation methods on triangular meshes. *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1021–1026, 2003.

[49] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. *Proceedings of the IEEE International Conference on Computer Vision*, pp. 902–907, 1995.

[50] S. M. Yamany and A. A. Farag. Surface signatures: an orientation independent free-form surface representation scheme for the purpose of objects registration and matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(8):1105–1120, 2002.