Semi-Differential Invariants for Tactile Recognition of Algebraic Curves

Rinat Ibrayev and Yan-Bin Jia

Department of Computer Science
Iowa State University
Ames, IA 50011-1040, U. S. A.
rinat,jia@cs.iastate.edu

Abstract

This paper studies the recognition of low-degree polynomial curves based on minimal tactile data. Euclidean differential and semi-differential invariants have been derived for quadratic curves and special cubic curves that are found in applications. These invariants, independent of translation and rotation, are evaluated over the differential geometry at up to three points on a curve. Their values are independent of the evaluation points. Recognition of the curve reduces to invariant verification with its canonical parametric form determined along the way. In addition, the contact locations are found on the curve, thereby localizing it relative to the touch sensor. Simulation results support the method despite numerical errors. Preliminary experiments have also been carried out with the introduction of a method for reliable curvature estimation. 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.

KEY WORDS—touch sensing, differential invariant, semi-differential invariant, curvature, signature curve, shape parameter, resultant

1 Introduction

Human can feel a shape through touch. Essentially, the action is performed to detect some geometric features on the shape which are then subconsciously synthesized in the brain. Typical geometric features include, for instance, smoothness, saliences, concavities, etc.

Supported by touch sensing, the robot can also obtain shape information without the help of a vision system. Such shape inference is important, for example, when camera occlusion becomes inevitable or when motion is involved. Since tactile data are local (and one-dimensional for point contact), seemingly they convey a very limited amount of geometric information. But how much shape knowledge can the robot really acquire then?

Figure 1 illustrates a hand with two tactile fingers touching an object. Suppose through local movements the fingers are able to estimate information such as the curvatures at several points of contact. And suppose the shape is known to be from a finite number of families of parametric curves. Can we *recognize the shape as well as determine the finger placement*?

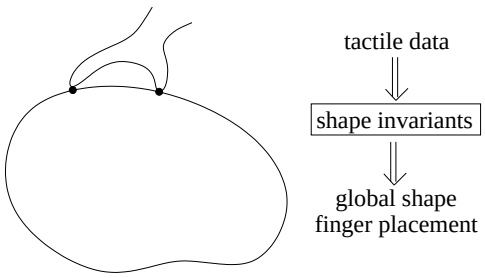


Figure 1: A robotic hand touching an object to recognize its shape.

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 parametrized in the same form. Second, we would like to keep the sensor data to the minimum. This is because a touch sensor, unlike a vision system, does not yield global shape data. Third, we hope to determine where the tactile data were obtained on the shape.

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 this paper, 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 need to be computable from measurements at just a few points. Our investigation will be focused on quadratic and cubic spline curves.

Section 2 goes over some geometric background and overviews our approach to invariant design. Sections 3 and 4 derive Euclidean semi-differential invariants for quadratic curves and three classes of cubic curves. Section 5 presents some simulation results to verify the derived invariants and demonstrates how they are used in recognition. Some experiments with real tactile data are then presented in Section 6. Finally, Section 7 summarizes all results and points to some future work.

1.1 Related Work

There are two primary recognition strategies in model-based vision. The first one hinges on the recovery of viewing parameters (thus the pose). Kriegman and Ponce (1990) constructed implicit shape equations from image contours and then solved for viewing parameters through data fitting. The second approach is to develop descriptors that are invariant to Euclidean, affine, or projective transformation, or to camera-dependent parameters (Mundy and Zisserman 1992, Weiss 1993).

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 (1993), 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. Forsyth et al. (1991) and Keren (1994) presented efficient methods for finding algebraic invariants and demonstrated on recognition of real objects. Civi, Christopher, and Ercil (2003) 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 serious occlusion.

Differential invariants depend on local data and deal with situations like occlusion well. Up till

now, vision- and invariant-based recognition has mainly focused on differential invariants that are independent of various transformation groups but not of point locations on a shape. Calabi, Olver, and Shakiban (1998) introduced the "signature curve" which is invariant to Euclidean or affine transformation. Rivlin and Weiss (1995) 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.

Semi-differential invariants combine global constraints and local information to ease the correspondence issue faced by non-invariant-based methods and also to relieve the burden on estimating higher order derivatives for the evaluation of differential invariants. The theoretical foundation for this type of invariants was presented by Moons et al. (1995). Pajdla and Van Gool (1995) used simple semi-differential invariants to match curves extracted from range data in the presence of partial occlusion.

In touch sensing, shape recognition has long built on the notion of "interpretation tree", which represents all possible correspondences between geometric features of an object and tactile data. Grimson and Lozano-Pérez (1984) identified and localized a 3-D polyhedron from a set of models using tactile measurements of positions and surface normals. Fearing (1990) described how a cylindrical tactile fingertip could recover the pose of a generalized convex cone from a small amount of data. Allen and Michelman (1990) fit a superquadric surface to sparse data obtained by a Utah-MIT hand around an object as its reconstructed shape. Moll and Erdmann (2002) showed how to simultaneously estimate the shape and motion of an unknown convex object from tactile readings on multiple manipulating palms under frictionless contact.

A method based on the interpretation tree or least-squares fitting needs to recover the pose. This may be costly and often unnecessary. Not until very recently did differential invariants start to find applications with touch sensing. For spheres, cylinders, cones, and tori, Keren et al. (2000) constructed descriptors in terms of curvatures and torsions and their derivatives (up to the third order) estimated at points on one or two curves embedded in these surfaces. Their derivation of differential invariants exploited Taylor expansion in the local frame. Shape parameters were solved through least-squares optimization.

In this paper, we derive semi-differential invariants that not only recognize several curve classes but also allow us to recover the algebraic descriptions of any curve from one of these classes based on curvature and derivative measurements.

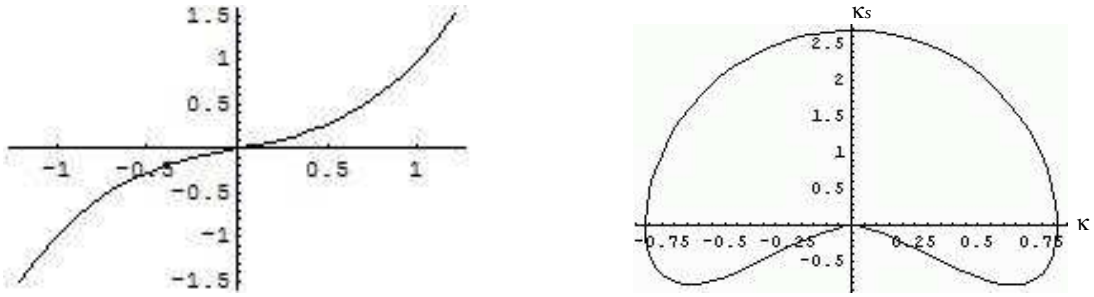
2 Curve Invariants

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 ϕ the tangential angle formed by the tangent of the boundary curve $\alpha(t) = (x(t), y(t))$ with the x-axis. The *curvature* κ is the rate of change of ϕ with respect to arc length s, that is,

$$\kappa = \frac{d\phi}{ds} = \frac{x'y'' - x''y'}{(x'^2 + y'^2)^{3/2}}. \tag{1}$$

Curvature is independent of parametrization, rotation, and translation. We are also interested in the derivative of curvature with respect to arc length:

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



(a) (b)

Figure 2: (a) A cubical parabola $y = 0.6x^3 + 0.4x$; (b) its signature curve $\{ (\kappa, \kappa_s) \mid -\infty < x < \infty \}$ that characterizes curvature and its derivative with respect to arc length along the curve. The signature curve is independent of Euclidean motions in the plane.

Section 6 will look at how to reliably estimate curvature and its derivative from tactile data. Until then we will just assume that *these two quantities are measurable.*

The touch sensor obtains data in Euclidean coordinates. The transformation group considered in this paper is the Euclidean group $SE(2)$ which includes all rotations and translations in the plane. A *differential invariant* of a curve is a real-valued function that does not depend on any specified transformation group or parametrization. More intuitively, the value of a differential invariant depends on the point location on the curve but not on the curve's position and orientation. The following result is well known (Guggenheimer 1977, Calabi et al. 1998):

Theorem 1 *Every Euclidean differential invariant of a plane curve is a function of the curvature κ and its derivatives with respect to arc length.*

2.1 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 2. It turns out that the signature curve of a curve determines its shape up to rotation and translation, as stated in the following theorem (Calabi et al. 1998).

Theorem 2 *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 (Pauwels et al. 1995, Rivlin and Weiss 1995, Calabi et al. 1998) based on matching signature curves. Construction of the signature curve, nevertheless, requires global shape information, which the touch sensor does not provide. Our aim is to make use of the local geometry at a small number of points (and thus much less data) and still be able to perform the recognition task.

2.2 Semi-Differential Invariants as Curve Descriptors

Suppose the curve $\alpha(t) = (x(t), y(t))$ is known to be from a certain family. Often we can derive a *canonical parametric form* of the family through proper rotation, translation, and reparametriza-

tion.[1] The canonical form describes the curve in a coordinate system determined by its geometry. It should have minimum number of indeterminacies (other than t) that parametrize the family. These independent indeterminacies are referred to as *shape parameters* and denoted by $a_1, \ldots, a_n$. For instance, the class of all ellipses are parametrized with the semimajor axis a and the semiminor axis b.

Since the parameter t, which specifies the location of contact on the shape (with the touch sensor), is not measurable, we try to eliminate it from the expressions (1) and (2) of curvature and derivative. If this is feasible, then the result is an equation for the signature curve:

$$f[a_1, \ldots, a_n](\kappa, \kappa_s) = 0. \tag{3}$$

The closed form of a signature curve can always be derived for an *algebraic curve*, which is defined by some equation $p(x, y) = 0$ with p a polynomial in x and y. Appendix C gives a derivation based on computing Sylvester resultants described in Appendix A.

The simplest case is when the function f can be split into two parts and rewritten as

$$I(\kappa, \kappa_s) = g(a_1, \ldots, a_n).$$

Then I is an expression whose value depends on the shape of the curve not on any specific point at which it is evaluated. It is thus an invariant for the curve, or a *curve invariant*. That the expression assumes the same value at different points is a necessary condition for an unknown curve to be from the family. The family of parabolas will be given as such an example in Section 3.1.

When a curve family has n shape parameters, we need n independent differential invariants to uniquely identify a curve from that family. If only one point on the curve is considered, this requires up to the nth derivative of the curvature (and thus $(n+2)$-nd derivative of the curve). Numerical computation of high order derivatives is very unreliable and sensitive to noise. The solution is to trade the order of derivative for extra points. So, we consider the curvatures and derivatives at n points and derive Euclidean *semi-differential invariants*. They are functions of the $2n$ curvatures and derivatives but assume values depending on $a_1, \ldots, a_n$ only. For some curves, such as ellipses and hyperbolas (Sections. 3.2–3.3), semi-differential invariants can be found through algebraic manipulation.

However, derivation of semi-differential curve invariants appears to be very difficult, if not impossible, for many curves. It is more likely that we have to solve for the shape parameters $a_1, \ldots, a_n$ using curvature and derivative estimates at $m \geq n$ points. Viewed differently, the semi-differential invariants now have values equal to the shape parameters but their evaluation can only be done through solution or optimization.

In the rest of the section, we focus on curves parameterized by polynomials. From (1) and (2) we derive two polynomial equations in t:

$$\kappa^2 \left(x'^2 + y'^2\right)^3 - (x'y'' - x''y')^2 = 0, \tag{4}$$

$$\frac{\kappa_s}{\kappa^2}(x'y'' - x''y')^2 - (x'y''' - x'''y')\left(x'^2 + y'^2\right) + 3(x'x'' + y'y'')(x'y'' - x''y') = 0. \tag{5}$$

Here, κ and κ_s can be measured using a method described in Section 6.1. They are thus not treated as indeterminacies. Suppose the polynomials $x(t)$ and $y(t)$ have degrees d_x and d_y, respectively.

[1] See Appendix B for an example of such derivation for cubic spline segments.

Equations (4) and (5) then have degrees $6\max\{d_x, d_y\} - 6$ and $d_x + d_y + 2\max\{d_x, d_y\} - 6$, respectively, in terms of t. We can eliminate t from these two equations by computing their resultant as described in Appendix A. This will yield a polynomial equation in terms of the coefficients $a_1, \ldots, a_n$ of the original curve. The equation, which defines the signature curve, can have degree as high as $d_x + d_y + 8\max\{d_x, d_y\} - 12$ in the n coefficients.

Once d_x or d_y exceeds 3, it becomes very difficult to rewrite the terms into some involving κ and κ_s and the others involving the shape parameters a_is only. Most likely, we will have to solve for these n shape parameters. The solution requires at least n points, each of which yields a polynomial equation. Since every such polynomial equation has degree at least 16 when $d_x > 3$ or $d_y > 3$, the task may become computationally too expensive, if not impossible.

2.3 Semi-Signature Curve

We can lower the degree of the resultant polynomial in $a_1, \ldots, a_n$ by considering the slope $\lambda = y'/x'$. This is rewritten as a polynomial equation:

$$\lambda x' = y'. \tag{6}$$

Assume that $x' \neq 0$ at every point measured by the touch sensor. Equations (1) and (5) are rewritten as

$$\kappa x'^2(1+\lambda^2)^{3/2} - (y'' - x''\lambda) = 0, \tag{7}$$

$$\frac{\kappa_s/\kappa^2 + 3\lambda}{1+\lambda^2}(y'' - x''\lambda)^2 + 3x''(y'' - x''\lambda) - (y''' - x'''\lambda)x' = 0. \tag{8}$$

With the slope λ treated as a separate variable, we compute the resultants of (6) with (7) and (8), respectively, to eliminate t and obtain

$$f_1[a_1, \ldots, a_n](\lambda, \kappa, \kappa_s) = 0 \quad \text{and} \quad f_2[a_1, \ldots, a_n](\lambda, \kappa, \kappa_s) = 0.$$

The two functions f_1 and f_2 have degrees not exceeding $3\max\{d_x, d_y\} - 3$ and $3\max\{d_x, d_y\} - 5$, respectively, in $a_1, \ldots, a_n$.

The slope λ depends on the orientation of the curve in its canonical parametrization but not on the position. For this reason, we refer to the point set $\{(\lambda, \kappa, \kappa_s)\}$ as the *semi-signature curve* of $\alpha(t)$. An example is shown in Figure 3. Its projection onto the κ-κ_s plane is the signature curve. The semi-signature curve is dependent on the chosen canonical parametrization.

What is the reason for using the slope λ defined in the canonical form? Though not able to measure λ directly, the touch sensor can measure the tangent rotation from one point to another. In Jia (2005), a jaw joined to a robot by a 2-axis force/torque sensor is aligned with the tangent direction at its contact with the object. Thus the tangent can be directly read from the robot's controller. In this paper, a joystick sensor is used in the experiments for more accurate contact measurement. Measurement of the tangent direction is carried out by local fitting (in the world coordinate system), as will be presented in Section 6.1.

The slope λ_1 at the first point and the slope λ_i at the ith point are related as

$$\lambda_i = \frac{\lambda_1 + \tan\Delta\theta_{1i}}{1 - \lambda_1\tan\Delta\theta_{1i}}, \qquad \theta_{1i} \text{ tangent rotation.} \tag{9}$$

Since all $\Delta\theta_{1i}$'s are measurable, as explained above, only one new variable λ_1 has been introduced.

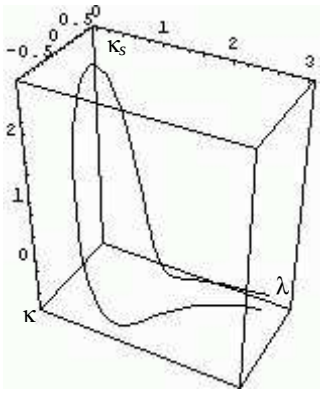


Figure 3: The semi-signature curve of the cubical parabola $y = 0.6x^3 + 0.4x$ drawn in Figure 2(a). It represents the point set $\{(\lambda, \kappa, \kappa_s) \mid -\infty < x < \infty\}$, where λ, κ, and κ_s are respectively the slope, curvature, and its derivative with respect to arc length.

Now, we may employ similar methods to derive expressions in terms of $\lambda, \kappa, \kappa_s$ but whose values depend on the shape parameters only. These "pseudo-invariants" together with (9) are solved for the slopes first, and then the shape parameters. The details will be described in Section 4 on cubical and semicubical parabolas and cubic spline curves.

3 Quadratics

A quadratic curve has the general form[2]:

$$c_{20}x^2 + 2c_{11}xy + c_{02}y^2 + 2c_{10}x + 2c_{01}y + c_{00} \;=\; 0. \tag{10}$$

It is well known that, through proper rotation and translation, the above equation can be transformed into one of the following canonical forms:

$$\begin{aligned}
\frac{x^2}{a^2} + \frac{y^2}{b^2} &= 1, && \text{if } c_{20}c_{02} - c_{11}^2 > 0; && \text{(ellipse)}\\
\frac{x^2}{a^2} - \frac{y^2}{b^2} &= 1, && \text{if } c_{20}c_{02} - c_{11}^2 < 0; && \text{(hyperbola)}\\
y^2 &= 4ax, && \text{if } c_{20}c_{02} - c_{11}^2 = 0. && \text{(parabola)}
\end{aligned}$$

These three types together are referred to as the *conics*.

To recognize a conic we may first identify its type and then recover the shape parameters a and/or b. To save some effort, we will start with some parametrization and derive expressions that are independent of the parametrization. These expressions build on local geometry at one or two points, namely, on their curvatures and derivatives with respect to arc length. But the values of the expressions are independent of specific points.

[2]The determinant

$$\Delta = \det \begin{pmatrix} c_{20} & c_{11} & c_{10} \\ c_{11} & c_{02} & c_{01} \\ c_{10} & c_{01} & c_{00} \end{pmatrix} \neq 0$$

but $\Delta/(c_{20} + c_{02}) < 0$ when $c_{20}c_{02} - c_{11}^2 > 0$.

3.1 Parabola

Parabolas are identified with curves parametrized by quadratic polynomials:

$$x = a_2t^2 + a_1t + a_0, \tag{11}$$
$$y = b_2t^2 + b_1t + b_0, \qquad a_2b_1 - a_1b_2 \neq 0. \tag{12}$$

To verify the statement, we first observe that every parabola $y^2 = 4ax$ has a parameterization in the form below:

$$\begin{aligned} x &= at^2, \\ y &= 2at. \end{aligned} \tag{13}$$

Conversely, given a curve parametrized as (11) and (12), we can treat each equation as a polynomial equation in t with x and y as constant terms. The resultant of the two equations is in the form of (10), where the coefficients c_{20}, $2c_{11}$, and c_{02} are b_2^2, $-2a_2b_2$, and a_2^2 respectively. Since $c_{20}c_{02} - c_{11}^2 = b_2^2a_2^2 - (-a_2b_2)^2 = 0$, we know that the curve is indeed a parabola.

The shape of a curve is independent of any particular parametrization. It makes sense to use the simplest one. So we use the canonical form (13) with only one shape parameter a.

First, we obtain the curvature and its derivative with respect to arc length:

$$\kappa = -\frac{1}{2a(t^2+1)^{3/2}}; \tag{14}$$
$$\kappa_s = \frac{\kappa'}{v(t)} = \frac{3t}{4a^2(t^2+1)^3}. \tag{15}$$

Equation (14) yields an expression for t^2:

$$t^2 = \frac{1}{(2a\kappa)^{2/3}} - 1. \tag{16}$$

Meanwhile, take the squares of both sides of (15):

$${\kappa_s}^2 = \frac{9t^2}{16a^4\,(t^2+1)^6}. \tag{17}$$

Substitution of (16) into (17) eliminates t^2 and yields an equation describing the signature curve of the parabola:

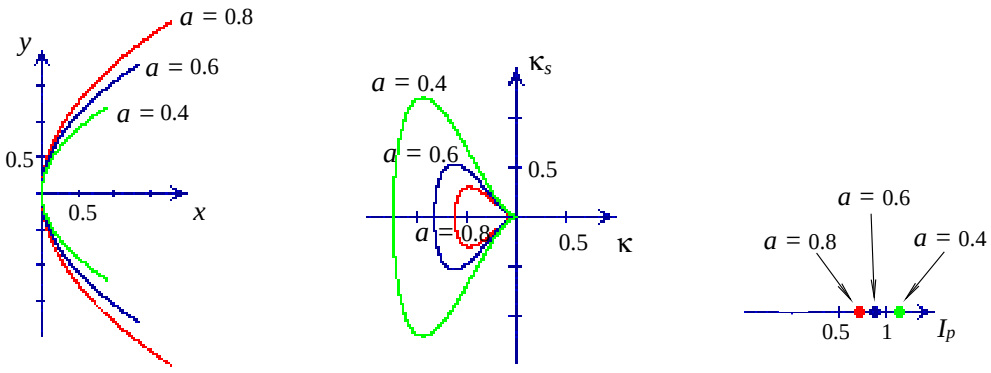
$$\kappa^{2/3}\left(\frac{\kappa_s^2}{9\kappa^4} + 1\right) = \frac{1}{(2a)^{2/3}}. \tag{18}$$

Define I_{p} as the expression on the left hand side of (18):

$$I_{\mathrm{p}}(\kappa, \kappa_s) \equiv \kappa^{2/3}\left(\frac{\kappa_s^2}{9\kappa^4} + 1\right). \tag{19}$$

This expression has value $1/(2a)^{2/3}$ which is independent of the point location t. It is an *invariant* that has a one-to-one correspondence to the shape of the parabola.

Figure 4 illustrates three parabolas distinguished by I_{p}. Since κ and κ_s are measurable, from (18) we can determine the shape parameter a and thus the parabola up to rotation and translation.



(a) (b) (c)

Figure 4: (a) Three parabolas of the form $y^2 = 4ax$; (b) their signature curves $\{(\kappa, \kappa_s) \mid -\infty < y < \infty\}$; (c) corresponding values of the invariant I_{p} defined in (19). The invariant is evaluated using an arbitrary point from each parabola.

3.2 Ellipse

Let us start with the canonical parametrization:

$$\begin{aligned} x &= a\cos(t), \\ y &= b\sin(t), \qquad a, b > 0. \end{aligned}$$

The curvature and its derivative with respect to arc length are

$$\kappa = \frac{ab}{\left(a^2\sin^2(t) + b^2\cos^2(t)\right)^{3/2}}, \tag{20}$$

$$\kappa_s = -\frac{3ab\left(a^2 - b^2\right)\sin(t)\cos(t)}{\left(a^2\sin^2(t) + b^2\cos^2(t)\right)^3}. \tag{21}$$

Again, we first derive an equation for the signature curve, and then move on to find some invariants.

Substitute the identity $\cos^2(t) = 1 - \sin^2(t)$ into (20) and obtain

$$\sin^2(t) = \frac{\left(\frac{ab}{\kappa}\right)^{2/3} - b^2}{a^2 - b^2}. \tag{22}$$

Next, square both sides of (21):

$${\kappa_s}^2 = \frac{9a^2b^2\left(a^2 - b^2\right)^2\sin^2(t)\left(1 - \sin^2(t)\right)}{\left(\left(a^2 - b^2\right)\sin^2(t) + b^2\right)^6}. \tag{23}$$

Substitute (22) into (23) to eliminate $\sin^2(t)$. A few more steps of manipulation result in an equation describing the signature curve of the ellipse:

$$I_{\mathrm{p}}(\kappa, \kappa_s) + \frac{1}{(ab\kappa)^{2/3}} - \frac{a^2 + b^2}{(ab)^{4/3}} = 0, \tag{24}$$

where $I_{\rm p}$ is the expression of κ and κ_s defined in (19).

Since there are two unknown quantities a and b, at least two points on the ellipse are required. Let κ_i and κ_{si}, $i = 1, 2$, be the curvature and its derivative at the ith point. Then we end up with two equations in the form of (24), for $i = 1, 2$, respectively. Subtracting one of them from the other yields the following:

$$I_{\rm p}(\kappa_1, \kappa_{s1}) + \frac{1}{(ab\kappa_1)^{2/3}} - I_{\rm p}(\kappa_2, \kappa_{s2}) - \frac{1}{(ab\kappa_2)^{2/3}} \quad = \quad 0.$$

Next, we rearrange the two sides of the above equation (assuming $\kappa_1 \neq \kappa_2$):

$$\frac{(\kappa_1\kappa_2)^{2/3}}{\kappa_1^{2/3} - \kappa_2^{2/3}}\Big(I_{\rm p}(\kappa_1, \kappa_{s1}) - I_{\rm p}(\kappa_2, \kappa_{s2})\Big) \quad = \quad \frac{1}{(ab)^{2/3}}. \tag{25}$$

Now, denote by $I_{\rm c1}$ the left hand side of equation (25). We have

$$\begin{aligned} I_{\rm c1}(\kappa_1, \kappa_2, \kappa_{s1}, \kappa_{s2}) \quad &\equiv \quad \frac{(\kappa_1\kappa_2)^{2/3}}{\kappa_1^{2/3} - \kappa_2^{2/3}}\Big(I_{\rm p}(\kappa_1, \kappa_{s1}) - I_{\rm p}(\kappa_2, \kappa_{s2})\Big) \qquad (26) \\ &= \quad \frac{(\kappa_1\kappa_2)^{2/3}}{\kappa_1^{2/3} - \kappa_2^{2/3}} \left(\kappa_1^{2/3}\left(\frac{\kappa_{s1}^2}{9\kappa_1^4} + 1\right) - \kappa_2^{2/3}\left(\frac{\kappa_{s2}^2}{9\kappa_2^4} + 1\right)\right), \qquad \text{by (19).} \end{aligned}$$

The expression $I_{\rm c1}$ 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 used.

Note that $I_{\rm c1} \cdot \pi^{2/3} = (\frac{\pi}{ab})^{2/3}$ is the constant affine curvature (Guggenheimer 1977, pp. 147-152) of the ellipse. The affine curvature of a plane curve involves up to the fourth order derivative at one point. Interestingly, the semi-differential invariant I_{c1} involves two points on the curve but only up to the third order derivative.

The invariant $I_{\rm c1}$ alone cannot distinguish ellipses with the same area ab. So we need a second invariant. Substitute $I_{\rm c1}(\kappa_1, \kappa_2, \kappa_{s1}, \kappa_{s2})$ for $1/(ab)^{2/3}$ into the second term in equation (24) for the first point:

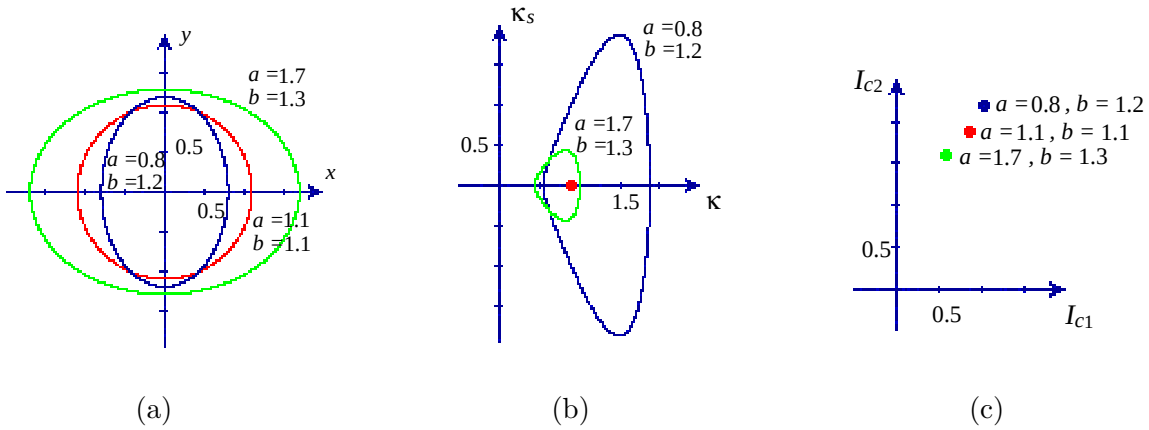
$$I_{\rm p}(\kappa_1, \kappa_{s1}) + \frac{I_{\rm c1}(\kappa_1, \kappa_2, \kappa_{s1}, \kappa_{s2})}{\kappa_1^{2/3}} = \frac{a^2 + b^2}{(ab)^{4/3}}. \tag{27}$$

The right hand side of the above equation depends on shape parameters a and b only. Now, denote its left hand side by $I_{\rm c2}$ and substitute (26) in:

$$\begin{aligned} I_{\rm c2}(\kappa_1, \kappa_2, \kappa_{s1}, \kappa_{s2}) \quad &\equiv \quad I_{\rm p}(\kappa_1, \kappa_{s1}) + \frac{\kappa_2^{2/3}}{\kappa_1^{2/3} - \kappa_2^{2/3}}\Big(I_{\rm p}(\kappa_1, \kappa_{s1}) - I_{\rm p}(\kappa_2, \kappa_{s2})\Big) \\ &= \quad \frac{1}{\kappa_1^{2/3} - \kappa_2^{2/3}} \Big(\kappa_1^{2/3} I_{\rm p}(\kappa_1, \kappa_{s1}) - \kappa_2^{2/3} I_{\rm p}(\kappa_2, \kappa_{s2})\Big) \\ &= \quad \frac{1}{\kappa_1^{2/3} - \kappa_2^{2/3}} \left(\kappa_1^{4/3}\left(\frac{\kappa_{s1}^2}{9\kappa_1^4} + 1\right) - \kappa_2^{4/3}\left(\frac{\kappa_{s2}^2}{9\kappa_2^4} + 1\right)\right). \qquad (28) \end{aligned}$$

From (27), $I_{\rm c2}(\kappa_1, \kappa_2, \kappa_{s1}, \kappa_{s2})$ always assumes the value $(a^2+b^2)/(ab)^{4/3}$. It is our second invariant.

A one-to-one correspondence exists between the tuples $(I_{\rm c1}, I_{\rm c2})$ and (a, b). Figure 5 compares two ellipses and a circle distinguished by the invariants $I_{\rm c1}$ and $I_{\rm c2}$.



(a) (b) (c)

Figure 5: (a) Three ellipses of 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}), where the two invariants are defined in (26) and (28), respectively. These invariants can be evaluated using two points arbitrarily chosen from an ellipse.

Having evaluated the invariants at two points, we can recover the shape parameters a and b. From the equations $I_{c1} = \frac{1}{(ab)^{2/3}}$ and $I_{c2} = \frac{a^2+b^2}{(ab)^{4/3}}$, we first obtain

$$\begin{aligned} ab &= \frac{1}{I_{c1}^{3/2}}, \\ a^2 + b^2 &= \frac{I_{c2}}{I_{c1}^2}. \end{aligned}$$

Then we solve for a and b (assuming $a \geq b$) from the above two equations:

$$\begin{aligned} a &= \sqrt{\frac{(a^2+b^2) + \sqrt{(a^2+b^2)^2 - 4(ab)^2}}{2}} \\ &= \sqrt{\frac{I_{c2} + \sqrt{I_{c2}^2 - 4I_{c1}}}{2I_{c1}^2}}, \qquad (29) \\ b &= \sqrt{\frac{I_{c2} - \sqrt{I_{c2}^2 - 4I_{c1}}}{2I_{c1}^2}}. \qquad (30) \end{aligned}$$

3.3 Hyperbola

A hyperbola has the canonical parametric form

$$\begin{aligned} x &= a\cosh(t) = a\,\frac{e^t + e^{-t}}{2}, \\ y &= b\sinh(t) = b\,\frac{e^t - e^{-t}}{2}, \qquad a, b > 0. \end{aligned}$$

The curvature and its derivative with respect to arc length are respectively

$$\kappa = -\frac{ab}{(a^2\sinh^2(t) + b^2\cosh^2(t))^{3/2}}, \qquad (31)$$

$$\kappa_s = \frac{3ab(a^2+b^2)\sinh(t)\cosh(t)}{(a^2\sinh^2(t)+b^2\cosh^2(t))^3}. \tag{32}$$

Using equations (31), (32), and $\cosh^2(t) = \sinh^2(t)+1$, we eliminate $\cosh(t)$ and $\sinh(t)$ and obtain the signature curve equation:

$$I_{\mathrm{p}}(\kappa,\kappa_s) - \frac{1}{(ab\kappa)^{2/3}} - \frac{a^2-b^2}{(ab)^{4/3}} = 0. \tag{33}$$

Two points on the hyperbola yields two copies of equation (33), from which we can derive two invariants:

$$\begin{aligned} I_{\mathrm{c1}}(\kappa_1,\kappa_2,\kappa_{s1},\kappa_{s2}) &= -\frac{1}{(ab)^{2/3}}, \\ I_{\mathrm{c2}}(\kappa_1,\kappa_2,\kappa_{s1},\kappa_{s2}) &= \frac{a^2-b^2}{(ab)^{4/3}}. \end{aligned}$$

These two invariants are in the same forms (26) and (28) as for an ellipse but their values are in different expressions of a and b. In particular, I_{c1} is always negative for the hyperbola.

The invariants I_{c1} and I_{c2} completely determine the hyperbola. Similar to the case of the ellipse, we first solve for a^2-b^2 and ab from I_{c1} and I_{c2}. Then the shape parameters are obtained (assuming $a \geq b$):

$$a = \sqrt{\frac{\sqrt{I_{\mathrm{c2}}^2 - 4I_{\mathrm{c1}}} + I_{\mathrm{c2}}}{2I_{\mathrm{c1}}^2}}, \tag{34}$$

$$b = \sqrt{\frac{\sqrt{I_{\mathrm{c2}}^2 - 4I_{\mathrm{c1}}} - I_{\mathrm{c2}}}{2I_{\mathrm{c1}}^2}}. \tag{35}$$

3.4 Invariants for Conics

The expressions I_{c1} and I_{c2} are also semi-differential invariants for a parabola. To see this, note that, at two points on a parabola, $I_{\mathrm{p}}(\kappa_1,\kappa_{s1}) = I_{\mathrm{p}}(\kappa_2,\kappa_{s2})$. From the definitions (26) and (28), we see that $I_{\mathrm{c1}} = 0$ and $I_{\mathrm{c2}} = I_{\mathrm{p}} = 1/(2a)^{2/3}$.

Thus I_{c1} and I_{c2} are invariants for all conics. They describe the correlations between the local geometry at any two points on a conic. The sign of the invariant I_{c1} discriminates one type of conic from another. When the invariant 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. For a given curve, if the values of I_{c1} and I_{c2} do not stay constant for different point tuples, we can conclude that the curve is not quadratic.

4 Cubics

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

$$\begin{aligned} x &= a_3t^3 + a_2t^2 + a_1t + a_0, \\ y &= b_3t^3 + b_2t^2 + b_1t + b_0, \qquad\qquad a_3 \neq 0 \text{ or } b_3 \neq 0. \end{aligned}$$

A canonical form can always be obtained through a sequence of translations, rotations, and reparametrizations described in Appendix B. This form represents either a cubical parabola:

$$\begin{aligned} x &= t, \\ y &= at^3 + ct, \qquad a \neq 0, \end{aligned} \tag{36}$$

or a semi-cubical parabola:

$$\begin{aligned} x &= t^2, \\ y &= at^3 + bt^2, \qquad a > 0, \end{aligned} \tag{37}$$

or a general spline segment:

$$\begin{aligned} x &= t^2, \\ y &= at^3 + bt^2 + ct, \qquad a > 0 \text{ and } c \neq 0. \end{aligned} \tag{38}$$

In the below, we examine these three subclasses one by one.

4.1 Cubical Parabola

From parametrization (36), the curvature and its derivative with respect to arc length are given as

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

Figures 2 and 3 plot a cubical parabola together with its signature and semi-signature curves.

Unfortunately, it is not obvious how to eliminate the parameter t from the above expressions for the curvature κ and its derivative κ_s. So we employ the method in Section 2.3 and utilize the slope

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

First, we rewrite the curvature (squared) and its derivative in terms of the slope:

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

Note that we can easily obtain a and c from the above once λ, κ, and κ_s are known:

$$a = \frac{(\kappa_s + 3\lambda\kappa^2)(1 + \lambda^2)^2}{6} \equiv I_{\text{cp1}}(\lambda, \kappa, \kappa_s), \tag{39}$$

$$c = \lambda - \frac{\kappa^2(1 + \lambda^2)}{2(\kappa_s + 3\lambda\kappa^2)} \equiv I_{\text{cp2}}(\lambda, \kappa, \kappa_s). \tag{40}$$

The expressions $I_{\rm cp1}$ and $I_{\rm cp2}$ map any point on the semi-signature curve to the shape parameters a and c, respectively. They are invariants of the curve provided that the slope λ in the canonical parametrization can be determined.

How to obtain the slope λ? We look at two points. Measure the tangent rotation $\Delta\theta_{12}$ from point 1 to point 2. This is done through fitting in Section 6.1. Write $\delta_{12} = \tan(\Delta\theta_{12})$. Since the value of c remains the same, we have

$$I_{\rm cp2}(\lambda_1, \kappa_1, \kappa_{s1}) = I_{\rm cp2}(\lambda_2, \kappa_2, \kappa_{s2}),$$

namely,

$$\lambda_1 - \frac{\kappa_1^2\,(1+\lambda_1^2)}{2\,(\kappa_{s1} + 3\lambda_1\kappa_1^2)} = \lambda_2 - \frac{\kappa_2^2\,(1+\lambda_2^2)}{2\,(\kappa_{s2} + 3\lambda_2\kappa_2^2)}. \tag{41}$$

The slopes λ_1 and λ_2 are related to each other through δ_{12} according to equation (9) with $i = 2$. Substitution of the equation into (41) results in a quartic polynomial:

$$d_4\lambda_1^4 + d_3\lambda_1^3 + d_2\lambda_1^2 + d_1\lambda_1 + d_0 = 0, \tag{42}$$

where the coefficients are

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

The polynomial equation (42) can be solved for λ_1 (and hence λ_2).

The values of κ_1, κ_{s1}, κ_2, κ_{s2} can all be estimated by a method to be presented in Section 6.1. Having solved for λ_1 and λ_2, we calculate the shape parameters a and c of the curve according to (39) and (40).

The invariants $I_{\rm cp1}$ and $I_{\rm cp2}$ of the cubical parabola are different from the invariants for quadratic curves, their evaluation requires the solution of the slope (λ_1) at one point. To verify the invariants, we may use, say, two pair of points. Solve for the slopes from (42) and then evaluate (39) and (40) for a and c. Compare whether their values stay the same.

4.2 Semi-Cubical Parabola

First, we reparametrize the curve (37) with the slope $\lambda = y'/x' = 3at/2 + b$ and derive the curvature and its derivative:

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

From the above two equations we determine the two shape parameters in terms of λ, κ, and κ_s:

$$a^2 = -\frac{8\kappa^3(1+\lambda^2)^{5/2}}{9(\kappa_s+3\lambda\kappa^2)} \equiv I_{\text{scp1}}(\lambda,\kappa,\kappa_s), \tag{43}$$

$$b = \lambda + \frac{\kappa^2(1+\lambda^2)}{\kappa_s+3\lambda\kappa^2} \equiv I_{\text{scp2}}(\lambda,\kappa,\kappa_s). \tag{44}$$

The invariants for this class of curves are I_{scp1} and I_{scp2}.

Using two points, we can set up an equation:

$$I_{\text{scp2}}(\lambda_1,\kappa_1,\kappa_{s1}) = I_{\text{scp2}}(\lambda_2,\kappa_2,\kappa_{s2}).$$

Eliminate λ_2 using the above equation and (9) with $i=2$. We again obtain a quartic polynomial in λ_1:

$$e_4\lambda_1^4 + e_3\lambda_1^3 + e_2\lambda_1^2 + e_1\lambda_1 + e_0 = 0,$$

where the coefficients are

$$\begin{aligned}
e_0 &= \kappa_1^2\left(\kappa_{s2}+3\kappa_2^2\delta_{12}\right) - \kappa_{s1}\left(\kappa_{s2}\delta_{12}+\kappa_2^2\left(1+4\delta_{12}^2\right)\right),\\
e_1 &= \delta_{12}\left(\kappa_{s1}\left(\kappa_{s2}\delta_{12}-3\kappa_2^2\right) - 5\kappa_1^2\left(\kappa_{s2}+3\kappa_2^2\delta_{12}\right)\right),\\
e_2 &= \kappa_1^2\left(\kappa_{s2}\left(1+4\delta_{12}^2\right) - 9\kappa_2^2\delta_{12}\right) - \kappa_{s1}\left(\kappa_{s2}\delta_{12}+\kappa_2^2\left(1+4\delta_{12}^2\right)\right),\\
e_3 &= \delta_{12}\left(\kappa_{s1}\left(\kappa_{s2}\delta_{12}-3\kappa_2^2\right) - 5\kappa_1^2\left(\kappa_{s2}+3\kappa_2^2\delta_{12}\right)\right),\\
e_4 &= 4\kappa_1^2\delta_{12}\left(\kappa_{s2}\delta_{12}-3\kappa_2^2\right).
\end{aligned}$$

Solving the above quartic polynomial gives us λ_1, and consequently, the values of shape parameters a and b, respectively.

Similar to the case of cubical parabolas, verification of the invariants I_{scp1} and I_{scp2} for semicubical parabolas requires the solution of the slope in the canonical parametrization (37).

4.3 General Cubic Spline Segment

It is time now to turn to a general cubic spline segment described by (38). The slope is given by

$$\lambda = \frac{3a}{2}t + b + \frac{c}{2t}.$$

Different from its two subclasses the cubical parabola and the semicubical parabola, we cannot replace t completely with the slope λ in the expressions of the curvature and its derivative:

$$\begin{aligned}
\kappa &= \frac{3at^2-c}{4|t^3|(1+\lambda^2)^{3/2}},\\
\kappa_s &= \frac{\kappa^2\left(1+\lambda^2\right)\left(3(\lambda-b)-6at\right)}{(\lambda-b)^2-3ac} - 3\kappa^2\lambda.
\end{aligned}$$

So we resort to solving equations (6)–(8), which are simplified to the following:

$$3at^2 + 2(b-\lambda)t + c = 0, \tag{45}$$

$$Lt^2 - 3at - (b-\lambda) = 0, \tag{46}$$

$$6at + M\Big((b-\lambda)^2 - 3ac\Big) + 3(b-\lambda) = 0, \tag{47}$$

where

$$L = \begin{cases} 2(1+\lambda^2)^{3/2}\kappa, & \text{if } t \geq 0; \\ -2(1+\lambda^2)^{3/2}\kappa, & \text{if } t < 0, \end{cases}$$
$$M = \frac{\kappa_s + 3\kappa^2\lambda}{(1+\lambda^2)\kappa^2}.$$

First, we substitute (45) into (47) to eliminate c:

$$9a^2Mt^2 + 6a\Big(1 + M(b-\lambda)\Big)t + M(b-\lambda)^2 + 3(b-\lambda) = 0. \tag{48}$$

Next, the resultant of equations (46) and (48) is computed to eliminate t:

$$81Ma^4 + 18L\Big(1 + 3M(b-\lambda)\Big)a^2 + L^2(b-\lambda)\Big(M(b-\lambda)+3\Big)^2 = 0. \tag{49}$$

Since M can get very large when κ is small, we divide the left hand side of (49) 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 λ_1 at the first point can be estimated through a least-squares optimization:

$$\min_{a,b,\lambda_1} \sum_{i=1}^{l} g(a, b, \lambda_i)^2, \tag{50}$$

where every slope λ_i depends on λ_1 according to (9) and can be calculated from λ_1 and the measurable rotation angle of the tangent from the first point to the ith point.

To determine the third parameter c, we multiply both sides of equation (45) by L while both sides of equation (46) by $3a$. Subtract the second resulting equation from the first one. Then we have

$$\Big(2(b-\lambda)L + 9a^2\Big)t + cL + 3a(b-\lambda) = 0.$$

Next, multiply the above equation by $6a$ and subtract the product of equation (47) with $2(b-\lambda)L + 9a^2$. After a few more steps of manipulation, we obtain

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

5 Invariant-Based Recognition

Satisfaction of the invariants of a curve family is the necessary condition for a given curve to be from that family. An invariant satisfied by one particular family but not by other families can be used for recognizing the family. Although in general such condition appears to be too strong, it is easier to meet if only a finite number of curve families are under consideration. In this section, we use simulations to verify the invariants derived in Sections 3 and 4. Then we look at how to apply them for recognizing curves and localizing them (through the recovery of point locations at which curvatures were estimated). These locations would correspond to the contacts made by a touch sensor.

5.1 Verification of Invariants

In the simulation, we approximate the arc length between two points on the curve, close to each other, by their Euclidean distance. The curvature and its derivative are estimated using finite difference quotients from arc length s and tangential angle ϕ:

$$\kappa \approx \frac{\phi(s+\Delta s)-\phi(s-\Delta s)}{2\Delta s} \quad \text{and} \quad \kappa_s \approx \frac{\phi(s+\Delta s)-2\phi(s)+\phi(s-\Delta s)}{(\Delta s)^2}. \tag{51}$$

The rotation of the tangent from one point to another uses the exact value since it can be measured quite accurately in an experiment. Given the errors of finite differences, it is not very meaningful to introduce simulated noise, which may either reduce or magnify such errors.

The first group of simulations was conducted to verify the invariants of the three conics, cubical parabolas, and semi-cubical parabolas. One curve from each class was selected. And each invariant was evaluated 100 times using points randomly selected from the corresponding curve. The results are summarized in Table 1. Estimation errors of κ and κ_s accounted for the discrepancies between

inv.	$I_{\rm p}$	$I_{\rm c1}$ (26)		$I_{\rm c2}$ (28)		$I_{\rm cp1}$	$I_{\rm cp2}$	$I_{\rm scp1}$	$I_{\rm scp2}$
	(18)	ellipse	hyperbola	ellipse	hyperbola	(39)	(40)	(43)	(44)
real	0.2198	0.1857	−0.2678	1.2055	0.3222	6.9963	2.6127	1.8851	6.5107
min	0.2168	0.1801	−0.2729	1.1749	0.2937	6.7687	1.7312	1.9912	6.3945
max	0.2230	0.1863	−0.2655	1.2083	0.3615	7.0289	3.1684	2.0871	6.5834
mean	0.2198	0.1852	−0.2675	1.2035	0.3210	6.9355	2.5022	2.0223	6.5154

Table 1: Invariant verification on five curves: a parabola, an ellipse, a hyperbola, a cubical parabola, and a semi-cubical parabola. The invariants are labeled with their defining equations. On each curve, the corresponding invariant(s) is evaluated 100 times using randomly generated points.

the actual values of the invariants and their estimates.

In the second group of simulations shown in Table 2, curvature and derivative data from every curve were used to evaluate invariants for curve classes to which it did not belong. A total of 100 evaluations were performed on each curve. The results demonstrate that *an invariant for one curve class differs from another.*

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

Table 2: Evaluating three invariants on data obtained from curves for which these invariants are not defined. Every entry represents the statistics on 100 evaluations of a "wrong" invariant using random data points.

Once the class of a curve is recognized, its shape parameters can be estimated using the values of the corresponding invariants. For a parabola, we simply have that $a = 1/(2I_{\rm p}^{3/2})$. For an ellipse, its two shape parameters are given in (29) and (30). For a hyperbola, they are given in (34) and (35). For the three cubics, the shape parameters are simply the values of the corresponding invariants[3].

Table 3 reveals how much the recovered shape parameters $\bar{a}, \bar{b}, \bar{c}$ differ from the real ones a, b, c. A total of 100 curves for each class (only 25 curves for the cubic spline class) were randomly generated under uniform distributions of its shape parameters within some prescribed ranges. We use a relative error form $\sqrt{((a-\bar{a})/a)^2 + ((b-\bar{b})/b)^2 + ((c-\bar{c})/c)^2}$. From the table we see that on the average the relative errors are around 1% except for cubic splines.

	ellip.	hyper.	par.	cub. par.	semi-cub. par.	cubic spline
min	0.02%	0.10%	0.01%	0.02%	0.04%	1.57%
max	7.99%	9.71%	3.35%	7.49%	8.09%	10.22%
mean	0.40%	1.15%	0.36%	0.83%	1.23%	5.68%

Table 3: Relative errors on shape parameter estimation immediately after the invariant-based recognition. The statistics are summarized over 100 randomly generated curves from each of the first five classes and 25 cubic splines.

5.2 Recognition Tree

Figure 6 illustrates a general recognition strategy for quadratic and cubic spline curves in the form of a decision tree. Given a curve, we estimate the values of curvature κ and its derivative κ_s at as few as three points. Then we test the invariants down the tree to identify the curve type or to determine that it is unclassified. Afterward, we calculate the shape parameters of the curve from the invariants (and thus completely determine the curve). Finally, the locations of the data points on the curve are recovered.

For example, consider the ellipse in Figure 7(a). The values of curvature and its derivative are estimated at $t_1 = 0.36$, $t_2 = 1.86$, and $t_3 = 4.23$. Invariant $I_{\rm c1}$ yields values 0.3447, 0.3446, and 0.3449 at the three resulting pairs of points. So the curve is quadratic. Since these values are greater than zero, we infer that the curve is an ellipse. This is confirmed by different values 0.8971 and 0.4030 of $I_{\rm p}$ at the first two points. The recovered shape parameters from $I_{\rm c1}$ and $I_{\rm c2}$ (according to (29) and (30)) are $a \approx 2.8609$ and $b \approx 1.7275$.

For the cubical parabola in Figure 7(b), a test on the invariant $I_{\rm c1}$ has failed. So we know that the curve is not quadratic. Hypothesizing cubical parabola, we solve for λ_1 using (42). Subsequent tests on the invariant $I_{\rm cp1}$ (for shape parameter a) yield values 3.2244, 3.2536, and 3.1872, and on the invariant $I_{\rm cp2}$ (for b) yield values -2.3237, -2.3237, and -2.3972. The recognition of the semi-cubical parabola in Figure 7(c) is similar.

The last curve is a segment from a closed cubic spline shown in Figure 7(d). We begin with three points t_1, t_2, and t_3. Tests on $I_{\rm c1}$, $I_{\rm cp1}$, and $I_{\rm scp1}$ have all failed. Hypothesizing that the curve is a cubic spline segment, we estimate the shape parameters: $a = 2.77468$, $b = -1.30929$, $c = -2.3081$. To verify the hypothesis, we use a fourth data point t_4 and re-estimate a, b, c using t_1, t_2, and t_4. The new estimates are $a = 2.78605$, $b = -1.3196$, $c = -2.30318$. Since the two sets of estimates agree quite well, the curve segment is recognized as a cubic spline segment.

[3]except in the case of a semi-cubical parabola, where the shape parameter a is the square root of the invariant $I_{\rm scp1}$.

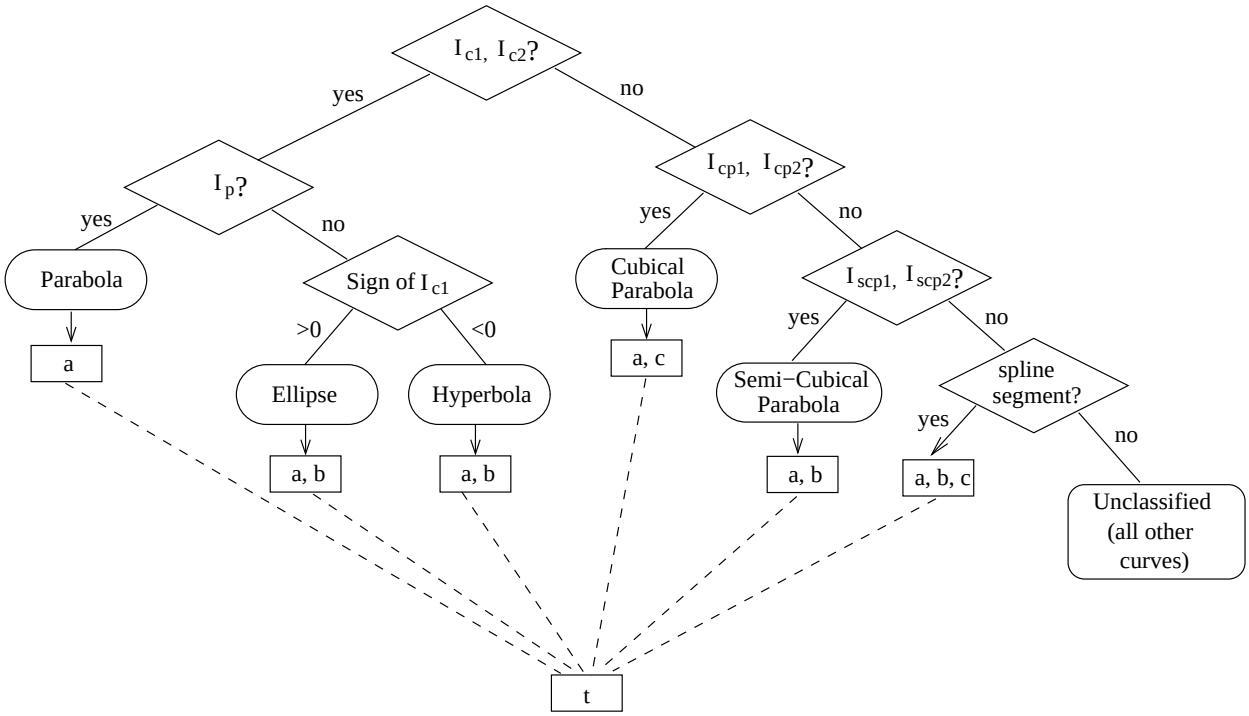
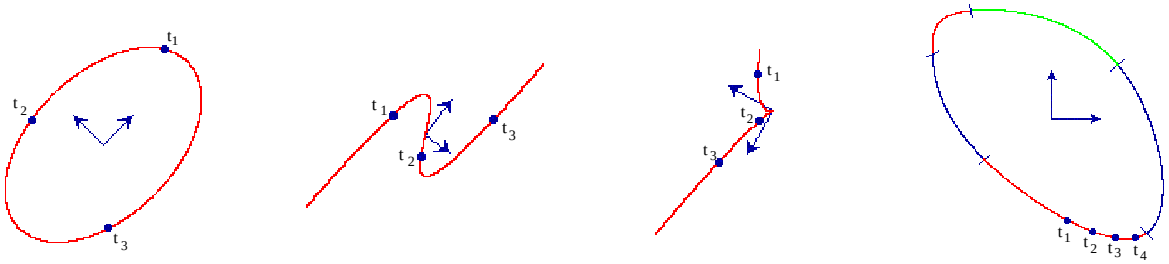
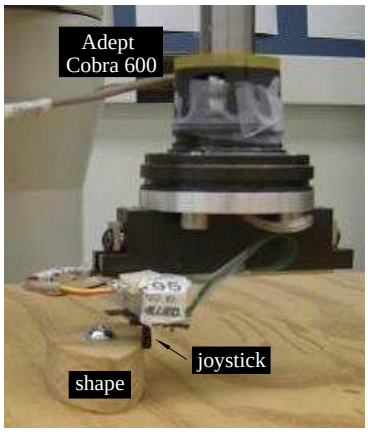


Figure 6: Recognition tree for quadratic and special cubic curves. It is a decision tree where every query node involves the evaluation of differential or semi-differential invariants.

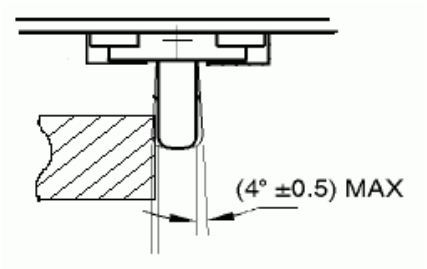


(a) (b) (c) (d)

Figure 7: Recognition of four shapes based on local geometry. Three data points are used in (a)-(c) each and four are used in (d). (a) An ellipse with $a = 2.8605$ and $b = 1.7263$; (b) a cubical parabola with $a = 3.2543$ and $c = -2.3215$; (c) a semi-cubical parabola with $a = 2.5683$ and $b = 1.4102$; and (d) one of five cubic spline segments with $a = 2.76866$, $b = -1.26705$, and $c = -2.32763$.



(a)



(b)

Figure 8: (a) Sensing a shape with a joystick sensor; (b) the sensor blowup.

5.3 Locating Contact

Having recognized the shape, the next step is to locate the points whose curvatures and derivatives are used in evaluating the invariants. Such a location is determined by the parameter value t. In the real situation, curvature is estimated from tactile data obtained with a touch sensor. So t locates the contact between the sensor and the curved shape. Since the tangent at the contact is measurable by the sensor, t also determines the relative pose of the shape to the sensor.

Finding the value of t is easy for the quadratic and cubic curves discussed in Sections 3 and 4. Here we simply give its expressions in terms of shape parameters, curvature and its derivative, and slope:

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

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

6 Experiments

In our experiments, tactile data were generated by a joystick sensor mounted on an Adept Cobra 600 robot[4], as shown in Figure 8. The joystick sensor is from Interlink Inc. and has a force range of 20-170g. The sensor's x- and y-axes were aligned with the robot's world coordinate frame so

[4]The robot has four DOFs though only the three positional DOFs were needed.

that force and position measurements had the same reference. The Adept robot has error ranges of ± 0.02mm in the x and y directions and ± 0.01mm in the z direction. A contact between the sensor and an object was detected when the sensor's force output exceeded certain threshold. The position of this contact was read from the robot controller. A calibration table was set up to compensate positional errors due to the joystick bending. The sensor was able to detect its contact with a shape within an error range of ± 0.1mm under calibration.

6.1 Slope and Curvature Estimation

The key for the applicability of our invariant-based approach lies in obtaining reliable estimates of curvature κ and its derivative κ_s from real tactile data. In the simulation, finite differences were used. But they are not expected to be robust in the presence of noise.

We first tested some existing numerical methods based on the Taylor expansion and finite differences. Curvature was estimated by approximating the osculating circle with the circle that passes through three adjacent points and then taking the inverse of its radius. This method was introduced in Calabi et al. (1998). The derivative of curvature was approximated by an involved difference quotient using the estimated curvature values (Boutin 2000). These two methods yield smaller numerical errors than the forms (51) used in our simulation earlier.

Some results of the above estimation are illustrated in Figure 9. From part (b) of the figure

Figure 9: Estimating κ and derivative κ_s from tactile data using the numerical methods from Calabi et al. (1998) and Boutin (2000): (a) an ellipse and 16 sample points; (b) estimates $(\hat{\kappa}, \hat{\kappa}_s)$ at these points plotted against the signature curve.

we see that the estimates of κ and κ_s were not good because most points (κ, κ_s) do not lie close to the signature curve for the ellipse. Since κ and κ_s are the second and third order derivatives of the curve, respectively, they are very sensitive to small measurement errors. This is why the two methods relying on Taylor expansions have failed on real data.

Differential filters (Meer and Weiss 1992) compute the derivatives of an image through convolution. The origin lies in fitting with higher order polynomials. Here we introduce a method that conducts two rounds of local fitting. The first round is performed to estimate the slope and the curvature, and the second round to estimate the derivative of the curvature with respect to arc length.

We fit a quadratic curve[5]

$$y = a_2x^2 + a_1x + a_0$$

over a sequence $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ of data points, generated by the joystick sensor on the shape boundary. The slope at the median point $p_1 = (x_{\lfloor n/2 \rfloor}, y_{\lfloor n/2 \rfloor})$ is approximated by $2a_2x + a_1$ in the world frame. The tangential angle θ_1 is determined accordingly.

The curvature at the point p_1 in the sequence is estimated through differentiating the curve fit:

$$\kappa_1 = \frac{2a_2}{\left(1 + \left(2a_2x_{\lfloor n/2 \rfloor} + a_1\right)^2\right)^{3/2}}.$$

Choose p_1 as the starting point so its arc length $s_1 = 0$. Next, we shift the sequence by l points ($l \ll n$ so that p_1 remains in the new sequence), and fit a new quadratic curve over $(x_{l+1}, y_{l+1}), (x_{l+2}, y_{l+2}), \ldots, (x_{n+l}, y_{n+l})$. Similarly, estimate the tangential angle θ_2 and curvature κ_2 at the median point p_2 of the new sequence. The tangent rotation $\Delta\theta_{12} = \theta_2 - \theta_1$ from p_1 to p_2 is independent of the shape orientation. It relates the slopes at p_1 and p_2 according to (9) so that only one of them needs to be determined. This relation is used in the least-squares optimization (50) which finds the shape parameters for a cubic spline segment.

Next, Simpson's method (Press et al. 2002, p. 136) is called upon to evaluate the integral

$$\int_{x_{\lfloor n/2 \rfloor}}^{x_{\lfloor n/2 \rfloor + l}} \sqrt{1 + (2a_2x + a_1)^2}\, dx,$$

which approximates the arc length between p_1 and p_2.

Now, we have two pairs $(0, \hat{\kappa}_1)$ and $(\hat{s}_2, \hat{\kappa}_2)$ of arc length and curvature estimates. Repeating this process m times generates a sequence of pairs $(0, \hat{\kappa}_1), (\hat{s}_2, \hat{\kappa}_2), \ldots, (\hat{s}_m, \hat{\kappa}_m)$. By fitting a quadratic curve over this sequence we obtain curvature as a function of arc length locally:

$$\kappa = b_2s^2 + b_1s + b_0.$$

Differentiating the above function gives us curvature derivative estimates at $0, \hat{s}_2, \ldots, \hat{s}_m$. Finally, we end up with m estimated pairs of κ and κ_s values: $(\hat{\kappa}_1, \hat{\kappa}_{s_1}), (\hat{\kappa}_2, \hat{\kappa}_{s_2}), \ldots, (\hat{\kappa}_m, \hat{\kappa}_{s_m})$. A total of $n + (m-1)l$ data points are used. In the experiments, we choose $n = 75$ and $l = 4$.

The curvature estimation strategy was applied on an elliptic part and a closed cubic spline part, shown in Figure 10(a) and (c), respectively. The model for the elliptic part is described by the equation $x^2/2.5^2 + y^2/1.75^2 = 1$. The model for the closed cubic spline part was the same as the curve in Figure 7(d). It had five segments.

Estimates $(\hat{\kappa}, \hat{\kappa}_s)$ at 20 points on the elliptic part were plotted against the signature curve for the part model, as shown in Figure 10(b). The estimates for one group of 10 points lie almost on the signature curve, while those for another group lie very close to the signature curve.

In Figure 10(c), eight sample point were taken out of one segment of the cubic spline part. Figure 10(d) shows the curvature and derivative estimates $(\hat{\kappa}, \hat{\kappa}_s)$ at these points. We observed good matches between the estimates and the signature curve for the corresponding segment in the shape model. Estimates obtained from two other segments also matched their corresponding signature curves well. But those from the two corner segments of the part produced poor matches. This was

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

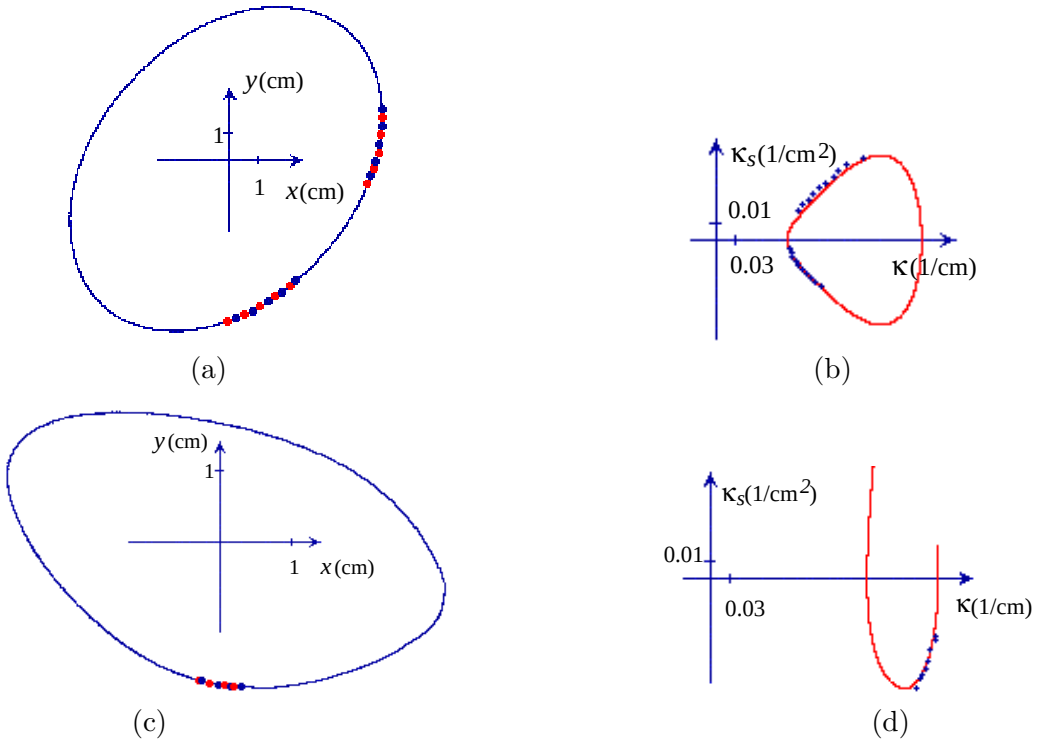


Figure 10: Estimating κ and derivative κ_s from tactile data generated by a joystick sensor: (a) an elliptic part and 20 estimation points; (b) estimates $(\hat{\kappa}, \hat{\kappa}_s)$ at these points plotted against the signature curve for the model curve of the part. (c) a closed cubic spline part with 8 estimation points from one of its segments; (d) estimates $(\hat{\kappa}, \hat{\kappa}_s)$ at these points and the signature curve for the corresponding segment in the part model. The estimates on curvature and its derivative are quite accurate as they lie very close to the signature curves.

mainly because, due to machining errors, the part's two corner sections were more rounded than in its geometric model. So the model segments have steeper signature curves than the corresponding segments of the real part. The eight pairs of estimates would have matched the signature curves for the actual corner sections better.

6.2 Validation of Invariants

The estimates $(\hat{\kappa}, \hat{\kappa}_s)$ at eighty pairs of points on the ellipse in Figure 10(a) were used for verifying the two invariants I_{c_1} and I_{c_2} for the ellipse. For each pair, the invariants were computed and the shape parameters a and b were recovered. The results are summarized in Table 4. As seen

	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

Table 4: Invariant evaluation and shape recovery of the elliptic part in Figure 10(a) from tactile data. The first row of data includes the part specification along with the values of the two invariants defined in (26) and (28). The last three rows are summaries over 80 values computed from different pairs of estimates $(\hat{\kappa}, \hat{\kappa}_s)$.

from the table, the average, minimum, and maximum estimated values of the invariants and shape parameters were all close to the exact values.

We also used two groups of three estimates $(\hat{\kappa}, \hat{\kappa}_s)$ out of the eight from a cubic spline segment in Figure 10(d). Due to inefficiency in nonlinear optimization, only two tests were performed. The results are summarized in Table 5. Larger errors in the shape parameter estimation were observed,

	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

Table 5: Recovering the cubic spline segment in Figure 10(d) from three of the eight data points. The segment is of the form (38).

compared to the case of the elliptic part.

7 Discussion

We have introduced an invariant-based method that aims at unifying shape recognition, recovery, and pose estimation with tactile information. Euclidean differential and semi-differential invariants have been developed for several classes of low-degree algebraic curves. Evaluated over a few points, these invariants capture intrinsic shape information about a curve. They allow us to recover the shape parameters of the curve as well as to obtain the locations of contact where the tactile data were supposed to be taken.

Although only quadratic curves and special cubic curves are treated in the paper, it is straightforward to extend the results to objects bounded by segments of these types. Nevertheless, data points need to be close enough to each other so they come from the same curve segment.

The derivation of a differential invariant consists of two steps. The first step is to eliminate the location parameter t or Euclidean coordinates to arrive at an equation that describes the signature curve. This is always achievable if the curve is algebraic. Appendix C shows that the implicit equation of an algebraic curve can be used along with curvature and its derivative to eliminate the two Euclidean coordinates x and y and yield a signature curve equation.

The second step takes the signature curve equation and separates terms involving the shape parameters from those involving curvature and its derivative with respect to arc length. This step turns out to be difficult if not impossible. If there are more than one shape parameters, at least an equal number of data points need to be used in deriving semi-differential invariants. This approach does not resort to higher order derivatives of the curvature (which are numerically unstable to evaluate). The curve slopes at these points, which are not invariant to rotation but whose dependencies on each other can be determined through tactile measurements, may be employed to facilitate invariant derivation.

We would like to investigate more on how to derive independent differential/semi-differential invariants for a given curve class. Unlike the situation with algebraic invariants, no automatic procedure of generating differential invariants is known to exist. Our aim is, nevertheless, to further the understanding on the limit of this invariant-based approach, both computationally and experimentally.

Extension to curves and surfaces in 3D is necessary for the invariant-based approach to become practical. For such a curve, invariants will have to involve torsion (and possibly its derivatives). For a surface, invariants will have to depend on the two principal curvatures and their derivatives. Invariant derivation is expected to be quite challenging, although curvature estimation is expected to rely on fitting to a local patch. We plan to conduct experiments on recognition and grasping of curved surfaces through minimal touch.

A The Resultant of Two Polynomials

The resultant of two univariate polynomials is an expression involving their polynomial coefficients such that the vanishing of the expression is necessary and sufficient for the two polynomials to have a common zero. More specifically, suppose we are given two polynomials:

$$\begin{aligned} f(x) &= a_n x^n + a_{n-1} x^{n-1} + \cdots + a_0, \qquad a_n \neq 0, \\ g(x) &= b_m x^m + b_{m-1} x^{m-1} + \cdots + b_0, \qquad b_m \neq 0. \end{aligned}$$

Consider the following $(m+n) \times (m+n)$ matrix made up of m rows of coefficients a_i followed by n rows of coefficients b_j:

$$A = \begin{pmatrix} a_n & a_{n-1} & \cdots & a_0 & & \\ & \ddots & & \vdots & & \\ & & a_n & a_{n-1} & \cdots & a_0 \\ b_m & b_{m-1} & \cdots & b_0 & & \\ & \ddots & & \vdots & & \\ & & b_m & b_{m-1} & \cdots & b_0 \end{pmatrix}. \tag{52}$$

The *Sylvester resultant* of the polynomials $f(x)$ and $g(x)$ is defined to be $R = \det(A)$, which is homogeneous of degree m in a_i and homogeneous of degree n in b_j (van der Waerden 1970, pp. 102–105). The following result holds.

Theorem 3 *If the resultant R vanishes, the polynomials f and g have a common non-constant factor (thus a common nontrivial zero), and conversely.*

Suppose a curve is parametrized as $(p(t), q(t))$, where $p(t)$ and $q(t)$ are polynomials of degrees n and m in t, respectively. We can implicitize the curve by eliminating the parameter t. This is done by treating the coordinates x and y as symbolic coefficients and computing the resultant of two polynomial equations in t:

$$\begin{aligned} p(t) - x &= 0, \\ q(t) - y &= 0. \end{aligned}$$

The major diagonal of the matrix A in (52) determines that the resultant is a polynomial equation in x and y. Furthermore, the degree of this polynomial is no greater than the maximum degree of p and q.

In general, suppose we are given m polynomial equations in n variables $x_1, x_2, \ldots, x_n$:

$$\begin{aligned} p_1(x_1, x_2, \ldots, x_n) &= 0, \\ p_2(x_1, x_2, \ldots, x_n) &= 0, \\ &\vdots \\ p_m(x_1, x_2, \ldots, x_n) &= 0. \end{aligned}$$

We can first eliminate x_n by treating $x_1, x_2, \ldots, x_{n-1}$ as constant terms. This yields $m-1$ polynomials in $x_1, x_2, \ldots, x_{n-1}$. Repeating the above step to eliminate the remaining variables one by one, until all variables have been got rid of or only one equation is left. The process generalizes Gaussian elimination to a system of nonlinear equations.

B Canonical Form of a Cubic Spline Segment

A segment of a cubic spline has the general form:

$$\begin{aligned} x &= a_3 t^3 + a_2 t^2 + a_1 t + a_0, \\ y &= b_3 t^3 + b_2 t^2 + b_1 t + b_0, \qquad a_3 \neq 0 \text{ or } b_3 \neq 0. \end{aligned} \tag{53}$$

First, we translate the curve by $(-a_0, -b_0)$ to eliminate the constant terms:

$$\begin{aligned} x &= a_3 t^3 + a_2 t^2 + a_1 t, \\ y &= b_3 t^3 + b_2 t^2 + b_1 t. \end{aligned}$$

Next, we perform a rotation by $\theta = \arctan\left(\frac{a_3}{b_3}\right)$ to remove the cubic term in x, yielding

$$\begin{aligned} x &= c_2 t^2 + c_1 t, \\ y &= d_3 t^3 + d_2 t^2 + d_1 t, \end{aligned} \tag{54}$$

where

$$\begin{aligned} c_i &= (a_i \cos\theta - b_i \sin\theta), && i = 1, 2; \\ d_j &= (a_j \sin\theta + b_j \cos\theta), && j = 1, 2, 3. \end{aligned}$$

Under the condition $a_3 \neq 0$ or $b_3 \neq 0$, it is easy to show that i) $d_3 \neq 0$ and ii) $\cos\theta = 0$ if and only if $b_3 = 0$. Two cases arise in further transformations:

(a) $a_2 b_3 - a_3 b_2 = 0$ It follows that $c_2 = 0$. When $a_1 b_3 - a_3 b_1 = 0$, c_1 vanishes. Subsequently, $x = 0$ and the curve degenerates into the y-axis. So we are interested in the non-degenerate case $c_1 \neq 0$, which is implied by $a_1 b_3 - a_3 b_1 \neq 0$. We reparametrize (54) with $t = \frac{u}{c_1} - \frac{d_2}{3d_3}$ to eliminate the quadratic term in y and then translate the curve to eliminate the new constant terms in both x and y. The result is the simplest form:

$$\begin{aligned} x &= u, \\ y &= au^3 + cu, \end{aligned}$$

where the new coefficients are

$$\begin{aligned} a &= \frac{d_3}{c_1^3}, \\ c &= \frac{d_1}{c_1} - \frac{d_2^2}{3c_1 d_3}. \end{aligned}$$

The curve is a *cubical parabola*.

(b) $a_2 b_3 - a_3 b_2 \neq 0$ It follows that $c_2 \neq 0$. In case of a negative c_2, rotate the curve by π to make the coefficient positive. We first reparametrize (54) using $t = \frac{u}{\sqrt{c_2}} - \frac{c_1}{2c_2}$ and then translate the curve to eliminate the constant terms emerging in x and y. This results in a canonical form:

$$\begin{aligned} x &= u^2, \\ y &= au^3 + bu^2 + cu, \end{aligned}$$

where

$$\begin{aligned} a &= \frac{d_3}{c_2^{3/2}}, \\ b &= \frac{d_2}{c_2} - \frac{3c_1 d_3}{2c_2^2}, \\ c &= \frac{3c_1^2 d_3}{4c_2^{5/2}} - \frac{c_1 d_2}{c_2^{3/2}} + \frac{d_1}{\sqrt{c_2}}. \end{aligned}$$

We can always ensure that $a > 0$. For if $a < 0$, we just substitute $-u$ for u in the parametrization and the coefficient of u^3 in y will then become $-a > 0$.

When c vanishes, the curve is a *semi-cubical parabola*. This happens, for example, if $a_1 = b_1 = 0$ in (53).

C Signature Curve for an Algebraic Curve

An algebraic curve is determined by some polynomial equation

$$f(x, y) = 0 \tag{55}$$

in x and y. Any curve parametrized as $\boldsymbol{\alpha}(t) = (p(t), q(t))$, where $p(t)$ and $q(t)$ are polynomials in t, is an algebraic curve. We can eliminate t as described in Appendix A.

Let us look at how to derive the signature curve equation for a given algebraic curve $f(x, y) = 0$. Denote the gradient by $\nabla f = (f_x, f_y)$, where f_x and f_y are the partial derivatives of f with respect to x and y, respectively. In the neighborhood of a non-singular point (where $\nabla f \neq 0$), the algebraic curve has a local parameterization by $\boldsymbol{\alpha}(t) = (x(t), y(t))$.

Differentiating the equation

$$f\Big(x(t), y(t)\Big) = 0,$$

according to the chain rule, we obtain

$$\boldsymbol{\alpha}' \cdot (f_x, f_y) = 0.$$

Therefore, $\boldsymbol{\alpha}'$ is orthogonal to (f_x, f_y), and we have $\boldsymbol{\alpha}' = (x', y') = \mu(f_y, -f_x)$, where $\mu \neq 0$ is a function of t. The Hessian matrix of $f(x, y)$ is

$$H = \begin{pmatrix} f_{xx} & f_{xy} \\ f_{xy} & f_{yy} \end{pmatrix}.$$

On differentiating once more we obtain

$$\boldsymbol{\alpha}'' \cdot (f_x, f_y) + (x', y')H \begin{pmatrix} x' \\ y' \end{pmatrix} = 0.$$

Therefore we have

$$\begin{aligned} x'y'' - y'x'' &= \boldsymbol{\alpha}'' \cdot (-y', x') \\ &= \mu \boldsymbol{\alpha}'' \cdot (f_x, f_y) \\ &= -\mu(x', y')H \begin{pmatrix} x' \\ y' \end{pmatrix} \\ &= -\mu^3(f_y, -f_x)H \begin{pmatrix} f_y \\ -f_x \end{pmatrix}. \end{aligned}$$

Now, substitute the above equation as well as $x' = \mu f_y$ and $y' = -\mu f_x$ into equation (4) and then divide its both sides by μ^6:

$$\kappa^2 \left(f_x^2 + f_y^2\right)^3 - \left((f_y, -f_x)H \begin{pmatrix} f_y \\ -f_x \end{pmatrix}\right)^2 = 0. \tag{56}$$

Next, we differentiate equation (56), obtaining

$$\begin{aligned} & 2\kappa\kappa' \left(f_x^2 + f_y^2\right)^3 + \kappa^2 \cdot 3 \left(f_x^2 + f_y^2\right)^2 \cdot \left(2f_x \cdot (f_{xx}x' + f_{xy}y') + 2f_y \cdot (f_{xy}x' + f_{yy}y')\right) \\ & \quad - 2 \left((f_y, -f_x) H \begin{pmatrix} f_y \\ -f_x \end{pmatrix}\right) \\ & \quad \left(2(f_{xy}x' + f_{yy}y', -f_{xx}x' - f_{xy}y') H \begin{pmatrix} f_y \\ -f_x \end{pmatrix} + (f_y, -f_x) \frac{dH}{dt} \begin{pmatrix} f_y \\ -f_x \end{pmatrix}\right) \quad = \quad 0. \end{aligned} \tag{57}$$

The derivative of the Hessian is linear in both x' and y' and thus a multiple of μ:

$$\frac{dH}{dt} = \begin{pmatrix} f_{xxx}x' + f_{xxy}y' & f_{xxy}x' + f_{xyy}y' \\ f_{xxy}x' + f_{xyy}y' & f_{xyy}x' + f_{yyy}y' \end{pmatrix}.$$

Substitute into (57) the derivatives x', y', and

$$\kappa' \quad = \quad \frac{d\kappa}{ds}\frac{ds}{dt} = \kappa_s \cdot |\mu| \left(f_x^2 + f_y^2\right)^{1/2},$$

and then divide both sides by 2μ. Next, we move all but the first term to the right hand side of the resulting equation. The left hand side becomes $\pm\kappa\kappa_s(f_x^2 + f_y^2)^{7/2}$. Squaring both sides of the equation gives us a polynomial equation in κ, κ_s, and up to the third order partial derivative of f. Eliminate x and y from this new polynomial equation and (55) and (56) using the method described in Appendix A. The result is an equation describing the signature curve.

Acknowledgment 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. The authors would like to thank David Kriegman for pointing to differential invariants for shape recognition, and Liangchuan Mi (Mi and Jia 2004) for providing the tactile data used in the experiments. The authors are grateful to the reviewers for their thorough reading of the paper and valuable feedback. Portions of this paper were presented at the 2004 IEEE International Conference and Robotics and Automation (ICRA '04) (Ibrayev and Jia 2004) and at the Sixth International Workshop on Algorithmic Foundations of Robotics (WAFR '04) (Jia and Ibrayev 2004). The authors would like to thank the corresponding reviewers for their helpful comments.

References

[1] Allen, P. K., and Michelman, P. 1990. Acquisition and interpretation of 3-D sensor data from touch. *IEEE Trans. Robot. and Automation*, 6(4):397–404.

[2] Boutin, M. 2000. Numerically invariant signature curves. *Intl. J. Comp. Vision*, 40(3):235–248.

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

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

[5] Fearing, R. S. 1990. Tactile sensing for shape interpretation. In S. T. Venkataraman and T. Iberall (eds.), *Dextrous Robot Hands*, pp. 209–238. Springer-Verlag.

[6] Forsyth, D., Mundy, J. L., Zisserman, A., Coelho, C., Heller, A., and Rothwell, C. 1991. Invariant descriptors for 3-D object recognition and pose. *IEEE Trans. Pattern Analysis and Machine Intell.*, 13(10).

[7] Grimson, W. E. L., and Lozano-Pérez, T. 1984. Model-based recognition and localization from sparse range or tactile data. *Intl. J. Robot. Res.*, 3(3):3–35.

[8] Guggenheimer, H. W. 1977. *Differential geometry*. Dover Publications.

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

[10] Ibrayev, R., Jia, Y.-B. 2004. Tactile recognition of algebraic shapes using differential invariants. *Proc. IEEE Intl. Conf. Robot. and Automation (ICRA)*, New Orleans, LA, pp. 1548-1553.

[11] Jia, Y.-B. 2005. Localization of curved parts through continual touch. *IEEE Trans. Robotics*, 21(4):726-733.

[12] Jia, Y.-B., and Ibrayev, R. 2004. Semi-differential invariants for recognition of algebraic curves. In *Proceedings of the 6th International Workshop on Algorithmic Foundations of Robotics*, Ziest, the Netherlands.

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

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

[15] Kriegman, D. J., and Ponce, J. 1990. On recognizing and positioning curved 3-D objects from image contours. *IEEE Trans. Pattern Analysis and Machine Intell.*, 12(12):1127–1137.

[16] Meer, P., and Weiss, I. 1992. Smoothed differentiation filters for images. *J. Visual Commu. and Image Representation*, 3(1):58-72.

[17] Mi, L., and Jia, Y.-B. 2004. High precision contour tracking with a joystick sensor. In *Proc. IEEE/RSJ Intl. Conf. Intell. Robots and Systems*, pp. 804–809.

[18] Moll, M., and Erdmann, M. A. 2004. Reconstructing the shape and motion of unknown objects with active tactile sensors. In J.-D. Boissonnat *et al.* (eds.), *Algorithmic Foundations of Robotics V*, pp. 293–309. Springer-Verlag.

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

[20] Mundy, J. L., and Zisserman, A. 1992. Introduction — towards a new framework for vision. In J. L. Mundy and A. Zisserman (eds.), *Geometric Invariance in Computer Vision*, pp. 1–39. The MIT Press.

[21] Pajdla, T., and Van Gool, L. 1995. Matching of 3-D curves using semi-differential invariants. In *Proc. IEEE Intl. Conf. Comp. Vision*, pp. 390–395.

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

[23] Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. 2002. *Numerical Recipes in C++*, 2nd ed. Cambridge University Press, 2002.

[24] Rivlin, E., and Weiss, I. 1995. Local invariants for recognition. *IEEE Trans. Pattern Analysis and Machine Intell.*, 17(3):226–238.

[25] van der Waerden, B. L. 1970. *Modern Algebra*, vol. 1. Frederick Ungar Publishing Co.

[26] Weiss, I. 1993. Geometric invariants and object recognition. *Intl. J. Comp. Vision*, 10(3):207–231.