Volume xxx, (1994) number yyy pp. 000–000

Isometric Piecewise Polynomial Curves

Eugene Fiume†

Department of Computer Science, University of Toronto, 10 King's College Circle, Toronto, Canada, M5S 1A4

Abstract

The main preoccupations of research in computer-aided geometric design have been on shape-specification techniques for polynomial curves and surfaces, and on the continuity between segments or patches. When modelling with such techniques, curves and surfaces can be compressed or expanded arbitrarily. There has been relatively little work on interacting with direct spatial properties of curves and surfaces, such as their arc length or surface area. As a first step, we derive families of parametric piecewise polynomial curves that satisfy various positional and tangential constraints together with arc-length constraints. We call these curves isometric curves*. A space curve is defined as a sequence of polynomial curve segments, each of which is defined by the familiar Hermite or Bézier constraints for cubic polynomials; as well, each segment is constrained to have a specified arc length. We demonstrate that this class of curves is attractive and stable. We also describe the numerical techniques used that are sufficient for achieving realtime interaction with these curves on low-end workstations.*

Keywords: isometric piecewise parametric polynomial curves, arc length, computer-aided geometric design, numerical methods.

1. Introduction

Research into the specification of piecewise curves and surfaces has very successfully attacked the problem of shape control and continuity across curve segments or surface patches[1, 2]. The main theme of these approaches is to define an overall space curve $\mathcal{C}(t)$ as a set of piecewise polynomials $\mathbf{p}_0, \mathbf{p}_1, \cdots, \mathbf{p}_m$. All curve segments are usually of the same low degree n, and most often $n = 3$. Each segment $\mathbf{p}_i(t)$ is defined by means of a sequence of control points $P_{ij},\ j = 0, \cdots, n$ that are weighted together by $n+1$ blending functions $B_{ij}(t)$ as in

$$\mathbf{p}_i(t) = \sum_{j=0}^{n} B_{ij}(t) P_{ij}$$

to form an overall piecewise polynomial curve of degree n. Typically the blending functions are the same for each curve segment i, so in this case we can write B_j rather than B_{ij}. There are many variations on this theme that enforce continuity and a variety of shape specification possibilities. Among the desired properties of a curve specification is *affine invariance*, sometimes also called *co-ordinate system independence*, which requires that for an affine transformation τ,

$$\tau \mathbf{p}_i(t) = \sum_{j=0}^{n} B_{ij}(t) \tau P_{ij}.$$

Working with control points is a convenient way of specifying curves and surfaces. However, this extra layer of indirection can make it difficult to work with actual spatial properties of the object, such as its surface area, curvature or arc length. The need to preserve length, area or volume often arises in computer animation and in physical and geometric modelling, but this is generally part of a non-interactive postprocess.

An interesting challenge, then, is to derive sets of blending functions that guarantee arc-length or

† The financial support of the Natural Sciences and Engineering Research Council of Canada and of the Information Technology Research Centre of Ontario is gratefully acknowledged.

surface-area invariance, but that retain traditional control-point techniques for interactivity. Furthermore, we wish such solutions to be as independent of co-ordinate system as possible. As a first step, we consider the problem of maintaining segment-wise arc-length constraints for piecewise polynomial curves of degrees 1-4. For curves of degree 3 and 4, the specification is based on the familiar cubic Hermite or Bézier basis. The basis will not be linear in arc length, but for a given arc length, the solution is both translation and rotation invariant. Our goals are to develop a curve-specification scheme that naturally extends familiar techniques, that has local control, that is largely co-ordinate system independent, and that can be efficiently implemented. Our resolution of these goals is a family of *isometric polynomials*.

An extremely-interesting related problem that we shall not discuss in depth is the following: given a sequence of points (possibly mixed with tangents and possibly with nonuniform knot spacing) P_i, $i = 0, \cdots, n$, derive an isometric curve $\mathcal{C}(t)$ of degree m that "optimally" interpolates the points. The notion of optimality may be based on quasi-physical notions such as strain energy, or geometric properties of the curve. Jou and Han give a mathematical development for generally nonpolynomial, arc-length parameterised curves of specified arc-length and interpolation constraints[3]. Roulier avoids arc-length parameterisation and concentrates on Bézier curves of arbitrary degree that meet arc-length and convexity constraints[4]. Although neither paper discusses the issue at length, it is possible to use either approach in a piecewise fashion to get some semblance of local control. What is less clear is whether these formulations can be used in an interactive design setting. Low-degree formulations with piecewise continuity have traditionally been used in this context. This is the focus of our work. Another motivation for our formulation is its expected ease of extension to surface-area constraints.

The use of isometric curves is most obvious in the specification of trajectories for computer animation: we may wish an object to follow a route of a certain distance over some time interval, perhaps meeting certain velocity and positional constraints. Another use is in texture-mapping, which has already been recognised (cf. Bennis *et al.*[5]). A third use is in the interpolation of scattered data (e.g., Lee[6]). Isometric curves have an string or wire-like feel about them, and are suitable for modelling such inelastic phenomena. Our curves may also be useful in filtering, where ringing in a reconstructed signal may be damped by "pulling" on the curve using an arc-length constraint. Ultimately, we wish to extend our approach to isometric surfaces including real "patches" of certain kinds of cloth, paper, moulded plastic, sheet metal, etc.

Our isometric curves are not necessarily *arc-length parameterised* curves. In fact, they are extremely unlikely to be so (see Farouki and Sakkalis[7]). An arc-length parameterised curve is such that equal steps in the parameter give equal steps in arc length along the curve. Such objects would be ideal candidates for minimising distortion in texture mapping, but there are very few classes of polynomials that can be analytically arc-length parameterised: the class is restricted to lines. Girard considered this problem in a trajectory-specification context and derived approximate table lookup techniques for arc-length reparameterisations of user-specified curves[8]. An interesting open problem is whether or not our isometric curve formulation easily admits arc-length reparameterisation.

Our curve formulation is based on non-rational polynomials. The generalisation of the approach to rational polynomial schemes is both interesting and challenging. The appearance of a polynomial in the denominator as well as the numerator results in a formulation that appears to be more difficult to solve than the non-rational form. This is an excellent topic for future research.

The next section formulates isometric curves of degrees 1-4. We then discuss the numerical methods required to develop an efficient implementation.

2. Mathematical Formulation of Isometric Curves

We shall formulate an arc-length constrained polynomial curve segment from first principles for curve segments of various degrees. The term *isometric* is synonymous for our purposes with *arc-length constrained*. Throughout, our discussion will be concerned with parametric curves $\mathbf{p}(t) = (x(t), y(t)),\ t \in [0, 1]$ in the plane. The extension to general space curves in R^n is straightforward.

From differential geometry[9, 2], the arc length of a space curve $\mathbf{p}(t)$ on domain $[a, b]$ is defined as

$$A(\mathbf{p}; [a, b]) = \int_a^b \|\dot{\mathbf{p}}(t)\|\, dt, \tag{1}$$

where $\dot{\mathbf{p}}$ denotes the component-wise derivative of $\mathbf{p}$ with respect to t. If $\mathbf{p}(t)$ is defined parametrically as $(x(t), y(t))$, then the derivative can be taken componentwise (giving a velocity), and the expression for arc length is thus

$$\begin{aligned} A(\mathbf{p}; [a, b]) &= \int_a^b \|(\dot{x}(t), \dot{y}(t))\|\ dt \\ &= \int_a^b \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2}\ dt. \end{aligned} \tag{2}$$

In the special case in which $\mathbf{p} = (x, y)$ is an explicit function of x, that is $y = f(x)$, then it is easy to show that

$$A(\mathbf{p}; [a, b]) = \int_a^b \sqrt{1 + \dot{f}(x)^2}\, dx. \qquad (3)$$

Analytic solutions for these elliptic integrals only exist for very simple functions. Among the polynomials, for example, analytic solutions exist only for polynomials of degree two or lower. Perhaps this fact helps to account for the lack of work in isometric polynomial curve specification. Another factor is that isometric polynomial curve families do not admit a linear polynomial basis. However, we shall see that it is possible to define such families that are "quasi-linear" in the sense that they are translation and rotation invariant, and that have the convex hull property. Furthermore, we shall demonstrate that the computation of these constraints is quite feasible and that the resulting curves are pleasing.

By way of example, consider the cubic space curve

$$\mathbf{p}(t) = (t^3 - 2t^2 + 3t + 1,\ 2t^3 + t^2 - 2t - 1).$$

A plot of this curve for $t \in [-1, 1]$ can be found in Figure 1. A plot of the arc-length integrand for this curve can be found in Figure 2. The area under this curve is the arc length. Notice that despite the variation in $\mathbf{p}$, the arc-length integrand is quite smooth, and would be even smoother after integration. This is borne out by Figure 3, which depicts the monotone-increasing integral function $A(\mathbf{p}(t); [-1, t])$. A four-term Simpson's rule evaluation of $A(\mathbf{p}(t); [-1, 1])$ gives a relative error of 1%, which is almost acceptable for display-screen precision. A sixteen-term evaluation achieves a precision comparable to the single-precision machine epsilon on Sun-based workstations. Gaussian quadrature or other quadrature schemes may also be employed (cf. pp346-350 of Watt and Watt[10], and Guenter and Parent[11]), but the added economy for low-degree curves is minimal.

We shall now consider possible arc-length constraint formulations for curves of varying degrees. As we shall see, quartic curves are required to give the same level of control as can be achieved with, for example, cubic Bézier or Hermite curves, together with the added arc-length constraint. However, a we shall introduce a useful class of Hermite-like isometric cubic curves that may be sufficient for many applications.

2.1. Degree One Curve Segments

Line segments with two arbitrary endpoints P_0, P_1 cannot of course be arc-length constrained, but a family of line segments given by one endpoint P, a direction $\mathbf{d}$, and a length $\alpha > 0$ in fact allow us to specify

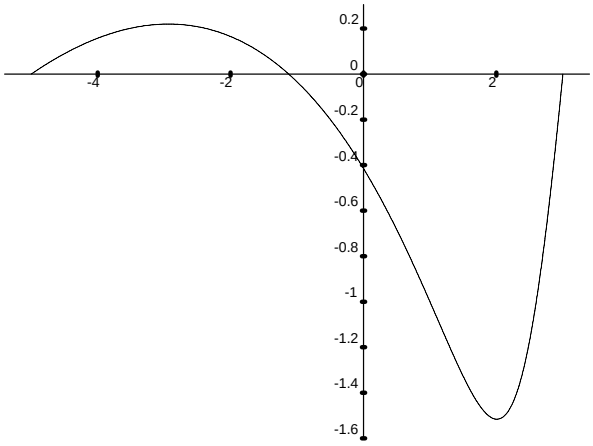

Figure 1: *A parametric cubic space curve.*

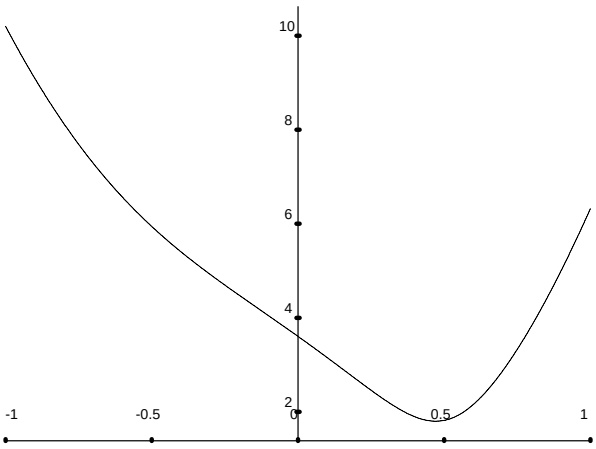

Figure 2: *The arc-length integrand for the above space curve.*

isometric line segments. Fixing α and P gives a circular locus of solutions for $\mathbf{d}$; fixing P and $\mathbf{d}$ gives a line of solutions for α; fixing $\mathbf{d}$ and α gives admits a plane of solutions for P. Finally, specifying each of P, $\mathbf{d}$ and α defines a unique line segment.

2.2. Degree Two Curve Segments

Traditional piecewise parabolic segments are often sufficient for applications needing only C^1 parametric continuity. However, with the addition of a segment-wise arc-length constraint it is not possible, for example, to interpolate two endpoints per parabolic segment with C^1 joint continuity, all the while maintaining an arc-length constraint. Nevertheless, this is

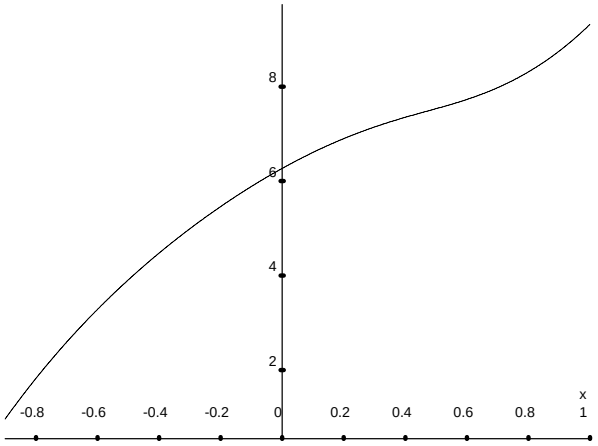

Figure 3: *The actual arc-length of the space curve, $A(\mathbf{p}(t); [-1, t])$, as t varies from -1 to 1.*

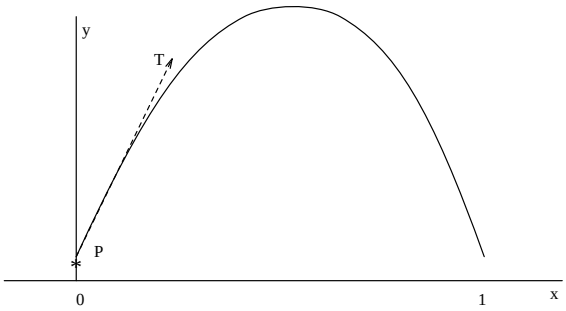

Figure 4: *Constraints on a parabolic curve segment.*

a nontrivial family of curves that will illustrate our derivation for higher-degree formulations discussed later.

2.2.1. A Single Parabolic Curve Segment

Suppose we require a parabola in explicit form

$$y = ax^2 + bx + c$$

to have arc length $\alpha > 0$ on the interval [0,1]. We require two more constraints to make this curve unique. From the point of view of curve design, a reasonable set of constraints would be that the curve must interpolate a y-value P at $x = 0$ and must have derivative $\dot{y}(0) = T$ at $x = 0$. This situation is depicted in Figure 4. Meeting the positional constraint requires that

$$c = P,$$

which is easily seen just by letting $x = 0$ in our expression for y. Similarly, differentiating y with respect to x and letting $x = 0$ shows that

$$b = T.$$

We can solve for a by meeting the arc-length constraint. If we require that

$$A(y; [0, 1]) = \alpha,$$

then

$$\begin{aligned} A(y; [0,1]) &= \int_0^1 \sqrt{1 + \dot{y}^2}\, dx \\ &= \int_0^1 \sqrt{1 + (2ax + b)^2}\, dx \\ &= \int_0^1 \sqrt{1 + (2ax + T)^2}\, dx. \end{aligned} \tag{4}$$

Note that the positional constraint does not figure in the above equation, which is sensible, since an arc length is independent of the curve's origin. The integral on the right-hand side has an analytic form:

$$\begin{aligned} A(y; [0,1]) &= \frac{2\,Aa + AT + \ln(4\,Aa + 8\,a^2 + 4\,aT)}{4\,a} \\ &- \frac{UT + \ln(4\,Ua + 4\,aT)}{4\,a}, \end{aligned} \tag{5}$$

where

$$\begin{aligned} A &= \sqrt{1 + 4aT + T^2 + 4a^2}, \text{ and} \\ U &= \sqrt{1 + T^2}. \end{aligned}$$

Since T is given, our goal is to solve for a subject to $A(y; [0, 1]) = \alpha$, for a given $\alpha > 0$. Note that the term U is constant, but A is not. Some trivial rearrangement of the symbols is possible, but the formula appears to resist further simplification. While a closed-form solution for a appears impossible, a solution, when it exists, can be efficiently found using numerical techniques, as is discussed below. Because the above equation is quadratic in a, there are usually two possible solutions. For example, Figure 5 depicts the solutions corresponding to

$$\alpha = 10, \quad T = 5, \quad P = 0.$$

Using numerical techniques described in Section 3, we find that the two quadratic curves that meet these constraints are (approximately)

$$\begin{aligned} p_1(t) &= -14t^2 + 5t, \\ p_2(t) &= 5t^2 + 5t. \end{aligned}$$

We can therefore choose the desired curve shape by choosing the solution that is concave up or down as needed, simply by selecting a positive or negative a.

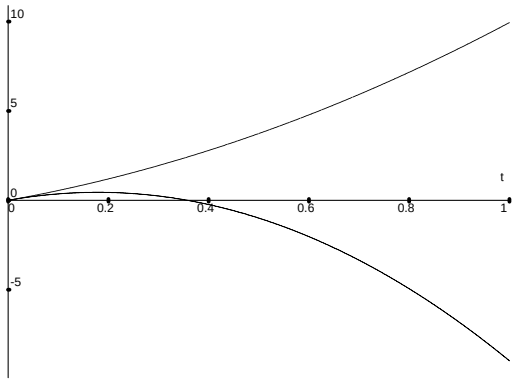

Figure 5: *Two parabolic curve segments with* $y(0) = 0,\ \dot{y}(0) = 5,\ \alpha = 10.$

An important point here is that the log in the equation is a *complex* natural log. In the above example, only the solution for positive a, namely p_2, is found with a real-valued log. Computing with a complex-valued log will find both solutions. Unfortunately, this means a direct implementation of Eq. 5 is likely to be slower than using quadrature on Eq. 4.

A solution will not exist when the desired arc length is not realisable for a given T. A lower bound for α given T may be obtained by considering the length of the line segment from an initial point P in the direction T travelling over unit distance in x. The length of this line segment is thus

$$\begin{aligned}\sqrt{\Delta x^2 + \Delta y^2} &= \sqrt{1 + T^2} \\ &= U.\end{aligned}$$

Rather than interpolating position and tangent, it is easy to formulate isometric quadratics that interpolate two positions P_0, P_1, at $x = 0, 1$, respectively, for example. Noting that $y(0) = P_0$ again implies that

$$c = P_0.$$

Moreover, requiring that $y(1) = P_1$ implies that

$$a + b + c = P_1 \Rightarrow b = P_1 - P_0 - a.$$

The integral equation once again has a closed (but even messier form), and solutions for a can be found numerically.

The generalisation of isometric parabolic curve segments from an explicit to parametric representation is straightforward. Suppose the space curve $\mathbf{p}(t) = (x(t), y(t))$ is to be constrained such that

$$\begin{aligned}\mathbf{p}(0) &= P = (p_x, p_y), \\ \dot{\mathbf{p}}(0) &= \mathbf{T} = (d_x, d_y),\end{aligned}$$

and suppose furthermore that the curve segment must satisfy an arc-length constraint. There are at least two plausible constraints. One constraint would be to treat the x and y components of $\mathbf{p}$ separately and impose independent arc-length constraints on them as follows:

$$\begin{aligned}A(x; [0,1]) &= \alpha_x, \\ A(y; [0,1]) &= \alpha_y.\end{aligned}$$

Another constraint would be to consider the overall arc length of $\mathbf{p}$:

$$A(\mathbf{p}; [0,1]) = \alpha.$$

Componentwise constraints are explicit functions in the parameter, and can therefore be solved from our earlier discussion. The second constraint, namely true arc length for a space curve, is both more interesting and more realistic, since a solution has some hope of being co-ordinate system independent. Not all solutions to the constraint are independent of co-ordinate system. Most are not. A scheme based on componentwise arc-length constraints is co-ordinate system dependent by definition. In the case of isometric curves, we cannot require complete affine invariance, because arc length is not invariant under shears and scalings. This does not mean there is no way to predict the change in the arc-length as a result of scaling. In fact, for uniform scaling this is easy. Our notion of co-ordinate system independence will be restricted to translations and rotations.

Let the curve segment $\mathbf{p}(t) = (x(t), y(t))$ be

$$\mathbf{p}(t) = \mathbf{a}t^2 + \mathbf{b}t + \mathbf{c},$$

denoting the components

$$\begin{aligned}x(t) &= a_x t^2 + b_x t + c_x, \\ y(t) &= a_y t^2 + b_y t + c_y.\end{aligned}$$

Most of the constraints can be resolved componentwise and are analogous to the solution in the explicit case:

$$\begin{aligned}\mathbf{p}(0) &= P \quad \Rightarrow (c_x, c_y) = P, \\ \dot{\mathbf{p}}(0) &= \mathbf{T} \quad \Rightarrow (b_x, b_y) = \mathbf{T}.\end{aligned}$$

It remains to solve for the leading coefficients (a_x, a_y). Recalling from Eq. 2 that the arc length for a space curve on [0,1] is

$$\begin{aligned}A(\mathbf{p}; [0,1]) &= \int_0^1 \|(\dot{\mathbf{p}}(t)\|\ dt \\ &= \int_0^1 \sqrt{\dot{x}(t)^2 + \dot{y}(t)^2}\ dt,\end{aligned}$$

after substituting the above constraints, our expression for arc length becomes

$$A(\mathbf{p};[0,1]) \;=\; \int_0^1 \sqrt{at^2+bt+c}\; dt, \qquad (6)$$

where $\mathbf{T} = (d_x, d_y)$, $a = 4\mathbf{a}\cdot\mathbf{a}$, $b = 4\mathbf{a}\cdot\mathbf{T}$, and $c = \mathbf{T}\cdot\mathbf{T}$. After some tedious algebra, a closed-form solution for the integral can be found (which is unsurprising, since a closed form exists for the explicit case). However, it is extremely messy and uninformative, and indeed it is more appropriate to solve numerically for (a_x, a_y) directly from the integral equation, since real square roots are less costly than complex natural logarithms.

Observe that while the positional and tangential constraints can be solved independently for each component, the arc-length constraint involves both components simultaneously. In fact, the system is underconstrained, since we have two unknowns, a_x and a_y, in one equation. There is thus a large set of possible solutions. One way to get a unique solution is to express a cross-constraint between a_x and a_y. For example, a_y could be a functional combination of the given values together with a_x. The simplest cross-constraint would be

$$a_y = \beta a_x,$$

for some constant $\beta \neq 0$. This allows us to write Eq. 6 as a function of a single variable a_x and, if a solution to the equation $A(\mathbf{p}(t;a_x);[t_0,t_1]) = \alpha$ for a_x exists, we can back-substitute to get a_y. (The notation $\mathbf{p}(t;a)$ means that the coefficient a is a parameter in the polynomial $\mathbf{p}(t)$. Thus a would be the indeterminate in the integral equation $A(\mathbf{p}(t;a);[t_0,t_1]) = \alpha$.) The advantage gained here is speed of computation, but a disadvantage is co-ordinate system dependence. In particular, our resulting curve will be translation invariant, but not rotation invariant. On the other hand, it has a rather nice property of being "bias" parameter. We shall more to say about this later.

2.2.2. Piecewise Formulations

It is not possible to define isometric everywhere C^1 quadratic curves that also interpolate two arbitrary prespecified endpoints. About the best we can do if C^1 joint continuity is desired is the "go with the flow" approach: specify an initial velocity and position for the first curve segment, and then use arc-length constraints to control the position and velocity of all subsequent segments. This is illustrated in Figure 6, in which the user is free to specify the initial position and tangent of first segment, as well as its arc length. The subsequent positional information for segment $i > 0$ is completely determined by α_{i-1}. This obviously does not result in a particularly powerful technique for curve design, but it may be of some use in trajectory simulation and in initial-value problems.

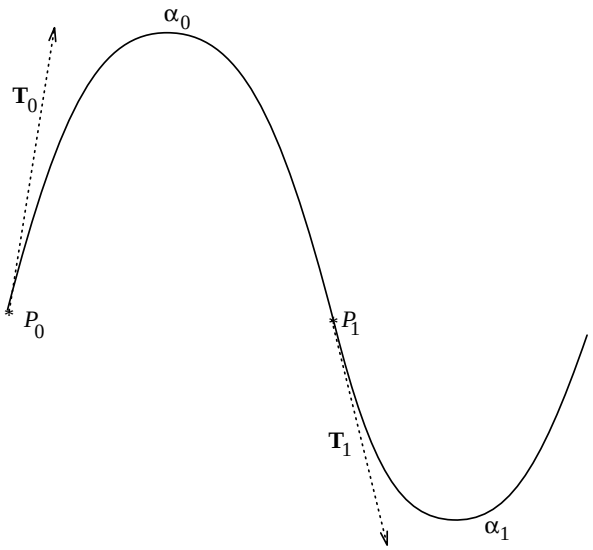

Figure 6: *Two parabolic curve segments with a user-specified initial tangent and position. Subsequent positions and tangents are controlled by the arc length for each segment.*

2.3. Degree Three Curves

2.3.1. A Single Cubic Curve Segment

The ability to meet four positional constraints exactly together with an arc-length constraint is not possible with a cubic curve segment. For example, we cannot hope to meet the cubic Hermite conditions of two endpoints P_0, P_1, two tangents $\mathbf{T}_0, \mathbf{T}_1$ with specific magnitudes, together with an arc-length constraint. However, by relaxing these constraints slightly, we can achieve a generally satisfactory solution. We shall first derive a solution for isometric cubic Hermite segments and subsequently apply this solution to the cubic Bézier form.

The well-known cubic Hermite curve is given by the constraints

$$\mathbf{p}(0) = P_0,\;\; \mathbf{p}(1) = P_1,\;\; \mathbf{T}_0 = \dot{\mathbf{p}}(0),\;\; \mathbf{T}_1 = \dot{\mathbf{p}}(1), \quad (7)$$

which leads easily to the monomial change-of-basis[12],

$$\mathbf{p}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ \mathbf{T}_0 \\ \mathbf{T}_1 \end{bmatrix}.$$

This gives rise to the following expression in the user-

specified constraints:

$$\begin{aligned}\mathbf{p}(t) &= \left(2\,t^3 - 3\,t^2 + 1\right) P_0 + \left(3\,t^2 - 2\,t^3\right) P_1 \\ &+ \left(t^3 - 2\,t^2 + t\right) \mathbf{T}_0 + \left(t^3 - t^2\right) \mathbf{T}_1 \\ &= (2(P_0 - P_1) + (\mathbf{T}_0 + \mathbf{T}_1))\,t^3 \\ &+ (3(P_1 - P_0) - 2\mathbf{T}_0 - \mathbf{T}_1))\,t^2 + \mathbf{T}_0 t + P_0.\end{aligned} \tag{8}$$

The resulting curve is unique for a given set of constraints. To meet an additional arc-length constraint, we relax the tangent constraint so that given user-specified tangents $\mathbf{T}_0, \mathbf{T}_1$, we find a Hermite curve that satisfies

$$\dot{\mathbf{p}}(0) = s\mathbf{T}_0, \quad \dot{\mathbf{p}}(1) = s\mathbf{T}_1.$$

We then solve for s subject to meeting the arc-length constraint. In this case, Eq. 8 becomes

$$\begin{aligned}\mathbf{p}(t) &= (2(P_0 - P_1) + s(\mathbf{T}_0 + \mathbf{T}_1))\,t^3 \\ &+ (3(P_1 - P_0) - s(2\mathbf{T}_0 - \mathbf{T}_1))\,t^2 + s\mathbf{T}_0 t + P_0.\end{aligned} \tag{9}$$

Observe that if we were able to derive a closed form for the arc length

$$A(\mathbf{p}(t;s);[0,1]) = \int_0^1 \sqrt{\dot{x}(t;s)^2 + \dot{y}(t;s)^2}\,dt, \tag{10}$$

then A would no longer be a function of t. Indeed, it would be a function of s alone. Furthermore, the integrand would contain terms in s^2 and s under the square-root (the actual formula is messy, but this is obvious from Eq. 10). Thus there would be at most two solutions to the nonlinear equation $A(s) = \alpha$, one with s positive, and the other with s negative. When a solution exists, we shall choose the solution $s > 0$; since s controls the magnitude of the curve's tangents at the endpoints of the interval, this will preserve the overall shape of the curve. Flipping a tangent would introduce or remove a point of inflection. Figure 7 illustrates how the curve changes to meet three different arc-length constraints while satisfying four constraints on position and tangent. Figure 8 illustrates the effect of changing the two tangents to the curve, while maintaining the same endpoints and arc length. For a wide-range of arc lengths, interacting with these constrained curves is quite similar to their unconstrained counterparts.

The classic cubic Bézier curve $\mathbf{p}(t)$ is given by four control points P_0, P_1, P_2, P_3. The constraints on $\mathbf{p}(t)$ are that

$$\begin{aligned}\mathbf{p}(0) &= P_0, & \mathbf{p}(1) &= P_3, \\ \dot{\mathbf{p}}(0) &= 3(P_1 - P_0), & \dot{\mathbf{p}}(1) &= 3(P_3 - P_2).\end{aligned} \tag{11}$$

This yields the following standard matrix representation:

$$\mathbf{p}(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -3 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}.$$

The cubic Hermite and Bézier curve families are directly related, and in fact the cubic Bézier form can be thought of as being a convenient specification technique for cubic Hermite curves. To solve for an arc-length constraint in a manner analogous to the Hermite case, we relax our tangent constraint to

$$\dot{\mathbf{p}}(0) = s(P_1 - P_0), \quad \dot{\mathbf{p}}(1) = s(P_3 - P_2).$$

Solving for s in this case is exactly as before, and we shall once again choose the solution for which

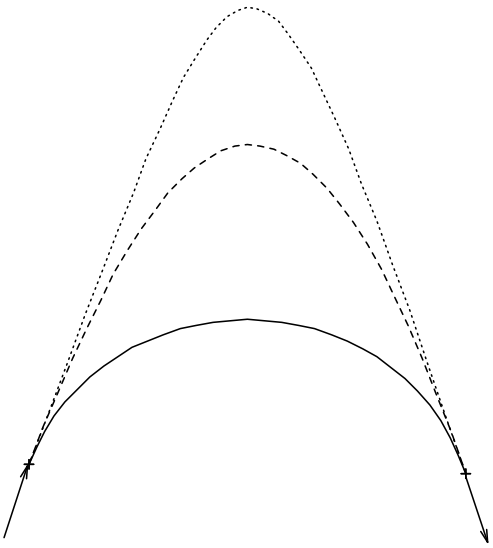

Figure 7: *Three Hermite cubic curves that each meet the same four constraints on positions and tangents, but which have varying arc lengths.*

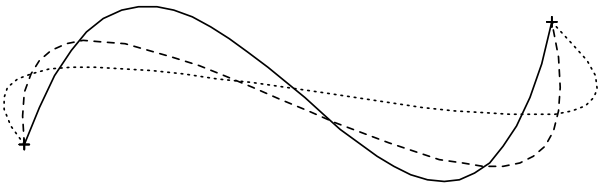

Figure 8: *The effect of varying the tangent of a constrained cubic hermite curve while maintaining the same arc length and endpoints.*

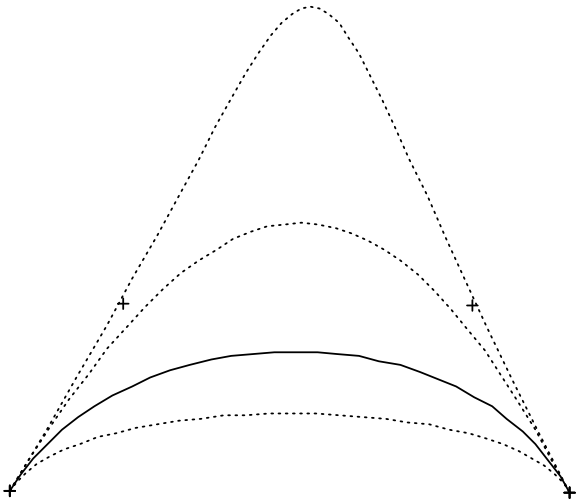

Figure 9: *Several constrained Bézier cubic curves that each meet the same four positional constraints, but which have differing arc lengths. The solid curve is the standard Bézier solution.*

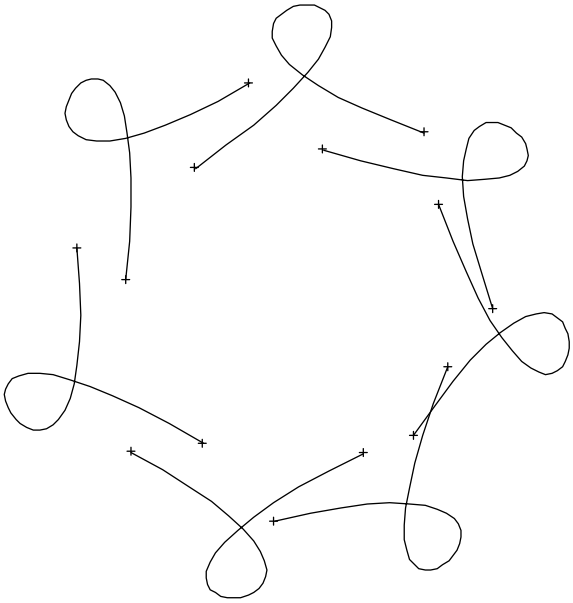

Figure 10: *Invariance of isometric cubic Bézier curves under rotation. For clarity, the two non-interpolated control vertices are not displayed.*

$s > 0$. Figure 9 illustrates the effect of changing the arc length of the Bézier curve while maintaining the other positional constraints. The solid curve depicts the standard cubic Bézier solution satisfying Eq. 11 (i.e., $s = 3$). Although these curves do not in general have the convex-hull property with respect to the original control points, they do have the property with respect to the control points $P_0, \hat{P}_1, \hat{P}_2, P_3$, where

$$\begin{aligned} \hat{P}_1 &= P_0 + s(P_1 - P_0), \\ \hat{P}_2 &= P_3 + s(P_3 - P_2). \end{aligned}$$

Overall, this curve family has many of the properties of the standard cubic Bézier form: it is invariant under translation and rotation (for example, notice the invariance of the curve under rotation in Figure 10), it has the convex hull property with respect to the updated control points given by the value s. Furthermore, as we shall see, s is cheap to compute.

2.3.2. Piecewise Formulation

Isometric cubic Bézier and Hermite curves can satisfy the expected constraints to ensure piecewise continuity. Because the magnitude of the tangents across curve segments cannot be preserved, we have a form of geometric, but not exact parametric continuity. In many cases, however, this is quite acceptable. The magnitude of the tangent vector is of course given by the s terms described above. In Figure 11, for example, we see three Bézier curves specified using eight control points in which we enforce G^1 continuity. Notice that when an interior (tangent) control point is moved, the solid curve becomes the dotted curve, preserving continuity. Furthermore, note the local control, evidenced by the fact that the final curve segment is invariant to a change in the tangent between the first two segments.

2.4. Degree Four Curve Segments

If our application requires that we meet position and velocity constraints exactly, as for example, in trajectory interpolation, then we must resort to quartic curves. Solving the system of constraints is analogous to the situation for parabolic segments. In particular, if we have a space curve

$$\mathbf{p}(t) = \mathbf{a}t^4 + \mathbf{b}t^3 + \mathbf{c}t^2 + \mathbf{d}t + \mathbf{e}$$

subject as in Eq. 7 to

$$\mathbf{p}(0) = P_0, \quad \mathbf{p}(1) = P_1, \quad \mathbf{T}_0 = \dot{\mathbf{p}}(0), \quad \mathbf{T}_1 = \dot{\mathbf{p}}(1),$$

then after some algebra,

$$\begin{aligned} \mathbf{e} &= P_0, \\ \mathbf{d} &= \mathbf{T}_0, \\ \mathbf{c} &= 3P_1 - 3P_0 - 2\mathbf{T}_0 - \mathbf{T}_1 + \mathbf{a} \\ \mathbf{b} &= 2P_0 - 2P_1 + \mathbf{T}_0 + \mathbf{T}_1 - 2\mathbf{a} \end{aligned} \tag{12}$$

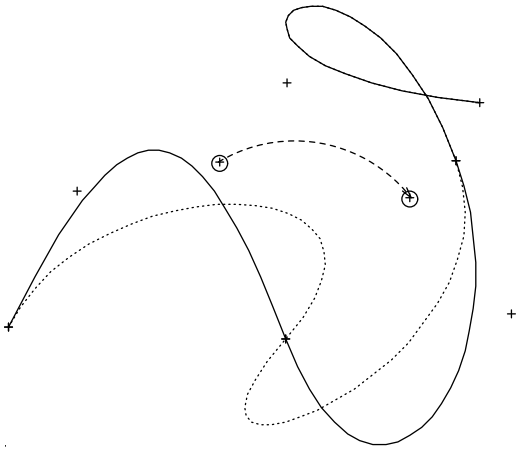

Figure 11: *Continuity and local control for isometric Bézier curves. The circled control point affecting the first two curve segments is moved, resulting in a change to these segments, while leaving the third invariant.*

which leaves $\mathbf{a} = (a_x, a_y)$ free. If we let $\mathbf{T}_0 = (\dot{x}_0, \dot{y}_0)$ and $\mathbf{T}_1 = (\dot{x}_1, \dot{y}_1)$, then each component $\dot{x}(t; a_x)^2$ and $\dot{y}(t; a_y)^2$ under the square root in an arc-length integral is of the form

$$\dot{x}(t; a_x)^2 = A_x a_x^2 + B_x a_x + C_x,$$

$$\dot{y}(t; a_y)^2 = A_y a_y^2 + B_y a_y + C_y,$$

where

$$\begin{aligned} A_y &= (4t^3 - 6t^2 + 2t)^2 \\ B_y &= (6(d_0 + d_1 + 2y_0 - 2y_1)t^2 \\ &\quad + 4(-d_1 - 2d_0 + 3y_1 - 3y_0)t + 2d_0)A_y) \\ C_y &= (3(2y_0 - 2y_1 + d_0 + d_1)t^2 \\ &\quad + 2(3y_1 - 3y_0 - 2d_0 - d_1)t + d_0)^2, \end{aligned}$$

and similarly for $\dot{x}(t; a_x)^2$. The important point here is that just as in the case for parabolic segments, the system is quadratic in $\mathbf{a}$. This is not a difficulty; indeed it allows us extra freedom in choosing an appropriate curve shape. We have found that most often, alternating positive and negative solutions values for $\mathbf{a}$ for each segment, meaning that the curve alternately goes from concave to convex, gives the most pleasing shape for trajectories. In geometric modelling applications, however, there may be special reasons for choosing convex curves[4].

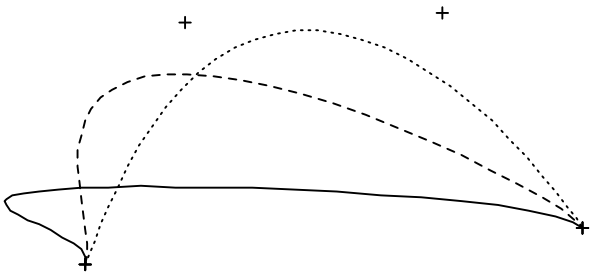

Figure 12: *The effect of β on an isometric quartic curve.*

As with the parabolic segments, we are faced with the problem of solving for the two $\mathbf{a}$ terms in one equation. If we impose the constraint that

$$a_y = \beta a_x, \tag{13}$$

then the solution for a_x can be quickly computed and back propagated. The choice of β affects the *bias* of the curve[1], as can be seen in Figure 12. For some applications, this bias term may be useful for design. However, because it directly biases the relative weight of the x or y components making up the space curve, the curve is not rotation invariant, though it is translation invariant.

Recalling Figure 12 once again, we note that, all other things being equal, the dotted curve is the most symmetric and pleasing. This happens to be the curve that minimises strain energy. For a scalar function $p(t)$, we define its *strain energy* over interval $[t_0, t_1]$ as

$$S(p(t); [t_0, t_1]) = \int_{t_0}^{t_1} \ddot{p}^2(t) dt. \tag{14}$$

The strain energy of a quartic polynomial $p(t) = at^4 + bt^3 + ct^2 + dt + e$ on [0,1] is

$$S(p(t); [0, 1]) = \frac{144}{5}a^2 + 12b^2 + 4c^2 + 16ac + 36ab + 12bc.$$

The strain of a space curve $\mathbf{p}(t) = (x(t), y(t))$ is simply

$$S(\mathbf{p}(t); [a, b]) = S(x(t); [a, b]) + S(y(t); [a, b]).$$

The second derivative is related to curvature, and indeed in an arc-length parameterisation it is equal to curvature. In general, it approximates curvature, and thus a polynomial from a class with the lowest strain energy essentially has the fewest "kinks" and wiggles in it. By using an affine-invariant measure such as

strain energy, we are able to remove a co-ordinate system bias from the solution for the leading coefficients **a**. This now gives us a curve formulation that is isometric and that exactly meets a set of cubic Hermite or Bézier style constraints. The price to pay is added computational difficulty, which we now address.

3. Numerical Techniques

Numerical implementations of the above formulations require careful study, but generally only elementary numerical techniques are required for acceptable realtime performance on cheap workstations. We first discuss stable fundamental techniques that were used in our implementation and then discuss approximation schemes when very fast computations are required.

As we saw earlier, an arc-length integral of a low-degree polynomial is a simple, smooth function. A 4-12 term Simpson's rule on a uniform subdivision of [0,1] quickly yields a satisfactory solution. Comparable results are achieved with gaussian quadrature[10, 11]. We shall see below how fast approximation schemes can be built from basic quadrature rules.

Solving the nonlinear equation

$$A(\mathbf{p}(t;s,a);[0,1]) = \alpha$$

for a scale factor s or for a single coefficient a requires some care. An ideal situation would have been one in which we would be solving for the bound variable t in the integral. Unfortunately, we are solving for a single variable within a square root and under an integral. The resulting derivative of A with respect to s or a as necessary is truly frightening and far too expensive to compute. This makes a derivative-free approach imperative. We have found that secant-rule works extremely well in this case[13]. Initial guesses are easy to manufacture heuristically, and the average convergence to achieve screen resolution is 3-4 iterations.

Solving for the isometric cubic and parabolic formulations requires nothing more than the above, and our entirely software implementation on a 0.5MFLOP Sun IPC without graphics assist permits the realtime manipulation of many curve segments. The computation is most certainly dominated by rendering time. As it happens, there is a surprising relationship between arc length and the leading coefficient a (and similarly for s). Figure 13 depicts how the arc length (vertical axis) varies with the leading coefficient a of a typical quartic curve. The picture for lower-degree curves is quite similar. Notice the nearly linear tails, tapering to a minimum. The trough of this curve clearly depends on the boundary conditions of the constrained polynomial, but in cases where the desired arc length is distant from the trough, a linear approximation is

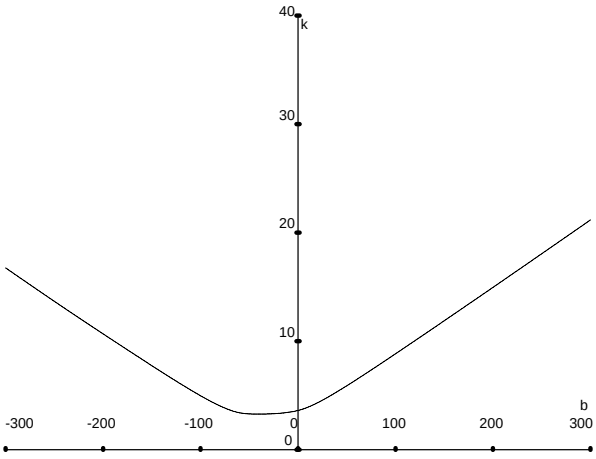

Figure 13: *The change in arc length (vertical axis) as the leading coefficient of a quartic varies.*

quite sufficient. This provides a very fast approximation when the length of a curve segment is changed while the control points remain the same.

Computing an isometric strain-minimisation functional is the most expensive step. To solve for the isometric polynomial that minimises strain we have observed that strain near the vicinity of the global minimum tends tends to have a rounded trough-like curve with respect to the leading coefficients. We have tried many approaches, but the best one seems to be successive parabolic interpolation, since it is derivative free, and since it quickly converges if the initial guesses are in the "valley" of the trough[14]. If this cannot be guaranteed, then a more conservative approach based on golden search can be employed[13].

Our strain-minimisation algorithm is as follows. Our goal is to find the quartic polynomial space curve $\mathbf{p}(t) = (x(t), y(t))$ with the least strain energy that meets the arc-length and positional constraints. As we saw in Eq. 13, the positional constraints prescribe the $\mathbf{b}, \mathbf{c}, \mathbf{d}$, and $\mathbf{e}$ coefficients of the space curve. Our only variables are the leading coefficients $\mathbf{a} = (a_x, a_y)$. If we use successive parabolic interpolation, we begin with three initial guesses for the leading coefficient a_x. Meeting the other constraints thus prescribes the corresponding a_y for these guesses. We evaluate the strain energy e_{a_x} of the resulting polynomials and fit a parabola through the points corresponding (a_x, e_{a_x}). We compute the minimum of this parabola, giving us a new a_x, and we discard the first guess. After an average of 4-5 iterations, we find a fixed point. Notice that this algorithm requires the both the arc-length integral and arc-length constraint satisfaction algorithms,

described above, as subroutines. As implemented, this implementation can compute the continuous update of about 10 such strain-minimal curves at about 10 frames/second on a Sun IPC.

When speed is at an absolute premium, we suggest the following scheme. For simplicity of notation, let us consider approximating the arc-length of a function $y = f(x;s)$, namely an explicit function in x with parameter s for which we ultimately wish to solve. When employing any nonadaptive compound quadrature rule on uniformly spaced samples, we approximate an arc-length integral on [0,1] as follows:

$$\begin{aligned} A(f;[0,1]) &= \int_0^1 \sqrt{1+\dot{f}(x;s)^2}\,dx \\ &\approx \sum_{i=0}^{n} w_i \sqrt{1+\dot{f}(i\Delta x;s)^2}\,dx, \quad (15) \end{aligned}$$

where the w_i are the weights given by the quadrature rule employed. As mentioned earlier, for a compound Simpson's rule, n is usually small (with $n = 8$ generally being quite sufficient). Thus evaluating Eq. 15 once is fast. The inefficiency arises because it is repeatedly evaluated to solve for a given arc length, and it is further repeated in strain-energy computations. It it would be worthwhile to approximate the compound quadrature rule by a low-degree polynomial, if possible.

In any of the above formulations, if we solve for either a leading coefficient or a scale factor, we always get a quadratic function in that parameter under the square root. That is,

$$A(f;[0,1]) \approx \sum_{i=0}^{n} w_i \sqrt{a_i s^2 + b_i s + c_i}, \quad (16)$$

where the a_i, b_i, and c_i are easily computed from the control points and $i\Delta x$. While techniques from computer algebra might sometimes be helpful in computing an analytic square root of the quadratic, we instead consider a series approximation. The Taylor's series for a single term within the summation is:

$$\begin{aligned} \sqrt{as^2+bs+c} \approx\ & \sqrt{c} + \frac{bs}{2\sqrt{c}} + \sqrt{c}\left(\frac{a}{2c} - \frac{b^2}{8c^2}\right)s^2 + \\ & \sqrt{c}\left(\frac{b^3}{16c^3} - \frac{ba}{4c^2}\right)s^3 + \quad (17) \\ & \sqrt{c}\left(\frac{3b^2a}{16c^3} - \frac{a^2}{8c^2} - \frac{5b^4}{128c^4}\right)s^4 + \\ & O(s^5). \end{aligned}$$

Thus each term involves the square-root of a quadratic function in the quadrature rule given by Eq. 16 requires the computation of a single square root, namely $\sqrt{c}$, in the approximation to the quadrature rule, Eq 17. If we sum the Taylor's series corresponding to each term in Eq. 16, we arrive at a polynomial approximation to the arc length. Letting the polynomial to be of low degree (e.g., 2-4) allows us to solve efficiently for arc-length directly as a function of s.

To summarise the technique, we approximate the arc-length integral for an arbitrary low-degree polynomial by writing it as a formal quadrature rule, performing a series approximation of the individual terms, and noting that when we sum the approximations, we get a low-degree polynomial in the parameter to solve, with only a small number of square roots to compute. The coefficients of the polynomial can be computed and directly substituted into the formulation. We stress that this solution is an approximation but the author has obtained good preliminary results. A more rigorous evaluation of the technique is necessary, however, and is the subject of current research.

4. Conclusion

We have presented an interesting class of Hermite-like isometric polynomial curves, and we have discussed a their implementation. Interacting with the isometric curves is as natural (or not) as working with the unconstrained Hermite form, and our formulation of these curves has local control and co-ordinate system independence.

One difficulty with the quartic form is that the strain-energy based objective function operates locally, in keeping with our theme of local control. This means that individual curve segments look quite good, but that the joints between curve segments may exhibit high local curvature. Several strategies may be employed to address this problem. One would be to optimise strain energy over more than one curve segment at a time. This would affect the local-control properties of the formulation. Another approach would be to formulate a different objective function to be minimised. After all, there is no requirement that a real trajectory must have minimum curvature at all times. A final approach would be to increase the degree of the curve and have it operate over wider support. Again, this undermines the local-support assumption.

As was mentioned earlier, we have only considered non-rational polynomial forms in this paper. Extensions to rational forms would be worthwhile.

Our formulation was chosen in the hope that it would make simpler the extension to isometric surfaces. We are just beginning that work now. Despite the apparent geometric simplicity of sheets of paper, patches of cloth, and pieces of sheet metal, surface-area constraints are mathematically much more subtle than arc-length constraints.

Acknowledgements

The author is indebted to Alain Fournier for first pointing him in the direction of the "scaled" Hermite form for isometric cubics. The anonymous referees have made valuable suggestions. The author also wishes to thank Brian Barsky for useful discussions on this topic while he was visiting the University of Toronto, and Peter Huang for discovering several typos in the mathematics.

References

1. R. Bartels, J. Beatty, and B. Barsky, *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*, Morgan Kaufmann, Los Altos CA, 1987.
2. G. Farin, *Curves and Surfaces for Computer-Aided Geometric Design*, Second Edition, Academic Press, Boston MA, 1990.
3. E. Jou and W. Han, "Minimal-energy splines with various end constraints", in *Curve and Surface Design*, SIAM 1992, H. Hagen (ed.), 23-40.
4. J.A. Roulier, "Specifying the arc length of Bézier curves", *Computer Aidied Geometric Design 10*, 1, (Jan. 1993), 25-56.
5. Chakib Bennis, J.-M. Vézien, and G. Iglésias, "Piecewise surface flattening for non-distorted texture mapping", *Proceedings of ACM SIGGRAPH 1991*, also published as *ACM Computer Graphics 25*, 4 (July 1991), 237-246.
6. E.T.Y. Lee, "Choosing nodes in parametric curve interpolation", *Computer Aided Design 21*, 6 (July/Aug. 1989), 363-370.
7. R.T. Farouki and T. Sakkalis, "Real rational curves are not 'unit speed' ", *Computer Aided Geometric Design 8* (1991), 151-157.
8. M. Girard, "Interactive design of 3D computer-animated legged animal motion", *IEEE Computer Graphics and Applications 7*, 6 (June, 1987), 39-51.
9. Kreyszig, E., *Differential Geometry*, Dover, New York; reprinted from University of Toronto Press, 1963.
10. A. Watt and M. Watt, *Advanced Animation and Rendering Techniques*, Addison-Wesley, Reading, Massachusetts, 1992.
11. B. Guenter, and R. Parent, "Computing the arc length of parametric curves", *IEEE Computer Graphics and Applications 10*, 3 (May, 1990), 72-78
12. J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes, *Computer Graphics–Principles and Practice*, Addison-Wesley, Reading, MA, 1990.
13. D. Kahaner, C. Moler and S. Nash, *Numerical Methods and Software*, Prentice-Hall, Englewood Cliffs, NJ, 1989.
14. R.P. Brent, *Algorithms for Minimization Without Derivatives*, Prentice-Hall, Englewood Cliffs, NJ, 1973.