Error Bounded Approximate Reparametrization of NURBS Curves

Mark Bloomenthal and Elaine Cohen

Department of Computer Science
University of Utah

Technical Report UUCS-00-005
January 24 2000

Abstract

This paper reports research on solutions to the following reparametrization problem: approximate $c(r(t))$ by a NURBS where c is a NURBS curve and r may, or may not, be a NURBS function. There are many practical applications of this problem including establishing and exploring correspondence in geometry, creating related speed profiles along motion curves for animation, specifying speeds along tool paths, and identifying geometrically equivalent, or nearly equivalent, curve mappings.

A framework for the approximation problem is described using two related algorithmic schemes. One constrains the shape of the approximation to be identical to the original curve c. The other relaxes this constraint. New algorithms for important cases of curve reparametrization are developed from within this framework. They produce results with bounded error and address approximate arc length parametrizations of curves, approximate inverses of NURBS functions, and reparametrizations that establish user specified tolerances as bounds on the Frechet distance between parametric curves.

1 Introduction

In designing free-form curves and surfaces, the user of a geometric modeling system generally prefers to concentrate on shape, not on the internals of curve and surface representations. Geometric constructions using parametrically defined representations, however, often lead to unwanted or surprising effects due to curve and surface parametrizations. Reparametrizing curve or surface representations is an important approach to minimizing these effects.

This paper focuses on new solutions to the following NURBS (Non-Uniform Rational B-Spline) reparametrization problem: approximate $c(r(t))$ by a NURBS where c is a NURBS curve and r may, or may not, be a NURBS function. Here c is a parametric, or vector valued, function (i.e. $c = c(u) : I_u \subset \boldsymbol{R} \rightarrow \boldsymbol{R}^n$) and r is a scalar valued change of parameter (i.e., $r(t) : I_t \subset \boldsymbol{R} \rightarrow I_u$). There are many practical applications of this problem within the context of NURBS based modeling systems. These include establishing and exploring correspondence in geometry, creating related speed profiles along motion curves for animation, specifying speeds along tool paths, and identifying geometrically equivalent, or nearly equivalent, curve mappings.

In these applications practical problems arise regarding the integration of reparametrization operations into the modeling system as usable design tools. Results must be predictable, take a usable form, and be capable of control by the user. Problems also arise in the representation of the results of such reparametrizations. These problems include:

Closure: It is important to represent the results of operations in NURBS form in a NURBS based modeling system so that operations in the system may be composed. However, reparametrization is not, in general, closed under the NURBS representation. This gives rise to the need for approximation in NURBS spaces.

Authors' current addresses: M. Bloomenthal, Think3, 2120 South, 1300 East, Suite 202, Salt Lake City, Utah 84106; E. Cohen, Department of Computer Science, University of Utah, 50 South Central Campus Drive, Room 3190, Salt Lake City, Utah 84112-9205.

Error Bounds: Users should be able to control the accuracy of approximations by specifying tolerances in meaningful metrics.

Data Complexity: Reparametrization involves (the approximation of) function composition. Many NURBS schemes to do this raise the degree of the polynomials used in the approximations. Increasing the accuracy of these approximations may also increase the number of polynomial pieces in the result. It is important to consider ways to manage this increased data complexity.

We describe a framework for NURBS curve reparametrization which addresses the above issues. New error bounded algorithms for important cases of NURBS curve reparametrization are developed from within this framework and include reparametrizations to approximate arc length parametrizations, approximate inverses of NURBS functions, and establish user specified tolerances as bounds on the Frechet distance between parametric curves.

2 Background

This section reviews some mathematical preliminaries used throughout this paper.

2.1 Distance Between Parametric Curve Maps

Given two mappings, $\alpha : I \to \boldsymbol{R}^n$ and $\beta : I \to \boldsymbol{R}^n$, the *equal-parameter distance* between these mappings is defined as:

$$d(\alpha, \beta) = \sup_t \|\alpha(t) - \beta(t)\|$$

for $t \in I$ and $\| \ \|$ the L^2 norm in $\boldsymbol{R}^n$.

Given two mappings, $\alpha : I_\alpha \to \boldsymbol{R}^n$ and $\beta : I_\beta \to \boldsymbol{R}^n$, the *Frechet distance* between these mappings is defined as:

$$\mathcal{F}(\alpha, \beta) = \inf_h \sup_t \|\alpha(t) - \beta(h(t))\|$$

where $t \in I_\alpha$ and $h : I_\alpha \to I_\beta$ is a homeomorphism.

The Frechet distance is essentially the minimum equal-parameter distance between α and possible reparametrizations of β. Whereas equal-parameter distance depends on the parametrizations of the curve mappings, Frechet distance is parametrization **independent**.

Frechet curves [14] are defined as equivalence classes of curve mappings under the relation: $\alpha \leftrightarrow \beta \equiv \mathcal{F}(\alpha, \beta) = 0$.

2.2 Piecewise Regular Curves

Throughout this paper we assume that all curve mappings are *piecewise regular* and that all change of parameter functions are *piecewise allowable*. Piecewise regular curves allow for intentional unit tangent and parametric derivative discontinuities, characteristics that are sometimes necessary in modeling environments.

A mapping, $\alpha(u) : I_u \to \boldsymbol{R}^n$, is *piecewise regular* if: $\alpha(u)$ is C^1 on I_u except at a finite number of points on I_u where it is constrained to be C^0, and $\frac{d\alpha(u)}{du} \neq 0 \ \forall u \in I_u$. (At points on I_u where $\alpha(u)$ is only C^0, left and right derivatives must exist and be nonzero.)

A real valued function $\theta(t) : I_t \to I_u$ is a *piecewise allowable change of parameter* if: $\theta(t)$ is C^1 on I_t except at a finite number of points on I_t where it is constrained to be C^0, and $\theta(t)$ is strictly monotonic on I_t.

2.3 NURBS Curves

NURBS curve and surface formulations have become fairly standard representation schemes in current CAD modeling systems. The NURBS representation has many attractive properties including computational stability, locality, and refinement. Below we highlight some important properties of NURBS that are used throughout this paper.

2.3.1 Some NURBS Closure Properties

If $\mathcal{N}$ is the class of all univariate NURBS functions defined on an interval I_t, and if $f(t), g(t) \in \mathcal{N}$ and $\lambda \in \boldsymbol{R}$ then:

$$f(t) + g(t) \in \mathcal{N}, \;\; \lambda f(t) \in \mathcal{N}, \;\; \frac{d}{dt} f(t) \in \mathcal{N}, \;\; \text{and } f(t)g(t) \in \mathcal{N}.$$

If $c(u) : I_u \rightarrow \boldsymbol{R}^n$ and $r(t) : I_t \rightarrow I_u$ are **both** NURBS then $c(r(t))$ is a NURBS.

NURBS representations for addition and multiplication of NURBS functions are discussed in [11] and [28]. Techniques for representing the composition of NURBS functions in NURBS form are discussed in [30] and [25].

2.3.2 Bounding Equal-Parameter Distance

In this section we use NURBS properties to outline methods for bounding the equal-parameter distance between NURBS curves. These techniques are used by the reparametrization algorithms of sections 4 and 5. See [5] for a more detailed discussion.

Given two NURBS curves with a common parametric domain $c_1(t), c_2(t) : I_t \rightarrow \boldsymbol{R}^n$ we seek to bound, from both above and below, $\|c_1(t) - c_2(t)\|$ over subintervals of I_t where $\| \; \|$ denotes the L^2 norm.

Without loss of generality assume that c_1 and c_2 are of the same polynomial order and defined over the same knot vector (this can always be achieved by using affine transformations of knot vectors, NURBS degree raising[8], and refinement[7]). The difference $v(t) = c_1(t) - c_2(t)$ can then be formed as a NURBS reducing the problem to bounding $\|v(t)\|$ for v a NURBS curve.

To solve this problem, the **scalar** valued function $\|v(t)\|^2 = < v(t), \; v(t) >$ can be formed as a NURBS and bounded from both above and below using convex hull analysis. A much more efficient method applies convex hull analysis directly to the **vector** valued function $v(t)$ (see [5]). NURBS curve refinement, applied prior to the convex hull analysis, can be used to obtain tighter error bounds.

3 Related Work

This section summarizes the previous work most directly related to the reparametrization algorithms presented in this paper.

3.1 Arc Length Parametrized Curves

Because of the importance of the problem there has been significant work related to approximating arc length functions for parametric curves and sampling parametric curves at equal spacings in arc length.

Farouki and Sakkalis show in [17] that it is impossible to parametrize any real curve, other than a piecewise straight line, by rational functions of its arc length. There are a number of methods, however, to extract arc length approximations for parametric curves at discrete points in the curve domain.

Fritsch and Nielson [18] form a piecewise linear approximation, $r(t)$, to the inverse arc length function generated from a specified number of evaluations equally spaced in the domain of a parametric curve $c(u)$. They then perform discrete evaluations of $c(r(t))$ (i.e., a curve form for $c(r(t))$ is **not** computed) to produce approximately equally spaced points on the curve. The authors use this technique in the context of the computation of a metric for curve/curve comparison. In [9] and [4] similar techniques are

used to approximate the arc length and inverse arc length functions for curves in the context of sweep surface construction.

Numerical integration techniques can be used to estimate arc lengths at discrete values in the curve domain. In [32] Newton-Raphson iteration in combination with Romberg quadrature is used to approximate discrete evaluation of the inverse arc length function for parametric curves. No error bounds are given, although the authors state that a relative error test is done when applying Romberg integration. In [21] adaptive Guassian quadrature is used to create a lookup table of approximate arc length function values over the domain of a parametric curve. Newton-Raphson iteration in combination with non-adaptive Guassian quadrature is then used to approximate discrete evaluation of the inverse arc length function for the curve. No true error bounds are given for their technique though the user may control relative error by the specification of a tolerance value used to control the adaptive quadrature.

Some approaches try to make the parametrization of interpolants closer to an arc length parametrization by enforcing unit derivative lengths at discrete points in the domain. In [35] a global G^2 parametric interpolation scheme is used which enforces unit derivative length at data interpolation points. Wang and Yang [34] present a technique to interpolate discrete data points using quintic splines. Their application is the conversion of parametric curves to a more nearly arc length form. No attempt is made to establish error bounds either in terms of arc length parametrization or deviation from the shape of the original sampled curve. More recently Srinivasan and Ge [33] have extended results in [34] to rational curves representing both translational and rotational motions.

In [22] discrete data points for rational spline motion are interpolated using a local cubic C^1 scheme. The context for this interpolation is real time control of industrial robots. A procedural reparametrization is applied to the interpolant to achieve a desired speed profile. This technique is concerned with real-time motion control and not with the actual expression of the reparametrized trajectory as a parametric curve.

Elber [12] presents error bounded algorithms to approximate both normalized univariate vector fields and arc length functions for NURBS curves. The results of both of these algorithms are bounded by a user specified tolerance on the variation of speed from unity. A NURBS approximation to inverse arc length is not computed and no algorithm for reparametrization by arc length is given. Instead root finding techniques are used for discrete point approximation of the inverse arc length function.

Blanc and Schlick [3] use quadratic rationals as change of parameter functions to improve the parametrization of NURBS curves. In particular they consider the use of these functions to improve the parametrization of quadratic NURBS representations of circular arcs. A heuristic is used to reduce the maximum value of the chordal deviation of circular arcs from constant speed. The reparametrization the authors use for circular arcs depends on the particulars of the behavior of the parametrization of the standard representation for these arcs. The authors do not give a generalization of their technique to approximate arc length parametrizations for more general curves.

The thrust in [16] is the reparametrization of polynomial curve segments to create nearly constant speed curves. The author adopts a "measure of closeness" to constant speed, and then analytically minimizes this measure subject to the use of a linear rational (Mobius) reparametrization. The author states that extending the specific minimization technique used to the reparametrization of rational curves is difficult.

Casciola and Morigi [6] create approximate arc length parametrized NURBS curves by using linear rational and C^1 piecewise linear rational change of parameter functions in an adaptive piecewise scheme. The optimization method they use to create an arc length approximation over a given parametric interval assumes detailed knowledge of error relative to the actual arc length function over that interval. The evaluation criteria require accurate knowledge of either the arc length function or the speed function on the curve. The authors do not state how the arc length or speed functions are approximated in these contexts and what properties these approximations have. True error bounds are not given for these algorithms.

3.2 Distance Between Parametric Curves

Section 5.3 presents an algorithm for bounding the Frechet distance between curves. Here we discuss previous work related to the computation of distance metrics between curves.

3.2.1 Frechet Distance

The Frechet distance between curves can be computed analytically for polygonal curves (i.e., piecewise linear). Alt and Godau [1, 20] give algorithms for determining the Frechet distance between polygonal curves assuming the two curves have linear parametrizations and unit length parametric intervals for each segment.

Elber in [12] observes that for $c_1(u)$ and $c_2(v)$ NURBS curves, the surface $\delta(u,v) = \|c_1(u) - c_2(v)\|^2 = < c_1(u) - c_2(v),\ c_1(u) - c_2(v) >$ can be represented as a NURBS. He further observes that establishing a value ϵ as a bound on Frechet distance between c_1 and c_2 is related to finding the zero set of $\delta(u,v) - \epsilon^2$. He also discusses using the surface $\delta(u,v)$ to find intersection, closest, and furthest points of the two curves.

The algorithm of section 5.3 uses closest point matches on two curves as a heuristic for finding a change of parameter function that demonstrates a bound on Frechet distance. Closest point matches have also been used in computer vision research. Zhang [36] gives an algorithm to match polygonal curves by using closest point matches on the curves. In [37] these techniques are extended to matching surfaces. Similar work has also been done in [2]. The primary application of these algorithms is an understanding of rigid body motion in computer vision. The authors do not apply these techniques to the reparametrization of curves or surfaces.

3.2.2 Other Metrics

The Frechet distance between curves is stated in terms of a min/max constraint. Its advantage is that it gives a measure of similarity between two curves over the entire path of both curves. Other metrics between parametrically defined curves are possible of course. Emery in [13] details an algorithm for the computation of the Hausdorff distance [29] between planar polygonal curves. This algorithm is applied to the creation of piecewise linear approximations to more general curves with bounded curvature.

Regular parametric curves are guaranteed to have arc length parametrizations. This notion can be used to form a metric between curves. In [18] the authors define a distance metric based on arc length: $d(a(u), b(v)) = \overset{\sup}{t} \|\alpha(t) - \beta(t)\|$ for α and β approximate normalized arc length parametrizations of a and b respectively. The arc length parametrization is approximated as referenced in section 3.1 above.

4 Framework

In this section we present a general framework for reparametrization algorithms using two algorithmic schemes, Methods 1 and 2. Sections 5.1 through 5.3 specialize this framework to specific reparametrization algorithms.

The general problem is to approximate $c(r(t))$ by a NURBS where c is a NURBS and r is assumed **not** to be a NURBS. (These methods also can be used when r **is** a NURBS and we want to represent r or $c(r)$ by NURBS functions of lower polynomial order.) The two schemes differ in the constraints applied to the approximation problem. Method 1 constrains the approximating NURBS function to have **exactly** the same shape as $c(u)$. Thus the approximation for $c(r(t))$ must be accomplished directly by NURBS function composition having the form $c(q(t))$ for some NURBS function q.

Method 2 relaxes this shape constraint, constraining the approximating function to be within some non-zero Frechet distance of $c(u)$. With the loosening of the shape constraint more general approximation techniques can be applied to the problem. The result is no longer exactly a function composition form but rather an *approximate function composition.*

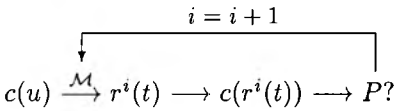


Figure 1: Schematic for Method 1. The iteration counter for the algorithm is denoted by i. $\mathcal{M}$ is a monotone approximation operator to scalar valued data. $P?$ checks for convergence to the correct parametrization.

4.1 Method 1

Figure 1 gives a schematic for Method 1. This method iteratively approximates $r(t)$ by a converging series of NURBS functions $\{r^i(t)\}$. At each iteration, the function composition $c(r^i(t))$ is formed as a NURBS. A test is performed to check convergence to the correct parametrization (signified by $P?$ in the diagram). When the convergence criterion is met the scheme terminates. Otherwise a more accurate NURBS approximation to r is generated and the scheme iterates.

For this scheme to succeed it is necessary to form a converging sequence of NURBS approximations $\{r^i\}$ and to test, with an appropriate metric, that the sequence $\{c(r^i)\}$ converges to the correct parametrization. These issues will be discussed in sections 5.1 through 5.3. Here we give a more detailed statement of the general method.

Denote by $\mathcal{S}^i = \mathcal{S}_{m,\tau^i}$ the ith approximating polynomial, or rational, spline space given by the ith knot vector τ^i and fixed order m. The ingredients for the general scheme are:

- A monotone approximation operator $\mathcal{M}[r, \mathcal{S}^i]$, which approximates r by a NURBS in space $\mathcal{S}^i$. Choices for this approximation operator are discussed in section 4.4.
- A metric, or pseudo metric, $d(a^i, f)$, which measures the distance between the ith approximation $a^i(t) = c(r^i(t))$ and $f(t) = c(r(t))$. It is important that this distance can be bounded. Bounds for $d(a^i(t), c(r(t)))$ are represented below by functions $b^i(t)$ whose range of values are assumed to be easily interrogated over intervals of the domain. Some general choices for d are discussed in section 4.6. Techniques for bounding d will be discussed in sections 5.1.1, 5.2.1, and 5.3.2.
- A refinement scheme **refine**$(\tau^i, b^i(t))$. Given error bound $b^i(t)$ and fixed order m this scheme will produce the knot vector τ^{i+1} which determines the approximating spline space for the next iteration of the algorithm. Some general characteristics of these refinement schemes are discussed in section 4.5.

The general scheme is as follows:

Input:

$c(u) : I_u \to \boldsymbol{R}^n$ a piecewise regular NURBS curve map,

$r(t) : I_t \to I_u$ a piecewise allowable change of parameter with $r(I_t) = I_u$,

m the order of approximating functions for r,

τ^0 the knot vector for the initial approximation of r, and

$\epsilon_1 > 0$ an error tolerance measure.

Output:

NURBS approximations for $f(t) = c(r(t))$ or $r(t)$.

Algorithm:

1. $i = 0$
2. $r^i(t) = \mathcal{M}[r, \mathcal{S}^i]$

3. form $a^i(t) = c(r^i(t))$ as a NURBS curve.
4. find error bound $b^i(t)$ such that $d(a^i(t), c(r(t))) \leq b^i(t)$.
5. if $\max_t b^i(t) > \epsilon_1$ then $\tau^{i+1} = \textbf{refine}(\tau^i, b^i(t))$, $i = i + 1$, go to 2.
6. return $r^i(t)$ or $a^i(t)$.

4.2 Method 2

This section develops an algorithmic approach which relaxes the constraint that the resulting curve have exactly the same shape as $c(u)$. The new scheme approximates $c(r(t))$ in spline spaces of arbitrary polynomial order. One of the primary advantages of this scheme, as compared with Method 1, is that it decouples the data complexity (order and number of control points) and smoothness of its result from the complexity and smoothness of the NURBS function composition form used. (See [5] for a more detailed comparison of the two methods.)

Figure 2 gives a schematic for Method 2. At each iteration, the NURBS function $a^i(t) \approx c(r(t))$ is

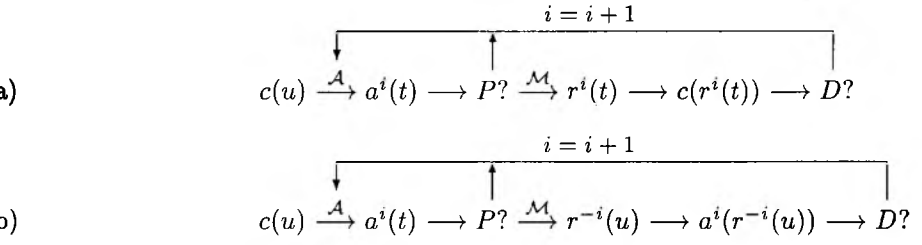


Figure 2: Schematic for Method 2. $\mathcal{A}$ is an approximation operator to vector valued data whereas $\mathcal{M}$ is a monotone approximation operator to scalar valued data. $P?$ checks for convergence to the correct parametrization. $D?$ checks equal-parameter distance between $c((r^i(t))$ and $a^i(t)$ in (a) and between $a^i(r^{-i}(u))$ and $c(u)$ in (b).

formed using an approximation operator $\mathcal{A}$. Test $P?$ is performed to measure convergence to the correct parametrization as in Method 1. Since operator $\mathcal{A}$ may change the shape of the approximation and since test $P?$ may not take shape into account, a further test, $D?$, is performed to make sure that $a^i(t)$ is within a user specified Frechet distance of $c(u)$. Figures 2(a) and (b) give two ways to accomplish this test; by using a convergent series of approximations to either r or to r^{-1}. (A slight abuse of notation appears here: r^{-i} denotes the ith approximation to the inverse function denoted as r^{-1}.)

For this scheme to work we must be able to construct two convergent sequences of NURBS functions, $\{a^i\}$ and either $\{r^{-i}\}$ or $\{r^i\}$, using distinct operators $\mathcal{A}$ and $\mathcal{M}$. This requirement appears to complicate Method 2 relative to Method 1. However, we will see that the two approximation problems of Method 2 are related very closely.

We now give a more detailed statement of Method 2 as it is depicted in Figure 2(b). Denote by $\mathcal{T}^i = \mathcal{T}_{m,\tau^i}$ the ith approximating polynomial, or rational, spline space given by ith knot vector τ^i and fixed order m. These spaces are used to formulate the spline approximations $a^i(t) \approx f(t) = c(r(t))$. Denote by $\mathcal{S}^i = \mathcal{S}_{l,\sigma^i}$ the ith approximating polynomial, or rational, spline space given by ith knot vector σ^i and fixed order l. These spaces are used to formulate the spline approximations $r^{-i}(u) \approx r^{-1}(u)$. The ingredients for this new scheme are:

- An approximation operator $\mathcal{A}[f, \mathcal{T}^i]$, which approximates **parametric** function f by a NURBS in space $\mathcal{T}^i$. Choices for this operator are discussed in section 4.4.2.
- A means of deriving $\mathcal{S}^i$ from space $\mathcal{T}^i$. Section 4.4.3 discusses the relationship between these spaces.
- A monotone approximation operator $\mathcal{M}[r^{-1}, \mathcal{S}^i]$, which approximates its first argument by a NURBS in the space of its second argument as in Method 1.

- A metric, or pseudo metric, $d(a^i, f)$, that measures the distance between the ith approximation $a^i(t)$ and $f(t) = c(r(t))$ as in Method 1.
- Refinement schemes **refine1**$(\tau^i, b^i(t))$ and **refine2**$(\tau^i, \beta^i(u))$. These schemes are discussed in section 4.5.

The new scheme is as follows:

Input:

$c(u) : I_u \to \boldsymbol{R}^n$ a piecewise regular NURBS curve map,

$r(t) : I_t \to I_u$ a piecewise allowable change of parameter with $r(I_t) = I_u$,

m the order of approximating functions for $c(r(t))$,

l the order of approximating functions for $r^{-1}(u)$,

τ^0 the knot vector for the initial approximation of $c(r(t))$, and

$\epsilon_1, \epsilon_2 > 0$ error tolerance measures.

Output:

NURBS approximation for $f(t) = c(r(t))$.

Algorithm:

1. $i = 0$
2. $a^i(t) = \mathcal{A}[f(t), \mathcal{T}^i]$
3. find error bound $b^i(t)$ such that $d(a^i(t), f(t)) \leq b^i(t)$.
4. if $\max_t b^i(t) > \epsilon_1$ then $\tau^{i+1} =$ **refine1**$(\tau^i, b^i(t))$, $i = i + 1$, go to 2.
5. derive $\mathcal{S}^i$ from $\mathcal{T}^i$.
6. $r^{-i}(u) = \mathcal{M}[r^{-1}, \mathcal{S}^i]$
7. form $q^i(u) = a^i(r^{-i}(u))$ as a NURBS curve.
8. find error bound $\beta^i(u)$ such that $\|q^i(u) - c(u)\| \leq \beta^i(u)$.
9. if $\max_u \beta^i(u) > \epsilon_2$ then $\tau^{i+1} =$ **refine2**$(\tau^i, \beta^i(u))$, $i = i + 1$, go to 2.
10. return $a^i(t)$.

4.3 Implementation Strategy

Method 1 must acquire data to approximate $r(t)$ iteratively. The data may include both function and derivative values. One way to drive this process is through acquisition of data at discrete locations in the domain of the original curve $c(u)$. This can be done when point-wise evaluation, or point-wise approximation, of $r^{-1}(u)$ is easier than evaluation, or approximation, of $r(t)$ itself. Examples are the arc length and inverse function algorithms of sections 5.1 and 5.2. This approach also can be used to avoid the use of numerical techniques for the inverse mapping of knot vector values when forming the NURBS composition form $c(r^i(t))$ (see [5]).

Tables can be created indexed by values in the domain of $c(u)$:

u domain		t domain
u^i_j	$\xrightarrow{r^{-1}}$	t^i_j, $\frac{d}{dt}r^i(t^i_j)$

Here j is the index into the table for iteration i, $t^i_j = r^{-i}(u^i_j)$ and $\frac{d}{dt}r^i(t^i_j) = 1/\frac{d}{du}r^{-i}(u^i_j)$. Since we assume piecewise allowable change of parameter functions r (without loss of generality, assume these

functions to be strictly increasing), tables sorted by increasing values of u will also be sorted by increasing values of t.

These tables then can be used by interpolation or approximation schemes to form NURBS approximation $r^i(t) \approx r(t)$ and subsequently NURBS curve $c(r^i(t)) \approx c(r(t))$.

Extended versions of these tables can be created as shown in Figure 3. Here $c(r^i(t^i_j)) = c(u^i_j)$ and $\frac{d}{dt}c(r^i(t^i_j)) = \frac{d}{du}c(u^i_j)/\frac{d}{du}r^{-i}(u^i_j)$. Method 2 can use the "$t$ domain" and "composed function" columns

composed function		u domain		t domain
$c(r^i(t^i_j)),\ \frac{d}{dt}c(r^i(t^i_j))$	$\xleftarrow{c}$	u^i_j	$\xrightarrow{r^{-1}}$	$t^i_j,\ \frac{d}{dt}r^i(t^i_j)$

Figure 3: Extended data table for use with Method 2.

to approximate $c(r(t))$ **directly**, and use the "t domain" and "u domain" columns to approximate r or r^{-1}.

4.4 Approximation Operators

This section discusses approximation operators $\mathcal{M}$ and $\mathcal{A}$ which are key ingredients of Methods 1 and 2. Operator $\mathcal{M}$ is used by both methods to approximate monotonic, scalar valued, change of parameter functions. Operator $\mathcal{A}$ is used by Method 2 to approximate parametric (vector valued) functions.

4.4.1 Operator $\mathcal{M}$

The monotone operator $\mathcal{M}$ creates an approximation to r (or its inverse) by a NURBS in a space $\mathcal{S}^i$. Generally this operator determines the approximation by fitting discrete data resulting from point-wise evaluation, or point-wise approximation, of r or r^{-1}. The data may include both function and derivative values. It is assumed the data reflect strictly monotonic functions.

Since we assume that all curves are piecewise regular and all change of parameter functions are piecewise allowable (see section 2.2), the data must be fit by a function in $\mathcal{S}^i$ that preserves monotonicity. Interpolation and approximation schemes that preserve convexity present in the data as well, may help to increase the rate of convergence of Methods 1 and 2. Simple choices for $\mathcal{M}$ include Schoenberg variation diminishing spline (SVDS) approximation and shape preserving interpolation.

SVDS approximation [27] requires only function values and generates a monotonic function given monotonic data. This choice for $\mathcal{M}$ allows an approximation to r (or r^{-1}) in any spline space $\mathcal{S}^i$, defined on a suitable parametric interval. The continuity of this scheme depends on the knot vector chosen for the space. For polynomial order n and singleton knots in the interior of the knot vector, this scheme is C^{n-2}. The major drawback of SVDS approximation is its slow convergence.

The simplest shape preserving **interpolant** is a piecewise linear function. This scheme trivially preserves both monotonicity and convexity in the data. Linear schemes also have the virtue of preserving the order of polynomial functions under composition.

Shape preserving quadratic and linear rational spline interpolation schemes are presented in [31] and [19] respectively. These schemes produce C^1 interpolants and require both function and derivative values. The C^1 nature of these schemes is not always appropriate if it is known a priori that the reparametrization function is **not** C^1 everywhere. The interpolation schemes can be augmented readily, however, to lower continuity at specified points in the interpolant's domain (see [5] and sections 5.1 through 5.3).

4.4.2 Operator $\mathcal{A}$

Operator $\mathcal{A}[f, \mathcal{T}^i]$ of Method 2 approximates functions $f = c(r(t))$ by NURBS curves in space $\mathcal{T}^i$. As with the monotone approximation operators discussed in section 4.4.1, this operator uses discrete data

resulting from point-wise evaluation, or point-wise approximation. Unlike the monotone operators of section 4.4.1 this operator is used to approximate **parametric** functions.

SVDS approximation, discussed in section 4.4.1 for use as a monotone approximation operator, can be used for approximation operator $\mathcal{A}$ as well.

The quasi-interpolant of deBoor and Fix [10] provides another choice for $\mathcal{A}$. It has optimal convergence properties but, for polynomial order n, requires function and derivative evaluation through order $n-1$. This scheme is useful, however, in cases where these derivatives are available. Other quasi-interpolant spline schemes trade more function evaluations for fewer, and lower order, derivative evaluations [26].

Approximation operator $\mathcal{A}$ could also employ techniques that interpolate discrete parametric data. C^1 piecewise cubic Hermite spline interpolation, requiring both function value and first derivative data, is an example of such a scheme. Many other interpolation schemes are possible. For introductions to techniques see [30] and [15].

Another choice for $\mathcal{A}$ is to use the NURBS function composition form for $c(r^i(t))$. Under this choice we see that Method 2 reduces to Method 1 (refer to Figure 2). Thus Method 2 is a true generalization of Method 1.

As in the case of the monotone operators of section 4.4.1, it is important to lower the continuity of C^n approximation operators $\mathcal{A}$ when it is known a priori that function $c(r(t))$ is **not** C^n everywhere.

4.4.3 Relationship Between $\mathcal{S}^i$ and $\mathcal{T}^i$

In Method 2 spline space $\mathcal{T}^i$ is used by operator $\mathcal{A}$ to approximate parametric functions $f = c(r(t))$, whereas spline space $\mathcal{S}^i$ is used by operator $\mathcal{M}$ to approximate scalar valued monotonic functions r or r^{-1}.

In theory there is little relationship between these spline spaces. In practice, however, these spaces are related by the data used for approximation and the particulars of the approximation schemes used for $\mathcal{M}$ and $\mathcal{A}$. Data from the same table can be used to approximate both r^{-1} (or r) and $c(r(t))$ as discussed in section 4.3. For operators $\mathcal{A}$ using interpolation at knot values, spline space $\mathcal{T}^i$ can be generated from the set $\{t^i_j\}_j$, along with particulars of operator $\mathcal{A}$, and consideration of known (function and derivative) discontinuities of $c(r(t))$. Similarly, for operators $\mathcal{M}$ using interpolation at knot values, spline space $\mathcal{S}^i$ can be generated from the set $\{u^i_j\}_j$ (or $\{t^i_j\}_j$), particulars of operator $\mathcal{M}$, and consideration of known discontinuities of $r(t)$.

4.5 Refinement Schemes

Refinement schemes for Methods 1 and 2 should be viewed in a general sense since τ^{i+1} may not be a simple refined partition of τ^i.

For algorithms that use interpolation or approximation to sample points for operators $\mathcal{M}$ and $\mathcal{A}$, it may be more natural to view the refinement schemes as generating a more refined or dense sampling of the function to be approximated. (See sections 5.1 through 5.3.) In cases where the change of parameter function is determined wholly, or in part, by c, the curve being reparametrized, this more refined sampling actually may be generated by embedding NURBS curve c in increasingly refined spline spaces. The new knot vector τ^{i+1} may then be determined by assignment of parameter values to these sample points, by consideration of known discontinuities that occur in $c(u)$ and $r(t)$, and by the exact nature of operators $\mathcal{M}$ and $\mathcal{A}$.

For Method 2 a very simple relationship exists between the two refinement schemes **refine1** and **refine2**. The first takes an error function, $b^i(t)$, in the domain of $c(r(t))$ whereas **refine2** takes an error function, $\beta^i(u)$, in the domain of $c(u)$. For **refine1** if $b(t)$ exceeds the specified tolerance on a data interval $[t^i_j, t^i_{j+1}]$ in the table of Figure 3, then the sampling in the next iteration should be increased in the interval $[u^i_j, u^i_{j+1}]$ (a new data point can be inserted at the midpoint of this interval for example).

For **refine2** if $\beta(u)$ exceeds the specified tolerance on a data interval $[u^i_j, u^i_{j+1}]$, then the sampling in the next iteration should be increased in this interval.

4.6 Metrics

The choice of metric, or pseudo metric, $d(f,g)$ for Methods 1 and 2 depends on the application domain. Some choices for d are given below. Their use in reparametrization algorithms is discussed in sections 5.1 through 5.3.

1. $\sup_t \|f(t) - g(t)\|$, equal-parameter distance between mappings.
2. $\mathcal{F}(f,g)$, Frechet distance.
3. $\sup_t \left| \|f'(t)\| - \|g'(t)\| \right|$, maximum difference of first derivative length functions.

5 Algorithms

In this section we specialize Methods 1 and 2 to perform specific reparametrizations. Section 5.1 discusses an algorithm to approximate arc length parametrizations of NURBS curves. Section 5.2 develops an algorithm to approximate inverses of scalar valued NURBS functions. Finally section 5.3 develops an algorithm for establishing user specified tolerances as bounds on the Frechet distance between NURBS curves.

5.1 Approximating Arc Length Parametrizations

Here Methods 1 and 2 are specialized to reparametrize a NURBS curve $c(u)$ by approximate arc length. The resulting approximate arc length parametrization will be C^1 if $c(u)$ is G^1 and have a variation in speed bounded by a user specified tolerance.

The arc length parametrization, $\gamma(t)$, is characterized by $\|\frac{d\gamma(t)}{dt}\| \equiv 1$; that is, an arc length parametrization has unit speed. This invariant will be exploited to measure convergence of the sequence of approximations produced by the algorithm.

Arc length parametrizations of NURBS curves can themselves be expressed as NURBS curves only for very restricted cases [17]. Hence, in general, a NURBS function can only **approximate** the arc length parametrization of a NURBS curve.

5.1.1 Bounding a Metric

Using pseudo metric 3 of section 4.6,

$$d(a^i(t), \gamma(t)) = \sup_t \left| \left\| \frac{d}{dt} a^i(t) \right\| - \left\| \frac{d}{dt} \gamma(t) \right\| \right| = \sup_t \left| \left\| \frac{d}{dt} a^i(t) \right\| - 1 \right|$$

for $a^i(t)$ the ith approximation to $\gamma(t) = c(r(t))$ the arc length parametrization for $c(u)$.

Since $a^i(t)$ is a NURBS curve, $\frac{d}{dt}a^i(t)$ can be represented as a NURBS curve. The techniques of section 2.3.2 can then be used to bound $\left\| \frac{d}{dt} a^i(t) \right\|$ from both above and below on intervals of the parametric domain. Hence $\left| \left\| \frac{d}{dt} a^i(t) \right\| - 1 \right|$ can be bounded over intervals of the domain to ensure that the speed of the approximations $a^i(t)$ converge uniformly to within a user specified tolerance of unit speed.

5.1.2 Approximating Inverse Arc Length

We discuss methods for approximating the inverse arc length function used as r in Methods 1 and 2. These techniques also can be used for approximating r^{-1} (the arc length function).

Function $r^i(t)$ of Methods 1 and 2 is computed from points $Q^i = \{(u_j^i, t_j^i)\}_j$, approximations of the arc length function for $c(u)$ at discrete points in the curve's domain. The points $P^i = \{(t_j^i, u_j^i)\}_j$ are then on an approximation to the inverse arc length function. There are many techniques for approximating the arc length function for NURBS (see [5]) including the use of inscribed polygons, partial sums of lengths on control polygons, and other quadrature methods.

To avoid oversampling, data should be acquired for the arc length approximation only where needed. If the error bound for $a^i(t)$ on an interval $I = [t_j^i, t_{j+1}^i] \subseteq I_t$ exceeds the specified tolerance then sampling for the arc length function in the next iteration is increased in an interval $\hat{I} = [u_j^i, u_{j+1}^i] \subseteq I_u$.

5.1.3 The Approximations $\mathbf{r^i(t)}$ and Continuity of $\mathbf{a^i(t)}$ in Method 1

There are a number of possible ways to approximate the inverse arc length function given sample points P^i. This section describes the use of the C^1 interpolants of [31] and [19]. Use of such interpolants allows Method 1 to construct C^1 approximations $a^i(t) = c(r^i(t))$ from G^1 curves $c(u)$.

Data supplied to these schemes must be in the form $\{(t_j, u_j, D_j)\}$ where the u_j represent function values, the D_j represent first derivative values, and the t_j are the locations in the parametric domain at which the given values are to be achieved. The $\{(t_j, u_j)\}$ correspond to the P^i above. The derivative values D_j can be computed from the speed function on the original curve $c(u)$ with $D_j = 1/\left\| \frac{dc(u)}{du}|_{u=u_j^i} \right\|$ in accordance with the inverse function theorem for derivatives. Note that because of the interpolation constraints, $\left\| \frac{dc(r^i(t))}{dt}|_{t=t_j^i} \right\| = \left\| \frac{dc(u)}{du}|_{u=u_j^i} \right\| \left\| \frac{dr^i(t)}{dt}|_{t=t_j^i} \right\| = \left\| \frac{dc(u)}{du}|_{u=u_j^i} \right\| / \left\| \frac{dc(u)}{du}|_{u=u_j^i} \right\| = 1$. Thus at each interpolation point t_j^i the speed of $c(r^i(t))$ in Method 1 is unity.

The interpolation schemes of [31] and [19] produce C^1 spline interpolants to the inverse arc length data. The arc length function, however, will be only C^0 at points where the speed function for $c(u)$ is discontinuous. Hence the inverse arc length function will be only C^0 at the associated points. The interpolation schemes can be augmented readily to constrain the resulting interpolants to be only C^0 at corresponding points in the interpolant's domain (see [5]). The augmented algorithm requires both left and right hand derivative information at such points. These data can be supplied from left and right speed values on $c(u)$. Again, because of the interpolation conditions, the left and right speed values for the composed function $c(r^i(t))$ will be 1.0 at these points. Thus if $c(u)$ is a G^1 curve, $c(r^i(t))$ of Method 1 will be C^1 for all i provided that each set Q^i includes all points in the domain of $c(u)$ where the curve is only C^0.

5.1.4 Example

Figure 4 shows a curve reparametrized using the approximate arc length algorithm. The change of parameter function used is only C^0 at a point corresponding to a speed discontinuity on the original G^1 curve. Figure 5 plots speed functions for this example. Figure 6 gives detailed results for executions of the algorithm for this example.

5.2 Approximating Inverse NURBS Functions

This section specializes Method 1 to an algorithm for approximating inverses of scalar valued NURBS functions. This is an approximation problem since for $c(u) : I_u \rightarrow \boldsymbol{R}$ a NURBS function, $c^{-1}(t)$ is usually **not** a NURBS.

The series of functions r^i of Method 1 become approximations to c^{-1}, and the invariant used to determine the convergence of the algorithm is that $a^i(t) = c(r^i(t))$ should converge to the identity

(a) (b) (c)

Figure 4: Reparametrization by arc length of a NURBS curve with a speed discontinuity: (a) original curve, (b) reparametrized curve. Dots represent equal spacing in the parametric domains. (c) Quadratic spline change of parameter function computed by Method 1.

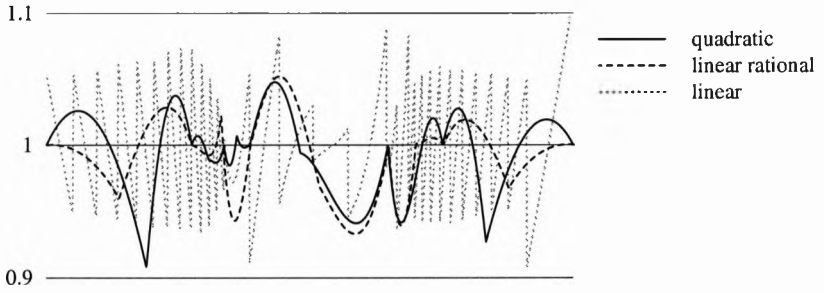


Figure 5: Graphs of speed functions for the example of Figure 4. Plots show errors for applications of Method 1 using piecewise linear, C^1 piecewise linear rational, and C^1 piecewise quadratic interpolation schemes. A tolerance of 0.1 was specified to the algorithm.

method	ϵ_1	ϵ_2	$\mathcal{M}$ or $\mathcal{A}$	order	size (# control pts)	time (secs)
1	.1	-	C^0 linear	4	91	0.0183
1	.1	-	C^1 linear rational	4	31	0.0842
1	.1	-	C^1 quadratic	7	63	0.0254
2	.1	.1	C^2 SVDS	4	28	0.1051
2	.1	.1	C^1 cubic Hermite	4	19	0.0270
2	.05	.05	C^2 SVDS	4	35	0.1349
2	.05	.05	C^1 cubic Hermite	4	23	0.0339

Figure 6: Results for arc length example of Figure 4. Order and size for reparametrized curve. Original curve order: 4, original curve size: 11. Method 1 lists scheme used for $\mathcal{M}$, Method 2 lists scheme used for $\mathcal{A}$.

function. The user specified tolerance can be given as an allowable distance of $a^i(t)$ from the identity, or, alternatively, an allowable distance of $r^i(t)$ from $c^{-1}(t)$.

Although we specialize Method 1 here, an extension of this algorithm can use the techniques of Method 2 (see section 6.1).

5.2.1 Bounding a Metric

Two metrics are proposed for measuring error for this algorithm. For $f(t) = c(c^{-1}(t)) = t$ the two metrics under consideration are:

$$d(a^i(t), f(t)) = \overset{\sup}{t} |c(r^i(t)) - t| = \overset{\sup}{t} |E(t)|, \tag{1}$$

and

$$d(r^i(t), c^{-1}(t)) = \sup_t |r^i(t) - c^{-1}(t)| = \sup_t |e(t)|. \tag{2}$$

Since c and r^i are NURBS, error function $E(t)$ can be represented as a NURBS and therefore can be bounded over intervals of its parametric domain.

Bounds on error function $e(t)$ can be related to bounds on $E(t)$ using the mean value theorem for derivatives. Assume $c(u)$ is a strictly monotonic increasing function. For a given interval $[u_a, u_b] \subset I_u$ let $t_a = c(u_a), t_b = c(u_b)$, and $\hat{I} = [u_a, u_b] \cup [r^i(t_a), r^i(t_b)]$. Let $c(u)$ be continuous on $\hat{I}$ and differentiable in the interior of $\hat{I}$. Then

$$|E(t)|/M \leq |e(t)| \leq |E(t)|/m \tag{3}$$

for $\forall t \in [t_a, t_b]$ with $m = \min_{u \in \hat{I}} \frac{d}{du} c(u)$ and $M = \max_{u \in \hat{I}} \frac{d}{du} c(u)$. Since c is a NURBS function, $\frac{d}{du} c(u)$ can be formed as a NURBS and its value bounded over intervals of the parametric domain.

5.2.2 The Approximations $\mathbf{r^i}$

The algorithm computes function r^i from sample points $Q^i = \{(u^i_j,\ t^i_j = c(u^i_j))\}_j$. The points $P^i = \{(t^i_j, u^i_j)\}_j$ are then on the function $c^{-1}(t)$. Bounds on the error function $E(t)$ (or $e(t)$) over intervals of the domain, can be used to control where new sample points are evaluated in each iteration in a manner similar to that given in section 5.1.2.

To use the C^1 shape preserving quadratic or linear rational approximation schemes of [31] and [19], derivative information is required at each sample point. This information can be acquired easily by application of the inverse function theorem for derivatives. For sample point (t^i_j, u^i_j) the derivative value for r^i is set to $1/\frac{d}{du} c(u^i_j)$. Again, as in section 5.1.3, the differentiability of the approximations r^i should be lowered to C^0 at those points corresponding to locations where $c(u)$ is only C^0.

5.3 Bounding Frechet Distance

This section specializes Method 1 to compute bounds on the distance between parametric mappings. This algorithm attempts to establish a user specified tolerance ϵ as an upper bound on the Frechet distance between two NURBS curves. The two curves, $c_1(t) : I_t = [t_s, t_e] \rightarrow \boldsymbol{R}^n$ and $c_2(u) : I_u = [u_s, u_e] \rightarrow \boldsymbol{R}^n$, are assumed to match at their end points; i.e., $\|c_1(t_s) - c_2(u_s)\| \leq \epsilon$ and $\|c_1(t_e) - c_2(u_e)\| \leq \epsilon$.

The algorithm reparametrizes c_2, using a sequence of piecewise allowable changes of parameter r^i, in an attempt to match c_1. To accomplish this, the algorithm uses the following heuristic: c_2 is reparametrized so that pairs of closest points on the two curves have the same parameter value. The rationale for this heuristic is that it holds for different parametrizations of the same Frechet curve and generalizes to many cases involving pairs of distinct Frechet curves.

Although we specialize Method 1 here, an extension of this algorithm can use the techniques of Method 2 (see section 6.2).

5.3.1 Closest Point Pairings

Our heuristic involves using pairs of closest points between two curves. Figure 7 demonstrates that such pairings are not necessarily unique and can give rise to invalid, nonmonotonic change of parameter functions.

One approach which solves these difficulties in many cases is to: a) perform the minimization operation exclusively over intervals on c_1 (i.e., curve c_1 is always searched for a closest point to an individual point on c_2) and b) use **local** solutions to the minimization problem (i.e., searches on c_1 are performed over restricted intervals of its domain). If $c_1(t_j)$ is already paired with $c_2(u_j)$ and $c_1(t_{j+1})$ is already paired with $c_2(u_{j+1})$, then in finding a match for a point $c_2(u)$ with $u \in (u_j, u_{j+1})$, the search on c_1 is restricted to the interval (t_j, t_{j+1}). See section 5.3.3 below for more details.

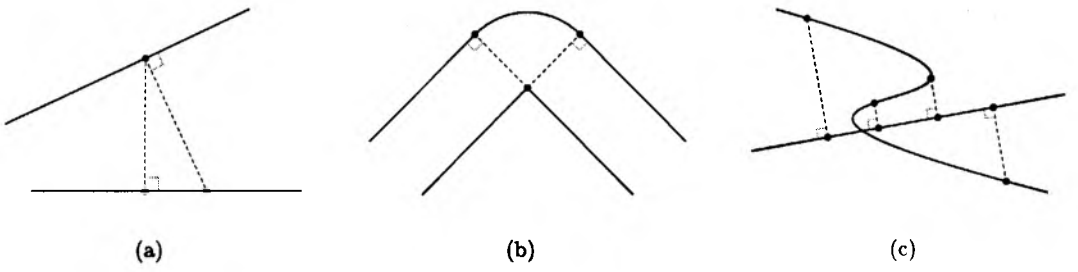


Figure 7: Closest point pairings. (a) Closest point relation is not always symmetric. (b) More than one solution to the minimum distance problem. (c) Point matches causing nonmonotonic parametric correspondence.

The algorithm makes use of the operation "find closest point on curve to point." See [24, 23] for recent work in this area.

5.3.2 Bounding a Metric

Using metric 1 of section 4.6

$$d(c_1(t), c_2(r^i(t)) = \overset{\sup}{t} \|c_1(t) - c_2(r^i(t))\| \geq \mathcal{F}(c_1(t), c_2(r^i(t)).$$

Establishing a bound for this choice of distance metric d is accomplished using the techniques of section 2.3.2.

5.3.3 Sample Points for r^i

The algorithm computes function r^i using sample points resulting from pairs of closest points on the two curves. For each iteration i of the algorithm, a list of sample points $Q^i = \{(t^i_j, u^i_j)\}_j$ is maintained, sorted by increasing values of t^i_j, where $t^i_j \in I_t \ \forall j$ and $u^i_j \in I_u \ \forall j$. These tuples have the property that for each j, $c_1(t^i_j)$ is that point on c_1 which is found to be closest to $c_2(u^i_j)$ using a restricted closest point search on c_1 (as described below). We now discuss how to create the sets Q^i from which $r^i : I_t \rightarrow I_u$ is computed.

For $I_t = [t_s, t_e]$ and $I_u = [u_s, u_e]$, set Q^0 to $\{(t_s, u_s), (t_e, u_e)\}$. For $i > 0$, data should be acquired for r^i only where most needed. For example suppose (t^i_j, u^i_j) and (t^i_{j+1}, u^i_{j+1}) are tuples in Q^i such that the error bound for $d(c_1(t), c_2(r^i(t))$ on $[t^i_j, t^i_{j+1}]$ exceeds the specified tolerance. Let $\hat{u} = (u^i_j + u^i_{j+1})/2$. In forming Q^{i+1} the tuple $(\hat{t}, \hat{u})$ is added to the set Q^i where $c_1(\hat{t})$ is the closest point on c_1 to $c_2(\hat{u})$, with the closest point search on c_1 restricted to (t^i_j, t^i_{j+1}). If the resulting $\hat{t}$ is within some predefined (small) distance of either end of interval $[t^i_j, t^i_{j+1}]$, move $\hat{t}$ slightly towards the interior of the interval to maintain monotonicity.

This algorithm has two stopping criteria. If $\|c_1(\hat{t}) - c_2(\hat{u})\| > \epsilon$, the algorithm terminates and reports a failure to establish ϵ as a bound on the Frechet distance in one of two subconditions. Curve c_1 is searched over its entire domain for the point closest to $c_2(\hat{u})$ and c_2 is searched over its entire domain for the point closest to $c_1(\hat{t})$. If either of these closest point distances are greater than ϵ then we know that $\mathcal{F}(c_1, c_2) > \epsilon$. Otherwise we can say only that the algorithm failed to establish ϵ as a bound. The algorithm terminates successfully if $d(c_1, c_2(r^i))$ is bounded by a value less than the user specified tolerance.

5.3.4 The Approximations r^i

To use the C^1 schemes described in [31] and [19], derivative information is required at each sample point in Q^i. Two approaches for deriving this information are given below.

Projection Approach: Assume c_1, c_2 piecewise regular, and $r : I_t \rightarrow I_u$ a piecewise allowable change of parameter with $\frac{d}{dt}r(t) > 0, \forall t$, i.e., a sense preserving change of parameter. If $c_1(t) = c_2(r(t)),\ \forall t \in I_t$ then $\frac{d}{dt}r(t) = \|\frac{d}{dt}c_1(t)\| / \|\frac{d}{du}c_2(u)\big|_{r(t)}\|$. This consideration leads to the following. Given the sample point (t_j^i, u_j^i) for some j, evaluate $V_1 = \frac{d}{dt}c_1(t_j^i)$ and $V_2 = \frac{d}{du}c_2(u_j^i)$. If the unit tangent directions for c_1 and c_2 are similar, approximate $\frac{d}{dt}r^i(t_j^i)$ by $\frac{\|V_1\|}{<V_1,V_2>/\|V_1\|} = \frac{<V_1,V_1>}{<V_1,V_2>}$.

It is possible, however, for two curves to have a small Frechet distance between them and yet have very dissimilar unit tangent directions. If the unit tangents $V_1/\|V_1\|$ and $V_2/\|V_2\|$ are dissimilar then a different method should be used for determining the derivative approximations.

Fitting Approach: One approach approximates derivative information based on the sample points Q^i alone. These data points can be fit locally with a polynomial or rational function whose derivative is then evaluated at discrete locations to serve as estimates for the $\frac{d}{dt}r^i(t_j^i)$. The use of a linear rational interpolation function for this purpose guarantees that derivative value estimates will be consistent with monotonic functions given monotonic data (see [19] and [5]).

5.3.5 Example

Figure 8 shows two NURBS curves with similar shape but very different parametrizations. The algorithm

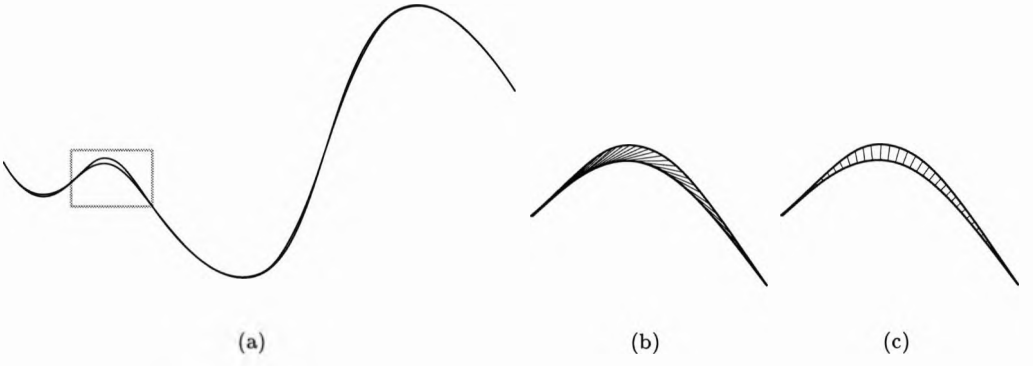


Figure 8: Example using the Frechet distance algorithm. (a) Two approximations to the same curve but with different parametrizations. Rectangle indicates area enlarged in (b) and (c). (b) Correspondence in the original parametrizations. (c) Correspondence after reparametrization to establish a user defined tolerance on Frechet distance.

of this section was used to bound the Frechet distance between these two curves. Figure 9 gives detailed results for running the algorithm on this example.

method	ϵ_1	$\mathcal{M}$	deriv estimation	order	size (# control pts)	time (secs)
1	.023	C^0 linear	-	2	21	0.0593
1	.023	C^1 linear rational	projection	2	41	0.2822
1	.023	C^1 quadratic	projection	3	42	0.0935
1	.023	C^1 linear rational	fitting	2	41	0.2832
1	.023	C^1 quadratic	fitting	3	42	0.0911

Figure 9: Results for Frechet distance example of Figure 8. Order and size for the change of parameter function used. The original curves of Figure 8 are order 4 and size 20.

6 Extensions

This section develops extensions to algorithms given in section 5. The extended algorithms illustrate further applications appropriate for Method 2. Section 6.1 extends the inverse function algorithm of section 5.2 to an algorithm which parametrizes curves along linear axes in space. Section 6.2 extends the Frechet distance algorithm of section 5.3 through consideration of a metric different from Euclidean distance. The resulting algorithm forms relative radial parametrizations of two curves.

6.1 Reparametrization by Axis

This section extends the algorithm of section 5.2 to parametrizations of NURBS curves along arbitrary linear axes in space. Using the affine invariance property of NURBS curves, it can be assumed, without loss of generality, that curve $c(u) = (x(u), y(u), z(u)) : I_u \rightarrow \boldsymbol{R}^3$ is to be parametrized along the X axis. It is further assumed that $x(u)$ is a strictly monotonic increasing function, although this algorithm can be generalized to piecewise monotonic coordinate functions.

The goal is to approximate the function

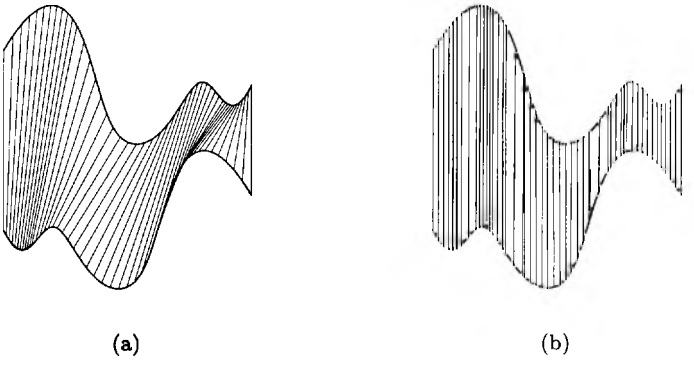
$$f(t) = (x(x^{-1}(t)),\ y(x^{-1}(t)),\ z(x^{-1}(t))) = (t,\ y(x^{-1}(t)),\ z(x^{-1}(t)))$$

by a NURBS curve given $c(u)$ a NURBS. This approximation can be formed as

$$a^i(t) = (x(r^i(t)),\ y(r^i(t)),\ z(r^i(t))).$$

The functions $r^i(t)$, approximations to $x^{-1}(t)$, are computed by the algorithm of section 5.2 using the metric of equation (1). This extension can use the techniques of Method 2 to compute NURBS approximations $a^i(t) \approx f(t)$ in spaces of arbitrary polynomial order.

Figure 10(a) shows the result of a ruled surface construction. The curves' parametrizations lead to



(a) (b)

Figure 10: Example using the reparametrization by X axis algorithm. (a) Ruled surface construction using curves with poor parametric correspondence. (b) Surface generated after reparametrization of the curves along the X axis.

poor correspondences which result in surface degeneracies. Figure 10(b) shows the ruled surface that results after the curves are reparametrized along the X axis. Figure 11 gives data from several runs of the algorithm for this example.

6.2 Radial Reparametrization

The Frechet distance algorithm of section 5.3 uses a metric and convergence criterion based on the Euclidean distance between points. Other metrics and convergence criteria could be employed instead.

method	ϵ_1	ϵ_2	$\mathcal{M}$ or $\mathcal{A}$	order	size (# control pts)	time (secs)
1	.01	-	C^0 linear	4	40	0.0059
1	.01	-	C^1 linear rational	4	37	0.0327
1	.01	-	C^1 quadratic	7	62	0.0092
2	.01	.01	C^2 SVDS	4	38	0.1280
2	.01	.01	C^1 cubic Hermite	4	22	0.0434
2	.005	.005	C^2 SVDS	4	55	0.2119
2	.005	.005	C^1 cubic Hermite	4	26	0.0460

Figure 11: Results for reparametrization by X axis for the bottom curve of Figure 10. Error metric for ϵ_1 given by equation (1). Extension of curves along the X axis is 2.15 units. Order and size for reparametrized curve. Original curve order: 4, original curve size: 13. Method 1 lists scheme used for $\mathcal{M}$, Method 2 lists scheme used for $\mathcal{A}$.

To demonstrate we modify the Frechet distance algorithm by using a distance metric and convergence criterion based on the *radial distance* between points. This results in an algorithm that forms relative **radial** reparametrizations of curves.

Figure 12 illustrates the situation under consideration. Assume that planar, piecewise regular curves

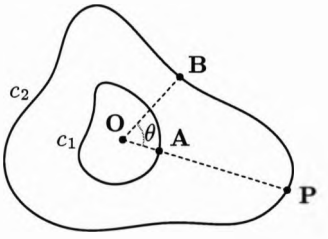


Figure 12: Radial distance and radial closest point operators between two curves.

$c_1(t) : I_t \rightarrow \boldsymbol{R}^2$ and $c_2(u) : I_u \rightarrow \boldsymbol{R}^2$ are "star-shaped" with respect to a common central point. Without loss of generality assume this point to be the origin **O**. Curve c_2 will be reparametrized to establish a *radial correspondence* with c_1. This correspondence is characterized by a change of parameter function, $r(t) : I_t \rightarrow I_u$, such that the line through $c_1(t)$ and $c_2(r(t))$ also goes through **O** for all t, and such that both curves move from the points $c_1(t)$ and $c_2(r(t))$ into the same half space as defined by this line.

Given point **A** on c_1 and point **B** on c_2, the "radial distance" between these points is given by the angle between vectors $\overrightarrow{\mathbf{OA}}$ and $\overrightarrow{\mathbf{OB}}$, which we denote by $\angle_{\mathbf{O}}(\mathbf{A}, \mathbf{B})$. Given point **A** on c_1, the radially closest point on c_2 is the point **P** at the intersection of c_2 with the half line starting at **O** and going through **A**. Thus the "find closest point on curve to point" operator of section 5.3.1 is a ray/curve intersection operation in this metric.

The algorithm of section 5.3 used the metric $\sup_t \|c_1(t) - c_2(r^i(t))\|$, for $\|\ \|$ the L^2 norm, to establish convergence and determine where more sample points for r^i, the approximation to r, should be generated. Here we use the pseudo metric

$$d(c_1(t), c_2(r^i(t))) = \sup_t \angle_{\mathbf{O}}(c_1(t), c_2(r^i(t))) \tag{4}$$

The user specifies a tolerance, θ, as the maximum allowable radial distance between $c_1(t)$ and $c_2(r^i(t))$. The pseudo metric of equation (4) can be bounded by:

$$\sup_t \left| 1 - \frac{< c_1(t), c_2(r^i(t)) >^2}{< c_1(t), c_1(t) >< c_2(r^i(t)), c_2(r^i(t)) >} \right| = \sup_t \sin^2\left(\angle_{\mathbf{O}}(c_1(t), c_2(r^i(t)))\right) \leq \sin^2(\theta). \tag{5}$$

For NURBS curves c_1, c_2, and r^i, the expression whose norm is taken in equation (5) is a NURBS function and hence its length can be bounded over intervals of its parametric domain (see section 2.3).

Substituting radial distance, the radial "find closest point on curve to point" operation, and the convergence criterion of equation (5) into the algorithm of section 5.3 yields an algorithm that produces a relative radial reparametrization of one curve to match another. This extension can use the techniques of Method 2 to compute reparametrizations in spaces of any desired polynomial order.

Figure 13 shows an example of the use of the radial reparametrization algorithm. Detailed results

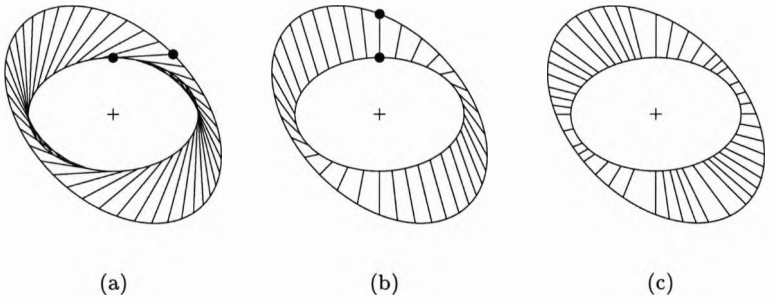


Figure 13: An example of the use of the radial reparametrization algorithm. Figure (a) shows the parametric correspondence between the original curves. Dots indicate the start/end points on the curves. Plus signs ("+") indicate the origin used for radial correspondence. Figure (b) shows the parametric correspondence between the curves once start/end points of the curves have been aligned radially. Figure (c) shows the results of running the radial reparametrization algorithm on the curves in (b).

for this example are given in Figure 14.

method	ϵ_1	$\mathcal{M}$	deriv estimation	order	size (# control pts)	time (secs)
1	1.0	C^0 linear	-	3	49	0.197
1	1.0	C^1 linear rational	projection	3	121	1.228
1	1.0	C^1 quadratic	projection	5	174	1.979
1	1.0	C^1 linear rational	fitting	3	69	0.556
1	1.0	C^1 quadratic	fitting	5	114	1.032

Figure 14: Results for radial reparametrization example of Figure 13(c). Error tolerance for ϵ_1 given in degrees. Order and size for the reparametrized curve. Original curve order: 3, original curve size: 14.

7 Conclusions

This paper has presented a framework for NURBS curve reparametrization. This framework takes the form of two algorithmic schemes with different constraints on resulting approximations. New algorithms for important cases of NURBS curve reparametrization have been developed as specializations of these schemes. Included are reparametrizations to approximate arc length parametrizations, approximate inverses of NURBS functions, and to establish bounds on the Frechet distance between curve mappings. These algorithms exhibit true error bounds in terms of meaningful metrics. Though important in their own right, the algorithms presented here are representative of a wider class of reparametrization algorithms in the methods they employ.

The core strategy afforded by the framework is to create adaptive algorithms that attempt to do work only where needed. This serves to restrain the growth in data complexity of the resulting approximations. This framework also provides the user with trade-offs in computing results. The user can trade accuracy for complexity of result and can place the resulting reparametrized curve into spaces of any desired polynomial order. The alternative general strategies provided by Methods 1 and 2 give the user further choices over properties of the result.

We believe that the framework given by Methods 1 and 2 is quite extensible. Its basic requirements are straightforward: a metric for measuring distance from a desired parametrization, a method for refining the space of approximation for the reparametrized curve (or change of parameter function), and a method for generating more data for the approximation. The substitution of different elements into this framework can lead to quite different algorithms (c.f. section 6.2).

Although the algorithms given here have been developed in the context of NURBS curves, it may be possible to use different curve representations. In this regard it is instructive to consider what properties of the NURBS representation the general framework uses. These properties include: closure under function composition, closure under algebraic operations, the ability to bound function and derivative values, and the existence of a refinement operation causing the coefficients of the representation to converge to the underlying function. Although these properties characterize the NURBS representation, they also guide the search for other representations useful for design and modeling.

Acknowledgments

This work was supported in part by the NSF Science and Technology Center for Computer Graphics and Scientific Visualization (ASC-89-20219) and DARPA (F33615-96-C-5621). All opinions, findings, conclusions or recommendations expressed in this document are those of the authors and do not necessarily reflect the views of the sponsoring agencies.

References

[1] Alt, H., and Godau, M. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications 5*, 1&2 (1995), 75–91.

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

[3] Blanc, C., and Schlick, C. Accurate parametrization of conics by NURBS. *IEEE Computer Graphics and Applications 16*, 6 (November 1996), 64–71.

[4] Bloomenthal, M. D. Approximation of sweep surfaces by tensor product B-splines. Tech. Rep. UUCS-88-008, Department of Computer Science, University of Utah, Salt Lake City, Utah, August 1988.

[5] Bloomenthal, M. D. Error bounded approximate reparametrization of non-uniform rational b-spline curves. Master's thesis, Department of Computer Science, University of Utah, Salt Lake City, Utah, August 1999.

[6] Casciola, G., and Morigi, S. Reparametrization of NURBS curves. *International Journal of Shape Modeling 2*, 2&3 (1996), 103–116.

[7] Cohen, E., Lyche, T., and Riesenfeld, R. F. Discrete B-splines and subdivision techniques in computer-aided geometric design and computer graphics. *Computer Graphics and Image Processing 14*, 2 (October 1980), 87–111.

[8] Cohen, E., Lyche, T., and Schumaker, L. L. Algorithms for degree-raising of splines. *ACM Transactions on Graphics 4*, 3 (July 1986), 171–181.

[9] Coquillart, S. A control-point-based sweeping technique. *IEEE Computer Graphics and Applications 7*, 11 (November 1987), 36–45.

[10] de Boor, C., and Fix, G. J. Spline approximation by quasi-interpolants. *Journal of Approximation Theory 8* (1973), 19–45.

[11] Elber, G. *Free Form Surface Analysis using a Hybrid of Symbolic and Numeric Computation.* PhD thesis, University of Utah, Salt Lake City, Utah, December 1992.

[12] Elber, G. Symbolic and numeric computation in curve interrogation. *Computer Graphics Forum 14*, 1 (March 1995), 25–34.

[13] EMERY, J. D. The definition and computation of a metric on plane curves. *Computer-Aided Design 18*, 1 (January/February 1986), 25–28.

[14] EWING, G. M. *Calculus of Variations with Applications.* Dover Publications, New York, 1985.

[15] FARIN, G. *Curves and Surfaces for Computer Aided Geometric Design – A Practical Guide.* Academic Press, San Diego, California, 1993.

[16] FAROUKI, R. T. Optimal parameterizations. *Computer Aided Geometric Design 14*, 2 (February 1997), 153–168.

[17] FAROUKI, R. T., AND SAKKALIS, T. Real rational curves are not 'unit speed'. *Computer Aided Geometric Design 8*, 2 (May 1991), 151–157.

[18] FRITSCH, F. N., AND NIELSON, G. M. On the problem of determining the distance between parametric curves. In *Curve and Surface Design*, H. Hagen, Ed. Siam, 1992, ch. 7, pp. 123–141.

[19] FUHR, R. D., AND KALLAY, M. Monotone linear rational spline interpolation. *Computer Aided Geometric Design 9*, 4 (September 1992), 313–319.

[20] GODAU, M. A natural metric for curves – computing the distance for polygonal chains and approximation algorithms. In *STACS 91. 8th Annual Symposium on Theoretical Aspects of Computer Science Proceedings* (Hamburg, Germany, 14-16 February 1991), C. Choffrut and M. Jantzen, Eds., Springer-Verlag; Berlin, Germany, pp. 127–136.

[21] GUENTER, B., AND PARENT, R. Computing the arc length of parametric curves. *IEEE Computer Graphics and Applications 10*, 3 (May 1990), 72–78.

[22] HORSCH, T., AND JÜTTLER, B. Cartesian spline interpolation for industrial robots. *Computer-Aided Design 30*, 3 (March 1998), 217–224.

[23] JOHNSON, D. E., AND COHEN, E. Minimum distance queries for polygonal and parametric models. Tech. Rep. UUCS-97-003, Department of Computer Science, University of Utah, Salt Lake City, Utah, February 1997.

[24] JOHNSON, D. E., AND COHEN, E. A framework for efficient minimum distance computations. In *Proceedings of the 1998 IEEE International Conference on Robotics & Automation* (Leuven, Belgium, May 16-21 1998), pp. 3678–3684.

[25] LEE, E. T. Y., AND LUCIAN, M. L. Möbius reparametrizations of rational B-splines. *Computer Aided Geometric Design 8*, 3 (August 1991), 213–215.

[26] LYCHE, T., AND SCHUMAKER, L. L. Local spline approximation methods. *Journal of Approximation Theory 15*, 4 (December 1975), 294–325.

[27] MARSDEN, M. J. An identity for spline functions with applications to variation-diminishing spline approximation. *Journal of Approximation Theory 3*, 1 (March 1970), 7–49.

[28] MØRKEN, K. Some identities for products and degree raising of splines. *Journal of Constructive Approximation 7*, 2 (1991), 195–208.

[29] MUNKRES, J. R. *Topology a First Course.* Prentice-Hall, Englewood Cliffs, New Jersey, 1975.

[30] PIEGL, L., AND TILLER, W. *The NURBS Book.* Springer-Verlag, Berlin, 1997.

[31] SCHUMAKER, L. L., AND STANLEY, S. S. Shape-preserving knot removal. *Computer Aided Geometric Design 13*, 9 (December 1996), 851–872.

[32] SHARPE, R. J., AND THORNE, R. W. Numerical method for extracting an arc length parameterization from parametric curves. *Computer-Aided Design 14*, 2 (March 1982), 79–81.

[33] SRINIVASAN, L. N., AND GE, Q. J. Fine tunning of rational B-splines motions – DETC97/DAC03984. In *Proceedings of DETC'97, 1997 ASME Design Engineering Technical Conferences* (Sacramento, California, September 14-17 1997).

[34] WANG, F.-C., AND YANG, D. C. H. Nearly arc-length parameterized quintic-spline interpolation for precision machining. *Computer-Aided Design 25*, 5 (May 1993), 281–288.

[35] WEVER, U. Optimal parameterization for cubic splines. *Computer-Aided Design 23*, 9 (November 1991), 641–644.

[36] ZHANG, Z. On local matching of free-form curves. In *BMVC92. Proceedings of the British Machine Vision Conference* (Leeds, UK, 22-24 September 1992), D. Hogg and R. Boyle, Eds., Springer-Verlag; Berlin, Germany, pp. 347–56.

[37] ZHANG, Z. Point matching for registration of free-form surfaces. In *Computer Analysis of Images and Patterns. 5th International Conference, CAIP '93 Proceedings* (Budapest, Hungary, 13-15 September 1993), D. Chetverikov and W. G. Kropatsch, Eds., Springer-Verlag; Berlin, Germany, pp. 460–467.