February 3, 1999
RT0302
Computer Science 10 pages

Research Report

A Shape-Preserving Data Embedding Algorithms for NURBS Curves and Surfaces

Ryutarou Ohbuchi, Hiroshi Masuda, and Masaki Aono

IBM Research, Tokyo Research Laboratory
IBM Japan, Ltd.
1623-14 Shimotsuruma, Yamato
Kanagawa 242-8502, Japan

Research Division
Almaden - Austin - Beijing - Haifa - India - T. J. Watson - Tokyo - Zurich

A Shape-Preserving Data Embedding Algorithm for NURBS Curves and Surfaces

Ryutarou Ohbuchi[1], Hiroshi Masuda[2], and Masaki Aono[1]
ohbuchi@acm.org, masuda@nakl.t.u-tokyo.ac.jp, aono@acm.org

[1] IBM Tokyo Research Laboratory
1623-14 Shimo-tsuruma, Yamato-shi, Kanagawa, 242-8502, Japan
[2] The University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan

ABSTRACT Three-dimensional (3D) (geometric) Computer Aided Design (CAD) has become increasingly popular in manufacturing industries. It is used to design, for example, automobile exterior, chassis, or engine blocks. A majority of these systems represent shapes by using parametric curves and surfaces, such as Bezier and NURBS (Non-uniform Rational B-Spline) curves and surfaces, as their main tools to define shapes.

Despite their popularity, to the author's knowledge, no data embedding algorithm designed specifically for CAD systems has been studied in the past. While algorithms did exist that embedded data into 3D polygonal meshes, most CAD systems employed curves and surfaces as their main shape-defining primitive. Even if polygonal meshes are used in a CAD, alterations of geometry and topology introduced by these data embedding algorithms could not be tolerated.

This paper proposes a new data embedding algorithm for NURBS curves and surfaces, which employed rational linear reparameterization for encoding messages. The most significant feature of the algorithms is exact preservation of geometric shape of its targets, that are, NURBS curves and NURBS tensor product surfaces. Furthermore, the algorithm preserves data size of the model.

The paper also suggests additional methods to embed data in parametric curves and surfaces, classified by shape- and size-preservation properties of the methods.

Keywords Computer aided design (CAD), parametric curves and surfaces, Non-Uniform Rational B-spline (NURBS), geometric modeling, information security, digital watermark.

I. INTRODUCTION

Data embedding, or *(digital) watermarking* put structures called *watermarks* into digital contents (e.g., images) in such a way that the structures do not interfere with intended use (e.g., viewing) of the contents. The watermarks carry information that can be used to manage the contents, in order, for example, to add annotations, to detect tampering, or to authenticate rightful purchasers. While data can be embedded in an analog media, digital media provided an opportunity for a robust data embedding with significant data capacity.

Previously, data embedding techniques for "traditional" digital multimedia content data types, such as text, image, video, and audio, have been the focus of study [Tanaka90, Walton95, Zhao95, Bender96, Braudway96, O'Ruanaidh96, Smith96, Cox97, Hartung97, Mintzer97, Yeung97a, Yeung97b]. Recently, 3D model gained status as an important member of multimedia data types, prompted by increasing popularity *of Virtual Reality Modeling Language* (VRML) [ISO97] and imminent standardization of *MPEG-4* [ISO98].

We have proposed several methods to embed data in the most important object type in a 3D model, that is, shapes of objects, in particular, shapes defined by using 3D polygonal meshes [Ohbuchi97, Ohbuchi98a, Ohbuchi98b]. In order to embed data into shape, our algorithms modify geometry (i.e., vertex coordinate), topology (connectivity of vertices) or both.

We tried to make the method reasonably robust against operations 3D models are routinely subjected to, e.g., an affine transformation or resection of a part of the model. For those algorithms that modify geometry, we employed affine transformation invariant quantities, such as a ratio of volumes of tetrahedrons created from a given triangular mesh. This made the watermark robust against affine transformation, a common manipulation for 3D models. Watermarks embedded by using topological modifications, for example, connectivity of vertices or connectivity of triangle strips, are not affected by geometrical transformation. In addition to algorithms that embeds information into shape, we have also proposed methods to embed data in attributes associated with each shape, such as per-vertex texture coordinates [Ohbuchi98b] and vertex color.

Several works have since been published on data embedding into shapes of 3D models. Kanai [Kanai98] developed a watermarking algorithm for 3D models that embeds data by using multiresolution wavelet decomposition of the models. The watermark withstands affine transformation, and is somewhat robust against random noise added onto vertex coordinates. An example of fragile watermarking (this term will be explained below) for 3D model is described in [Yeo99], which is a 3D version of the approach the authors have proposed previously for 2D image watermarking. Benedens [Benedens99] described a watermarking algorithm that employs a set of normal vectors derived from geometric shape of a 3D model for embedding.

In relation to 3D models, Hartung, et al [Hartung98] have developed a method to embed data in MPEG-4 facial animation parameters (FAP) sequences by using a spread-

spectrum technique. Remarkably, their watermarks could be extracted from rendered movie sequence of 2D images. To extract watermarks, they applied their facial feature tracking system being developed to generate FAP sequences from video sequences of real human faces.

For Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), or Computer Aided Engineering (CAE) applications, however, previous data embedding algorithms are not suitable most of the time. (In the following of this paper, we will use the acronym CAD loosely so that it implies all of the CAD, CAM, and CAE.) There are several reasons why these algorithms are not appropriate for CAD applications.

First, a significant proportion of CAD applications employ parametric curves and surfaces, such as *Bezier* or *Non-Uniform Rational B-Spline* (*NURBS*) curves and surfaces, not polygonal models, in order to define shapes. Models in these applications typically define a shape as a set of topologically connected *surface patches*, in which each patch is either a rectangular or triangular patch of (parametric) curved surface. In addition, curves may *trim* these patches so that arbitrary boundary shapes can be produced. Obviously, previous data embedding algorithms that targeted shapes defined by polygonal meshes can not be applied to these parametric curves and surfaces without significant modifications.

Second, even if a CAD system employed polygonal mesh to define shapes in a model, changes in the mesh's vertex coordinates due to data embedding are not acceptable. For example, a design of combustion engine cylinder deformed by data embedding will not be accepted. CAD applications demands data embedding technique that preserves exact shapes of models. An alternative, data embedding by using changes in topology, is also not acceptable to many, if not all, of CAD applications. Finite element analysis, for example, does rely on topology among model elements for computation.

In this paper, we propose data embedding methods for parametric curves and surfaces, the mainstay geometric elements of CAD applications. The contributions of this paper can be summarized as follows.

- Present an algorithm that embeds data in NURBS curves and surfaces by using *reparameterization*. The algorithm preserves exact geometric shapes of given curves and surfaces. In addition, data sizes of the curves and surfaces remain unchanged after data embedding.
- Outlines various alternative approaches to embed data in parametric curves and surfaces that are not limited to NURBS curves and surfaces. These approaches are classified by two properties, that are, preservation of exact shapes and preservation of model data sizes.

Being one of the most powerful yet practical methods to define curves and surfaces, NURBS are used in many CAD systems, for example, to define an automobile exterior shape. We expect there are many applications to the shape preserving data embedding algorithm for NURBS objects (i.e., NURBS curves and surfaces).

The following of this paper is organized as follows. In the remaining part of this section, we briefly review the concept of data embedding in general to develop terminology for later discussions. In Section II, we will describe an algorithm that embeds data in NURBS curves and surfaces while maintaining their exact shapes and data sizes. In Section III, we will discuss implementation and experimental results by using this algorithm. In Section IV, we will suggest possible alternatives to data embedding in parametric curves and surfaces. We conclude in Section V with summary and comments on future work.

A. Data Embedding

We introduce terminology on data embedding in this section. Following recommendations compiled in [Pfitzmann96], we call the act of adding watermark (data) *embedding* or *watermarking*, and retrieving the information encoded in the watermark for perusal *extraction*. The object in which the information is embedded is called *cover-*<*datatype*>, the object with watermark is called *stego-*<*datatype*>, and the information embedded is called *embedded-*<*datatype*>. The suffix "<datatype>" varies with data types, such as image, text, or 3D model. For example, an embedded-text is embedded in a cover-NURBS surface to result in a stego-NURBS surface with an embedded-text.

Traditionally, watermarks have been classified by their *visibility* (or, more generally, *perceptibility*) and *robustness*.

A *visible watermark* is made intentionally visible to serve their purposes, for example, to deter a third party from unauthorized sales of contents. On the other hand, an *invisible watermark* is imperceptible without processing by mechanical means.

A *robust watermark* should resists both intentional and unintentional modifications of the watermarked content. A *fragile watermark*, on the other hand, must be altered by intentional (and some unintentional) modifications so that it could detect tampering of or damage to the content (for example, [Yeo98]). Here, *unintentional modifications* are the kind a content should expect during a course of its intended use, while *intentional modifications* are the kind that are applied with an intention of modifying or destroying the watermark**.**

The classification above by perceptibility of watermark assumed human beings as observers of image, movie, text, or audio data. In case of watermarking of 3D models that are intended for viewing (that are, most VRML models), the observation is indirect since the model must be rendered before being viewed by human beings. Rendering method employed (e.g., wire frame vs. Gouraud shaded surface rendering) could greatly affect perceptibility of watermarks in models. In case of 3D models that are to be processed further by 3D CAD systems. Viewing is more indirect so that only a milled machine part (or whatever) produced by using the 3D CAD model may actually be visible.

A watermark can also be classified by its use of cover data for extraction. If an extraction algorithm requires original cover data as well as the (possibly corrupted) stego-data, the scheme is called *private watermarking*. Otherwise, the scheme is called *public watermarking*. An embedding scheme by Cox et al [Cox97] or Hartung [Hartung98] are examples of private

watermarking.

A watermarking scheme may employ a random sequence generator to make an embedded message secure from being read by a third party. For example, in an image watermarking, positions of pixels to be modified for watermarks can be scrambled by a pseudo-random sequence generated from a stego-key (or stego-keys) by using a public-key cryptographic method [9]. Scrambling of modulation values can also erase (reduce) statistical signature in order to make watermarking less detectable. At the same time, scrambling and spreading of modifications for data embedding could make the watermark more robust against interference, in a similar manner as spread-spectrum communication systems. Both public-key cryptography and shared-key (or private-key) cryptographic method can be used for this purpose.

Data embedding into CAD models defined by using parametric curves and surfaces has many potential applications. Requirements for a data embedding method vary depending on its intended application. A few of potential applications are;

- **Annotation**: Add annotations, such as date of the last modification and name of the person who modified it to a CAD model.
- **Fingerprinting:** A CAD model is "fingerprinted" with the identities (e.g., digital signatures) of its owner/designer and subcontractor by using a robust watermark. Circulation of unauthorized copies of the CAD model could be traced to the subcontractor.

Note that data embedding alone is not enough to realize these applications. For example, the last application requires an infrastructure, an organization to issue verifiable digital signatures and a trusted place to escrow original models (i.e., models without watermarks). Furthermore, digital signature and fingerprinting require appropriate laws as well as social acceptance.

II. Data Embedding in NURBS Curves and Surfaces by Using Reparameterization

A private data embedding algorithm for NURBS curves and surfaces that preserves exact geometric shape as well as data size of models is presented in this section. As a private data embedding scheme, the algorithm requires both the original data and watermarked data for extraction (Figure 1).

The fundamental idea behind this shape-preserving algorithm is that a NURBS curve or surface can be *reparameterized* without changing its geometric shape. In the following part of this section, following a definition of NURBS curve, the concept of reparameterization and data embedding method by using rational linear reparameterization is explained.

A shape-preserving data embedding algorithm is required by most CAD applications as discussed in the previous section. For example, if a model is deformed even slightly as a result of watermarking, a constructive solid geometry operation using the model will yield an erroneous result.

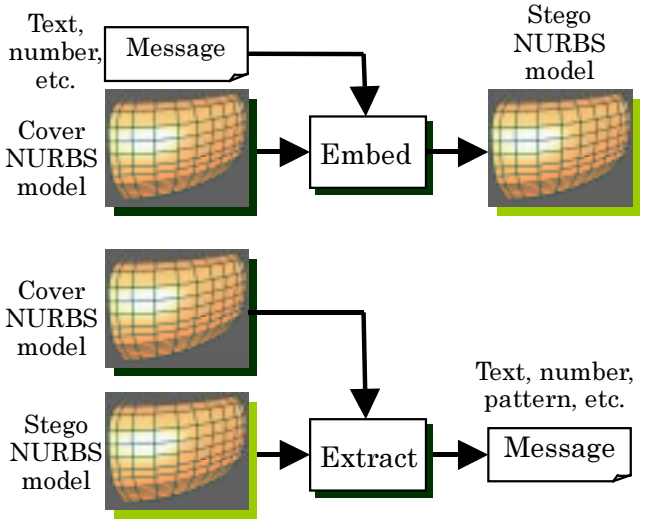


Figure.1 Flow of data in our data embedding algorithm for the NURBS curves and surfaces. It is a private data embedding scheme, which requires original cover-NURBS models for extracting embedded messages.

In addition to preserving shape, an algorithm that preserves data size is preferable most of the time, to contain communication and storage costs. In case of NURBS curves and surfaces, data size of models are determined mostly by the number of control points (including weights) and knots. Note that, in our paper, we consider a model data size is preserved if the numbers of control points and knots are unchanged. Their exact value, hence exact number of bits of a compressed model an after ideal entropy coding, may change.

In the following, we will explain the algorithm by using NURBS curve, but the same algorithm can be applied to NURBS surface created as a tensor product of two NURBS curves.

A. NURBS curve

This section defines a NURBS curve in order to explain data embedding algorithm. Detailed explanations of NURBS and other parametric curves and surfaces can be found in such books as Farin [Farin97] or Piegl and Tiller [Piegl97]. This paper follows notations of Piegl and Tiller.

A pth degree NURBS curve $\mathbf{C}(u)$ defines a point that traces a trajectory in 3D space as the scalar parameter value u vary within the range $[a,b]$.

$$\mathbf{C}(u)=\frac{\sum_{i=0}^{n} N_{i,p}(u)w_i\mathbf{P}_i}{\sum_{i=0}^{n} N_{i,p}(u)w_i} \quad u\in[a,b], \tag{1}$$

where a set of control points $\{\mathbf{P}_i\}$ forms a control polygon and $\{w_i\}$ are the weights. Increased weight w_i pulls the line closer to the control point $\mathbf{P}_i$. $N_{i,p}(u)$ is the ith B-spline basis function of degree p (order $p+1$), defined recursively as

$$N_{i,o}(u) = \begin{cases} 1 & \text{if } u_i \le u < u_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \quad (2)$$

Non periodic and non-uniform knot vector, a nondecreasing sequence of real numbers, is defined as

$$U = \{\underbrace{a,...,a}_{p+1}, u_{p+1},...,u_{m-p-1}, \underbrace{b,...,b}_{p+1}\} \quad (3)$$

Where $a \le u_i \le u_{i+1} \le b$ and $i = 0,...,m-1$. Knots in a NURBS curve are the points (in parameter space) where rational polynomial curves are grafted together to form a multi-segment curve.

B. REPARAMETERIZATION OF NURBS CURVE

A NURBS curve $\mathbf{C}(u) = \{x(u), y(u), z(u)\}$ defined on $u \in [a,b]$ is reparameterized by a function $u = f(s)$ so that the curve is computed as a function of a new parameter s instead of the original parameter u. We require the function $g(s)$ be increasing ($f'(x) > 0$ for all $s \in [c,d]$, in which $a = f(c)$ and $b = f(d)$) so that the same point x is not traced more than once. Detailed descriptions of reparameterization of NURBS curves and surfaces can be found, for example, in Piegl and Tiller [Piegl97].

Reparameterization of a parametric curve can be performed by using a large class of scalar function that satisfies the condition stated above, including polynomial, B-spline, or even trigonometric functions. Obviously, a NURBS curve reparameterized by using an arbitrary function is in general not a NURBS.

If a reparameterization function $u = f(s)$ is a polynomial of degree greater than one, the resulting function $\mathbf{C}(s)$ is a NURBS but it will have a raised degree, requiring larger numbers of knots and control points. Data size is not preserved if such reparameterization is employed for data embedding; description of $\mathbf{C}(s)$ will require more space to store than the original $\mathbf{C}(u)$ although both represents exactly the same geometric shape.

A NURBS curve reparameterized by using a polynomial or a rational polynomial of degree one is a NURBS curve with the same degree as the original. By manipulating the reparameterization function, e.g., slope of a linear reparameterization function, data can be encoded. In this paper, we chose rational-linear function to reparameterize a NURBS curve for data embedding. A rational linear function has higher degrees-of-freedom than a linear function, and thus allows us more control, for example, to better conform to given constraints or to embed more information. There are other possibilities than rational linear function. We will discuss one such possibility in Section II-F.

Reparameterization by using a rational-linear (alias linear fractional, bilinear, or Mobius) function has been studied by Lee and Lucian in [Lee91]. We summarize their result below. A rational-linear function $g(u)$ is defined as follows.

$$s = g(u) = \frac{\alpha u + \beta}{\gamma u + \delta} \quad (4)$$

$$u = f(s) = \frac{-\delta s + \beta}{\gamma s - \alpha} \quad s \in [c,d] \quad (5)$$

where $f(s)$ is the inverse of $g(u)$.

We let

$$\mu(u) = \gamma u + \delta \quad \lambda(s) = \gamma s - \alpha \quad (6)$$

To ensure that $g(u)$ and $f(s)$ are well-behaved, we assume

$$\alpha\delta - \gamma\beta > 0,$$
$$\mu(u) \ne 0 \quad \text{for all } u \in [a,b] \text{ , and} \quad (7)$$
$$\lambda(s) \ne 0 \quad \text{for all } s \in [c,d]$$

Then, the reparameterized curve $\mathbf{C}(s)$ is obtained as follows:

- The control points $\{\mathbf{P}_i\}$ remain the same.
- The new knots are the image under $g(u)$ of the original knots, $s_i = g(u_i)$.
- The new weights $\{\overline{w}_i\}$ (modulo a common nonzero factor) are obtained by either (8) or (9);

$$\overline{w}_i = w_i \prod_{j=1}^{p} \lambda(s_{i+j}) \quad (8)$$

$$\overline{w}_i = \frac{w_i}{\prod_{j=1}^{p} \mu(u_{i+j})} \quad (9)$$

s_{i+j} and u_{i+j} are the new and old knots, respectively.

C. A NURBS curve as an embedding primitive

In this paper, we call a minimum unit of modification for data embedding an *embedding primitive*. A rational linear reparameterization can be applied to a NURBS curve to use it as an embedding primitive.

Data can be embedded in the rational linear reparameterization function $g(u)$ by manipulating coefficients α, β, γ, and δ. Note, however, that the degrees-of-freedom of $g(u)$ is actually three, not four, which is obvious by rewriting the function as below.

$$s = g(u) = \frac{\alpha u + \beta}{\gamma u + \delta} = \frac{\frac{\alpha}{\gamma} u + \frac{\beta}{\gamma}}{u + \frac{\delta}{\gamma}} = \frac{k_1 u + k_2}{u + k_3}, \quad (10)$$

where $k_1 = \alpha/\gamma$, $k_2 = \beta/\gamma$ and $k_3 = \delta/\gamma$.

We find coefficients k_1, k_2, and k_3 that encode data, e.g., numbers. Coefficients k_1, k_2, and k_3 are computed by specifying three points (u_1, s_1), (u_2, s_2), (u_3, s_3) the function $g(u)$ must pass through (Figure 2). Substituting the three points in (4) and solving for k_1, k_2, and k_3 yields the following formula.

$$k_1 = \frac{(u_1 s_1 - u_2 s_2)(s_1 - s_3) - (u_1 s_1 - u_3 s_3)(s_1 - s_2)}{(u_1 - u_2)(s_1 - s_3) - (u_1 - u_3)(s_1 - s_2)}$$

$$k_2 = u_1 s_1 + k_3 s_1 - k_1 u_1$$

$$k_3 = \frac{(u_1 s_1 - u_3 s_3)(u_1 - u_2) - (u_1 s_1 - u_2 s_2)(u_1 - u_3)}{(u_1 - u_3)(s_1 - s_2) - (u_1 - u_2)(s_1 - s_3)} \quad (11)$$

While all three degrees of freedom may be manipulated to encode data, here, we constrain the two endpoints so that $u_1 = s_1$ and $u_3 = s_3$. This way, the range of parameters remains unchanged after the reparameterization. Such constraint is often necessary since certain CAD systems require that the parameterization be on a specified interval, e.g., [0, 1]. We encode data in a remaining degree-of-freedom, by setting the offset $D = s_2 - u_2$. The larger the magnitude of D, the more the curve of the function $s = g(u)$ deviates from the straight line $s = u$.

We have to pay attention to the amount of this offset for several reasons. First, with a large value of D, changes in the values of knots and weights can be quite large so that existence of data embedding can be noticed easily. Second, there is a notion of so-called "good parameterization". If parameterization of a curve is "good", a set of points on a curve is geometrically evenly spaced given a set of uniformly spaced parameter values. A good parameterization may be preferred, for example, if the curve (surface) is to be tessellated into straight line-segments (triangles) for display or numerically controlled milling. With a "bad" parameterization, unless an adaptive tessellation algorithm is used, tessellated line segments (triangles) will be uneven in size. Such uneven tessellation could lead to loss of accuracy of a milled machine part, for example. If the offset D is small, parameterizations of a curve before and after reparameterization should be similar. That is, a curve with "good" parameterization will also have a good parameterization after data embedding if the offset D is small.

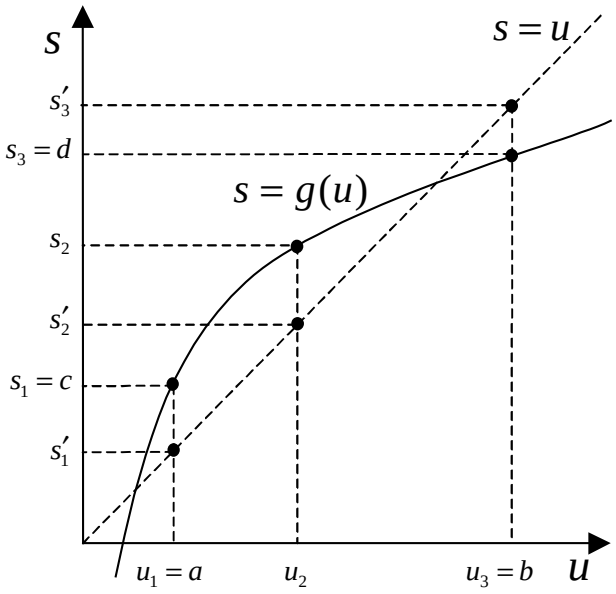


Figure 2. Determining a rational-linear function for reparameterization, which has three-degrees-of-freedom.

In our proof of concept implementation, we employed a simple method to encode a number into the offset D. The algorithm modified L lower (but not the lowest) order bit of the mantissa of the offset D to embed L bit of information.

Extraction starts by comparing the value of knots between a cover-model (original model) and a stego-model (watermarked model). At a predetermined element of the knot vector, the difference $D = s_2 - u_2$ is computed, and m bits at a known position in its mantissa is extracted. In our implementation, we used the knot in the middle of the sequence whose index $i = \lfloor m/2 \rfloor$ to compute the offset.

It is preferable to randomly scramble a message as it is embedded, so that the embedded bits won't show telltale pattern in the stego-data. In addition, all the curves and surfaces in the model should be reparameterized by using a random sequence so that curves and surface patches modified for embedding can not be distinguished easily from those without embedding. We have not included these two features in our proof-of-concept implementation.

D. A TENSOR-PRODUCT NURBS SURFACE AS AN EMBEDDING PRIMITIVE

Up to this point, the embedding algorithm is explained by assuming NURBS curves as its embedding primitive. It is trivial to extend the method to include a NURBS surface created as tensor products of two NURBS curves as another kind of embedding primitive.

A NURBS surface is defined as a tensor product of two NURBS curves with parameter u and v. A NURBS surface $\mathbf{S}(u,v)$ of degree p and q, respectively for the direction u and v, is a vector-valued bivariate piecewise rational function.

$$\mathbf{S}(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) w_{i,j} \mathbf{P}_{i,j}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u) N_{j,q}(v) w_{i,j}} \quad u,v \in [0,1] \quad (12)$$

where $\{\mathbf{P}_{i,j}\}$ is a 2D array of control points (or control net), $\{w_{i,j}\}$ are the weights, and the $\{N_{i,p}(u)\}$ and $\{N_{j,q}(v)\}$ are the nonrational B-spline basis as defined by formula (2) that is generated on the knot vectors U and V.

$$U = \{\underbrace{0,\ldots,0}_{p+1}, u_{p+1},\ldots,u_{r-p-1}, \underbrace{1,\ldots,1}_{p+1}\} \quad (13)$$

$$V = \{\underbrace{0,\ldots,0}_{q+1}, v_{q+1},\ldots,v_{s-q-1}, \underbrace{1,\ldots,1}_{q+1}\}$$

where $r = n + p + 1$ and $s = m + q + 1$.

As can be seen from (12), reparameterization of a NURBS surface can be accomplished by applying reparameterization as described in the previous section to each of the parameters u and v. If a reparameterization is able to embed m bit in a NURBS curve, a tensor product NURBS surface is able to store $2m$ bit.

E. ORDERING MULTIPLE EMBEDDING PRIMITIVES

A rational linear reparameterization function discussed above can only embed L bit of information per NURBS curve. A tensor product surface as an embedding primitive, with its two parameters, is able to embed $2L$ bit of information per

primitive. Such amount of information, however, is insufficient in most of the applications.

In order to embed a significant amount of information, e.g., tens to hundreds of bytes, multiple embedding primitives, that are, NURBS curves and surfaces, must be ordered.

A CAD model consisting of multiple surface patches and trim curves contains topology among these objects (Figure 3). The topology can be used to cerate a one-dimensional ordering among objects, for example, by creating spanning tree of objects and traversing the tree in a depth-first order. Furthermore, objects in a CAD model are often numbered sequentially so that various properties (e.g., color, material, etc.) can be associated with them. Such sequential identification number can also be employed to order curve and surface objects in CAD models.

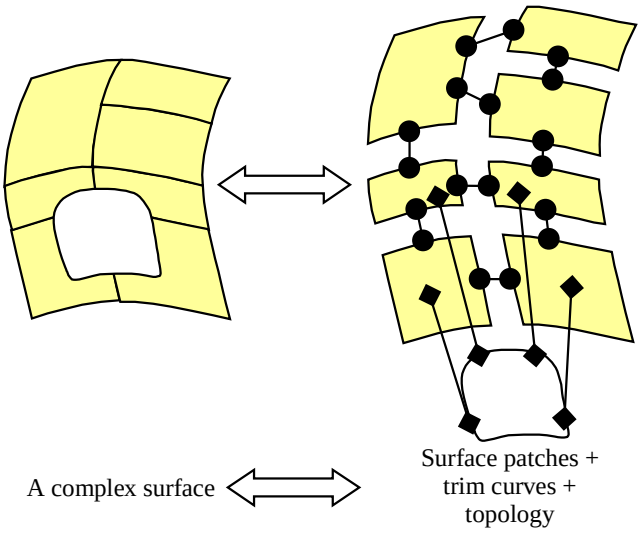


Figure. 3 A surface is consisting of multiple surface patches and trimming curves connected by topology.

Upon embedding, mapping from bit string in a message to an ordered set of embedding primitives can be scrambled by using a pseudo-random number sequence as mentioned before. Such scrambling could make the message secure from third parties, and if combined with repeated embedding, could make the embedding more robust against various disturbances.

F. Increasing Information Density

The algorithm described in Section II does not have high information density, that is, embedded bits per cover data. The algorithm embeds *L* bit in each curve and *2L* bit in a tensor product surface. The amounts of embedded data depend on number of parameters, instead of data size as measured by number of control points and knots. There are several ways to increase embedding information density.

If we use all three degrees of freedom in $g(u)$, three times as many bits per curve or surface can be embedded. However, this method may not be applicable if there is a constraint on knot values, e.g., if the parameter domain is constrained to [a, b]=[0, 1].

We can use a reparameterization function with higher degrees-of-freedom in order to increase amount of data embedded per curve. Such reparameterization is possible without changing degree of the NURBS curve if we employ a *linear rational B-spline* (LRBS) function, as described in Fuhr and Kallay [Fuhr92]. By using their method a C^1 continuous rational linear interpolator can be constructed that pass through an arbitrary number of monotonically increasing data points. A LRBS is specified by a set of monotonically increasing data points and a corresponding set of derivatives (>0) at each data point.

We can use the LRBS reparameterization in the following way. Based on the message bits to be embedded, we set the values of new knots s_i, $i = 0,...,m-1$ while observing the monotonicity requirement. Then, construct a LRBS function that interpolates data points (u_i, s_i), $i = 0,...,m-1$. We could then manipulate *m* values to embed mL bit of information in a curve. In case of a surface patch, assuming the number of knots *m* and *n* for parameters *u* and *v*, we can embed $mL+nL$ bit. If we assume that the first and last knot are excluded from embedding since they are constrained, e.g., to [a, b]=[0, 1], then we can embed $(m-2)L$ bit of information per curve. Similarly, a surface patch with number of knots *m* and *n* could embed $((m-2)L+(n-2)L)$ bits assuming the same constraint.

Regardless of the constraint on the values of knot at both ends, reparameterization by using a LRBS could bring significant improvement in data density if compared to reparameterization by using a simple rational linear function. Furthermore, the amount of data by using LRBS scales with data size, that is, the number of knots.

III. Experiments and Results

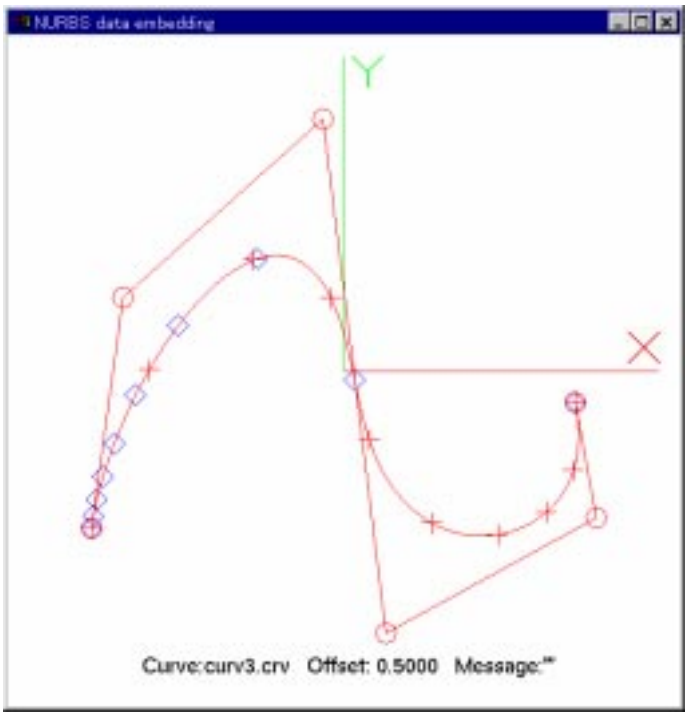
We have implemented the shape- and size-preserving data embedding algorithm for NURBS curves described in the previous section. Current implementation of the algorithm embedded *L* bit per NURBS curve by using a rational linear reparameterization. We have not implemented the method that employs LRBS. Ordering of multiple NURBS curves and surfaces are done by assuming that each curve or surface is explicitly and uniquely numbered. The code is written in C++ using OpenGL graphics API. It is written as a window-system independent code by using GLUT toolkit and GLUI user-interface toolkit, so that it runs on both X-window/UNIX based systems and on Windows NT/95 systems without modifications.

Figure 4 (a)-(d) show effects of reparameterization by using a rational linear function with varying offset *D*. The Straight line segments are the control polygon, in which control points are marked by the circles. On the curve, the domain of parameter [a, b] is uniformly divided into 9 sections so that ten markers corresponding to 10 uniformly spaced parameter values (including both the first and last knots with values a and b.) are marked. Crosses mark the original curve and the rectangles mark the reparameterized curve.

(a) Offset D=0.01

(b) Offset D=0.05

(c) Offset D=0.1



(d) Offset D=0.5

Figure 4. Examples of reparameterizatoin of a NURBS curve with offsets (a) *D*=0.01, (b) *D*=0.05, (c) *D*=0.1, and (d) *D*=0.5. With a small offets, e.g., *D*=0.01, parameterizations of the curves before and after rational linear reparameterization are nearly identical. With the offset *D*=0.5, parameterization is not “good” so that ten equally spaced parameter values mapped to skewed points. Crosses mark the original curve and the rectable marks the reparameterized curve, computed at ten equally spaced parameter values.

A curve reparameterized by a smaller offset value D=0.01 showed parameterization similar to the original curve. However, a curve reparameterized with a large offset, e.g., D=0.5, clearly showed a very skewed parameterization, which is not acceptable. In the experiment, we embedded 8 bit per double precision (64 bit) floating point number representing D, which uses 52 bit for mantissa. This is a significant and useful embedded information density.

IV. Alternative Data Embedding Approaches for Parametric Curves and Surfaces

In this section, we suggest various possible approaches to embed data in parametric curves and surfaces that include Bezier, rational Bezier, B-spline and other curves and surfaces, as well as NURBS curves and surfaces.

We classify data embedding approaches outlined in this section by using two properties of embedding methods, preservation of model shape and preservation of model data size. A *shape-preserving* method preserves exact shape of an original model, while a *shape-altering* method alters it. A size preserving method retains the original data size after embedding, while a size-altering method changes it. These classifications are orthogonal.

Shape-preserving data embedding is desired for most CAD applications. If shape of a model was changed, for example, an automobile door might fail to close properly. If the sole purpose of the model is viewing, however, the shape of the model need not be preserved exactly.

Size-preserving methods preserves data amount occupied by parameters that define curves or surfaces, that are, control points, knot vector elements, and others. By "data size preserving", we mean the numbers of these parameters to remain unchanged. Their exact values, hence exact number of bits of the model after ideal entropy coding, may change. Size-preservation property is preferable since increased data size may burden communication channels and/or storage devices. A bloated model may make a third party suspicious of embedded data in the model.

Please note, in the following, that a discussion on a type of curve can also be applied to a surface generated as tensor products of curves of that type.

A. SHAPE-PRESERVING, SIZE PRESERVING METHODS

Most CAD applications demand exact preservation of shape as well as data size. A shape- and size-preserving data embedding is possible for certain classes of curves that has enough redundancy, e.g., NURBS curves. One such method that employs rational linear reparameterization has been explained in Section II.

B. SHAPE PRESERVING, SIZE ALTERING METHODS

Data embedding that preserves shape, but alters model data size can be achieved by injecting redundancy into representations of parametric curves and surfaces. Such redundancy injection can be achieved by using such techniques as knots insertion, degree elevation, and reparameterization that accompanies degree elevation.

- **Knots Insertion:** For those parametric curves with multiple spans, e.g., nonrational B-spline curves and NURBS curves, new knots can be inserted into the curve. The values of the inserted knot, mere presence of the new knot, or location of the new knot in the knots vector could encode information to be embedded.
- **Degree Elevation:** For nonrational curves, such as Bezier and B-spline curves, degree elevating a curve introduces new control points. The increase in degree itself can be used to encode information; for example, an increase in degree by 2 could encode a number 2. While the location of the new control points can not be controlled, their mere presence/absence could also encode information.
- **Reparameterization Accompanying Degree Elevation:** Reparameterization by using a polynomial or rational polynomial of degree greater one can be applied to rational curves, such as rational Bezier and NURBS curves. If applied, such reparameterization will raise degree of the curve, introducing new control points as well as knots. Value of reparameterized knots can encode information, in a similar manner as the algorithm explained in Section II. A significant amount of information can be embedded into the curve if a function that interpolates through multiple data points such as LRBS is used to reparameterize, as suggested in Section II-F.

While these methods do preserve original geometry of the curve, increase in data size due to embedding can be significant.

C. SHAPE-ALTERING, SIZE-PRESERVING METHODS

If exact shape preservation is not an issue, additional approaches exist in order to embed information into parametric curves and surfaces.

- **Control Points and/or Weights Modulation:** Control points for a curve is a one-dimensional ordered set of 3D or 4D points. (If a set of weights is considered as an integral part of a set of control points, using homogeneous coordinate, each control point is a 4D point.) Similarly, control points for a tensor product surface is a regular two-dimensional (2D) array of 3D or 4D points. These values can be modulated as if they are simply ordered sets of floating point values.

  If geometric transformation is expected, transformation invariant quantities can be employed to encode information, as demonstrated in [Ohbhchi97, Ohbuchi98a]. An example of a transformation invariant quantity is a ratio of volumes of two tetrahedrons, which is affine transformation invariant. Modulations of control point coordinates or weights do change geometric shape of the model being embedding with information.

  If control points and weights are modulated together, shape deformation can be reduced. Modulation of control points is applicable to a large class of parametric curves and surfaces, including Bezier, rational Bezier, B-spline, and NURBS curves and surfaces.

- **Knot Vector Modulation:** Values of knots in the current knot vector, which is an ordered 1D sequence of scalar values, can be modified to encode information. This method is applicable to parametric curves with knots, e.g., B-spline and NURBS curves and surfaces.

Note that the modulation of values, such as control point coordinates, weights, and knot values may be performed either in their original domain or in a transformed domain. For example, an embedding method may compute a 2D discrete cosine transform (DCT) of 2D array of control points of a surface, modulate DCT coefficients to embed information, and transform the coefficients back into the original domain by using an inverse of the DCT. This embedding method may be integrated, for example, with a shape altering (i.e., lossy) compression algorithm for curved surfaces that employed DCT [Masuda98]. Note that the kind of transformation is not limited to DCT. Walsh-Hadamard transformation or wavelet transformations by using many different kinds of analysis/synthesis function pairs, for example, are viable candidates.

V. SUMMARY AND FUTURE WORK

This paper presented an algorithm that embeds data in rational parametric curves and surfaces, in particular, NURBS curves and surfaces, by using reparameterization. We employed rational linear reparameterization, since it enables data embedding that preserves exact geometric shape of the NURBS curves and surfaces as well data-size.

This paper also suggested alternative approaches to embed data in parametric curves and surfaces that are not limited to NURBS. These approaches are classified by using their shape preservation and size preservation properties. Knots insertion and degree elevation are among possible alternative data embedding methods that preserve shape but alter data size.

In the future, we would like to incorporate linear rational B-spline reparameterization into our implementation so that the data density is improved. We also would like to implement alternative data embedding approaches suggested in Section IV.

REFERENCES

[Bender96] W. Bender, D. Gruhl, and N. Morimoto, Techniques for Data Embedding, IBM Systems Journal, Vol. 35, Nos. 3 & 4, 1996.

[Benedens98] O. Benedens, Geometry-Based Watermarking of 3D Models, *IEEE CG&A*, pp. 46-55, January/February, 1999.

[Braudway96] G. Braudway, K. Magerlein, and F. Mintzer, Protecting Publicly-Available Images with a Visible Image Watermark, *IBM Research Report*, TC-20336 (89918), January 15, 1996.

[Cox97] I. J. Cox, J. Kilian, T. Leighton, and T. Shamoon, Secure Spread Spectrum Watermarking for Multimedia, *IEEE Trans. on Image Processing*, Vol. 6, No. 12, pp1673-1678, 1997.

[Farin97] G. E. Farin, *Curves and Surfaces for Computer-Aided Geometric Design, A Practical Guide*, Fourth Edition, Academic Press, San Diego, CA, 1997.

[Fuhr92] R. D. Fuhr and M. Kallay, Monotone linear rational spline interpolation, Computer Aided Geometric Design, Vol. 9, pp. 213-319, Elsevier, 1992.

[Cueziec97] A. Gueziec, G. Taubin, F. Lazarus, and W. Horn, Cutting and Stitching: Efficient Conversion of a Non-Manifold Polygonal Surface to a Manifold, *IBM Research Report* RC-20935 (92693), July, 1997.

[Hartung97] F. Hartung and B. Girod, Copyright Protection in Video Delivery Networks by Watermarking of Pre-Compressed Video, *Lecture Notes in Computer Science*, Vol. 1242, pp.423-436, Springer, 1997.

[Hartung98] F. Hartung, P. Eisert, and B. Girod, Digital Watermarking of MPEG-4 Facial Animation Parameters, *Computer and Graphics*, Vol. 22, No. 4, pp. 425-435, Elsevier, 1998.

[ISO97] ISO/IEC 14772-1 Virtual Reality Model Language (VRML).

[ISO98] ISO/IEC JTC1/SC29/WG11 *MPEG-4 Visual* and *MPEG 4 SNHC*.

[Kanai98] S. Kanai, H. Date, and T. Kishinami, Digital Watermarking for 3D Polygons using Multiresolution Wavelet Decomposition, Proc. of the *Sixth IFIP WG 5.2 International Workshop on Geometric Modeling: Fundamentals and Applications (GEO-6)*, pp. 296-307, Tokyo, Japan, December 1998.

[Lee91] E. T. Y. Lee, and M. L. Lucian, Mobius reparameterizations of rational B-splines, Computer Aided Geometric Design, Vol. 8, pp. 213-215, Elsevier, 1991.

[Masuda98] H. Masuda, R. Ohbuchi, and M. Aono, Compression and Progressive Transmission of Parametric Surfaces, accepted for publication in *Transaction of Information Processing Society of Japan (IPSJ)* (in Japanese).

[Menezes96] A. J. Menezes, P. C. van Oorshot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[Mintzer97] F. Mintzer, G. W. Braudway, and M. M. Yeung, Effective and Ineffective Digital Watermarks, Proceedings of the *IEEE International Conference on Image Processing (ICIP) '97*, Vol. 3, pp. 9-12, 1997.

[O'Ruanaidh96] J. J. K. O'Ruanaidh, W. J. Dowling and F. M. Boland, Watermarking Digital Images for Copyright Protection, *IEE Proc.-Vis. Image Signal Process.*, Vol. 143, No. 4, pp. 250-256, August 1996.

[Ohbuch97] R. Ohbuchi, H. Masuda, and M. Aono, Watermarking Three-Dimensional Polygonal Models, *Proceedings of the ACM Multimedia '97*, Seattle, Washington, USA, November 1997, pp. 261-272.

[Ohbuchi98a] R. Ohbuchi, H. Masuda, and M. Aono, Watermarking Three-Dimensional Polygonal Models Through Geometric and Topological Modifications, pp. 551-560, *IEEE Journal on Selected Areas in Communications*, May 1998.

[Ohbuchi98b] R. Ohbuchi, H. Masuda, and M. Aono, Geometrical and Non-Geometrical Targets for Data Embedding in Three-Dimensional Polygonal Models, *Computer Communications*, Vol. 21, pp. 1344-1354, Elsevier, (1998).

[Piegl97] L. Piegl, W. Tiller, *The NURBS Book*, 2nd Edition, Springer, Berlin, 1997.

[Pfitzmann96] B. Pfitzmann, Information Hiding Terminology, in R. Anderson, Ed., *Lecture Notes in Computer Science* No.1174, pp. 347-350, Springer, 1996.

[Smith96] J. R. Smith and B. O. Comiskey, Modulation and Information Hiding in Images, in R. Anderson, Ed., *Lecture Notes in Computer Science* No.1174, pp. 207-296, Springer, 1996.

[Tanaka90] K. Tanaka, Y. Nakamura, and K. Matsui, Embedding Secret Information into a Dithered Multilevel Image, *Proc. 1990 IEEE Military Communications Conference*, pp. 216-220, 1990.

[Walton95] S. Walton, Image Authentication for a Slippery New Age, *Dr. Dobb’s Journal*, pp. 18-26, April 1995.

[Yeo99] B-L. Yeo and M. M. Yeung, Watermarking 3D Objects for Verification, *IEEE CG&A*, pp. 36-45, January/February, 1999.

[Yeung97] M. M. Yeung, F. C. Mintzer, G. Braudway, and A. R. Rao, Digital Watermarking For High-Quality Imaging, Proceedings of the *First IEEE Workshop on Multimedia Signal Processing*, Princeton, NJ, USA, June, 1997, pp. 357-362.

[Yeung97] M. M. Yeung and F. Mintzer, An Invisible Watermarking Techniques for Image Verification, Proceedings of the *IEEE ICIP '97*, Vol. 2, pp. 680-683, 1997.

[Zhao95] J. Zhao and E. Koch, Embedding Robust Labels into Images for Copyright Protection, Proc. of *the Int’l. Congress on Intellectual Property Rights for Specialized Information, Knowledge, and New Technologies*, Vienna, August 1995.