In Visual Communications and Image Processing (VCIP) 2002, Proc. of SPIE, vol. 4671, pages 280–291, San Jose, CA, USA, Jan. 21–23, 2002. `http://ltswww.epfl.ch/~dsanta.`

Compression of parametric surfaces for efficient 3D model coding

Diego Santa-Cruz[a], Touradj Ebrahimi[b]

Signal Processing Laboratory – Swiss Federal Institute of Technology, Lausanne

ABSTRACT

In the field of compression, the type of 3D models traditionally considered is that of polygonal meshes, for which several efficient compression techniques have been proposed in the recent years. Nowadays, an increasing proportion of 3D models are created by a synthesis or modeling process, instead of captured from the real world. Such models are most often given as parametric surfaces, which have several advantages over polygonal meshes, such as resolution independence and a more compact representation. This paper proposes a method to code parametric surfaces, given as Non-Uniform Rational B-Splines (NURBS). The coding scheme consists in coding the NURBS parameters (knots and control points) using a predictive scheme, coupled with uniform quantization and entropy coding. The multiplicity of knots is preserved by decomposing the knot vectors in a break vector (the values) and a multiplicity map. The rate-distortion of the proposed scheme is evaluated and compared against compressed triangular meshes. The results show that a considerable compression ratio is achievable under visually lossless conditions, that outperforms by far triangular meshes. In addition of having a better rate-distortion performance, the coding scheme enables the efficient transmission of synthesized 3D models retaining their resolution independence.

Keywords: B-Spline, NURBS, compression, 3D model, coding, MPEG-4

1. INTRODUCTION

In the recent years, the use of 3D models in multimedia applications has received increased attention. In the field of compression the source of 3D models has traditionally been some capture device, possibly coupled with an analysis system. It could be a direct 3D acquisition system, such as a 3D scanner, or an analysis system that would infer the 3D shape from a two dimensional image or video, possibly stereo. The former usually generate high resolution polygonal models which are subsequently simplified to match the complexity, bandwidth and storage requirements of the application. The latter have been extensively studied but the relatively poor performance and high complexity of current systems (except for specific cases where *a priori* knowledge about the nature of the objects is available) has refrained a wide deployment of these techniques. Progress in computer graphics has changed this situation. Nowadays an increasing proportion of visual content is created through some synthesis and modeling process, rather than captured from the real world.

On a parallel but related path, the way we consume audio-visual information is changing. As opposed to recent past and a large part of today's applications, interactivity is becoming a key element in the way we consume information. In the context of interest in this paper, this means that when coding visual information (an image or a video for instance), previously obvious considerations such as decision on sampling parameters are not so obvious anymore. To be more clear, let us provide an example. In a conventional digital video coding problem, the sampling (i.e. number of pixels in the image) mostly depends on the display resolution. Knowing the resolution of the display, there is no need to acquire and then to code a signal with sampling characteristics beyond those that the display is capable of reproducing. In an interactive environment, this is not true anymore. Even using a conventional display with, for instance, a resolution of 300×400 pixels, one could request to examine (and therefore display) more closely an object in a scene. This means that because of interactivity, the representation used to code the scene should allow the display of objects in a variety of

[a] Diego.SantaCruz@epfl.ch (corresponding author), [b] Touradj.Ebrahimi@epfl.ch. EPFL / DE / LTS, CH-1015 Lausanne, Switzerland.

resolutions, and ideally up to infinity. One effective way to resolve this problem would be to use a resolution independent representation.

In the field of compression, the type of 3D models traditionally considered is that of piecewise planar polygons, or more precisely polygonal meshes and their associated properties (normals, colors, textures, etc.). While very flexible, this class presents the major limitation of being resolution dependent, which is not suitable to the interactive setting described above. However, as we have mentioned, 3D models are increasingly created through some synthesis or modeling process, rather than being captured. These 3D models are described in terms of piecewise smooth functions, most often parametric surfaces, instead of piecewise planar polygons. Such descriptions are therefore inherently resolution independent and thus more suitable to the interactive setting described above.

Techniques to efficiently represent and compress polygonal meshes have been investigated[1,2,3] and even standards have been produced, such as MPEG-4 version 2.[4,5] However, only few techniques to efficiently compress 3D models in terms of smooth piecewise functions have been proposed. Gopi *et al.* propose one such approach in.[6] However, their method relies on converting a model to an intermediate representation based on fixed elements (triangular bi-cubic B-Spline patches), which incurs some loss of information from the original 3D model. This paper proposes an alternate solution that preserves the original information more closely.

Many schemes for representing parametric surfaces exist, of which Non-Uniform Rational B-Spline (NURBS) patches[7] is one of the most popular. NURBS patches are a common tool for surface modeling, popular in CAD and virtual character generation, among others. NURBS have been already proposed for inclusion[8] in the VRML standard,[9] but in an uncompressed form. Here we propose a method to efficiently compress NURBS with a small and controllable loss.

The paper is organized as follows. Sec. 2 provides a brief overview of NURBS and introduces the concepts and notation necessary to the development of the subsequent sections. Sec. 3 explains the coding scheme and Sec. 4 evaluates the proposed scheme in terms of its rate-distortion performance. Finally Sec. 5 draws conclusions and points out future work.

2. NURBS MODELS

As was previously mentioned NURBS are a popular way of representing parametric surfaces. In general, each 3D NURBS model is composed of several independent and relatively simple NURBS patches that together form a complex 3D surface. There is a variety of different schemes for representing NURBS surface patches. However the most common is *tensor product*, which is the one used in this paper. Another popular scheme is *triangular* patches.[10] In the following we will briefly review B-Spline functions, tensor product surfaces and NURBS patches. A detailed description of these concepts can be found in.[11,12]

As an example, Fig. 1 shows three NURBS models. The goblet model is made of just one patch, quadratic in one direction and cubic in the other. This demonstrates the flexibility of NURBS, where part of the patch is a surface of revolution, while another is a quadrilateral. The coke and scissors models are made of 21 and 7 patches, respectively.

2.1. B-Splines and NURBS

The basis functions for NURBS are the B-Spline functions. Given $U = \{u_0, \ldots, u_r\}$, a non-decreasing sequence of $r+1$ real numbers (i.e. $u_i \leq u_{i+1}, 0 \leq i \leq r-1$), the ith B-spline function of pth degree (order $p+1$), denoted by $N_{i,p}(u)$, is defined, recursively, as

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

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

(a) (b) (c)

Figure 1. Rendered surfaces of NURBS models: (a) goblet, (b) coke and (c) scissors

where the quotient 0/0 is defined to be zero. The u_i values are called *knots*, and U is the *knot vector*. Note that knots can have a *multiplicity* higher than one.

A B-Spline tensor product surface $\mathbf{S}(u,v)$ is defined as a mapping from a rectangular region of the (u,v) plane in $\mathbb{R}^2$ into Euclidean $\mathbb{E}^3$ space. The mapping is given by

$$\mathbf{S}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)\mathbf{P}_{i,j} \qquad \begin{matrix} u_{p+1} \le u \le u_{r-p-1} \\ v_{q+1} \le v \le v_{s-q-1} \end{matrix} \tag{2}$$

where $\mathbf{P}_{i,j}$ is a set of $(n+1)\times(m+1)$ points, referred to as *control points*, and u and v are the two independent variables. The two sets of B-Spline functions $\{N_{i,p}(u)\}$ and $\{N_{j,q}(v)\}$, respectively of degree p and q, are defined on the knot vectors U and V. The number of knots in U, $r+1$, and V, $s+1$, and the number of control points are related by $r = n+p+1$ and $s = m+q+1$. The set of control points are referred to as the *control net*.

A rational B-Spline surface is obtained by considering a polynomial surface in projective $\mathbb{P}^4$ space, which is then projected to Euclidean $\mathbb{E}^3$ space. The resulting expression is thus

$$\mathbf{S}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} R_{i,j}(u,v)\mathbf{P}_{i,j} \qquad \begin{matrix} u_{p+1} \le u \le u_{r-p-1} \\ v_{q+1} \le v \le v_{s-q-1} \end{matrix} \tag{3}$$

where $R_{i,j}(u,v)$ are the rational B-Spline functions of degree p and q, given by

$$R_{i,j}(u,v) = \frac{N_{i,p}(u)N_{j,q}(v)w_{i,j}}{\sum_{k=0}^{n}\sum_{l=0}^{m} N_{k,p}(u)N_{l,q}(v)w_{k,l}} \tag{4}$$

and $w_{i,j}$ is the homogenizing coordinate, or *weight*, of control point $\mathbf{P}_{i,j}$. That is, the homogeneous coordinates corresponding to a point $\mathbf{P}_{i,j} = (x,y,z)$ are $(xw_{i,j}, yw_{i,j}, zw_{i,j}, w_{i,j})$, or equivalently $(w_{i,j}\mathbf{P}_{i,j}, w_{i,j})$, with some abuse of notation. Note that, strictly speaking, a NURBS surface is not a tensor product surface once it has been projected to Euclidean space.

The rational functions $\{R_{i,j}\}$ are well defined (i.e. the denominator is non-zero) over the parametric domain of definition of the surface, provided that all weights are non-zero and of the same sign. This is because of the non-negativity and partition of unity properties of the B-Spline functions.[11] In the following it is assumed that all weights are positive, without loss of generality (if the weights are negative the surface is not changed by multiplying all weights by -1).

2.2. NURBS properties

NURBS surface patches possess several interesting properties, many of which are at the origin of their popularity in geometric modeling. Here we only mention those relevant to this study, as well as some of the most important.

In general, a NURBS surface does not interpolate the control points that define it, but it is approximated by any triangulation of its control net. In addition, a NURBS patch is always contained in the convex hull of its control points. A control point only influences the shape of the surface in its vicinity, that is a point $\mathbf{P}_{i,j}$ affects $\mathbf{S}(u,v)$ only for $(u,v) \in [u_i, u_{i+p+1}) \times [v_j, v_{j+q+1})$. The effect of a weight is to move the surface closer or farther away from its corresponding control point. The larger the weight, the closer the surface is to the control point.

NURBS surfaces can exactly represent sections of arbitrary conics, such as spheres, cylinders and ellipsoids. In addition, the parametric continuity of a NURBS surface is always C^∞, except at knot values. At a u knot of multiplicity k it is $C^{(p-k)}$ in the u direction. Analogously, at a v knot of multiplicity k it is $C^{(q-k)}$ in the v direction. The geometric (i.e. visual) continuity is however not entirely determined by parametric continuity. At a knot, it can be increased by placing the control points in a special way, or decreased by using multiple coincident control points.

The first and last knots of any knot vector have no influence on a NURBS. In fact, given a knot vector U with $r+1$ knots, it is possible to modify the values u_0 and u_r without affecting the NURBS, provided that the new knot vector is also a non-decreasing sequence of real values. Furthermore, the knot vectors can be normalized to the $[0,1]$ range without modifying the shape of a NURBS surface, and without modifying its control points.

Although knot vectors can be an arbitrary sequence of non-decreasing values, special classes can be distinguished which are used very often. These are:

- *Clamped* knot vectors, where the multiplicity of the first and last knots equals the order of the B-Spline function. That is, for the u direction, $U = \{\underbrace{a, \ldots, a}_{p+1}, u_{p+1}, \ldots, u_{r-p-1}, \underbrace{b, \ldots, b}_{p+1}\}$. For a NURBS surface which has clamped knot vectors in both directions, the corner control points coincide with the corners of the surface.
- *Uniform* knot vectors, where all the knots are uniformly spaced. That is $u_{i+1} - u_i \equiv c$ for $i = 0, \ldots, r$.
- *Clamped uniform* knot vectors, where all the interior knots of a clamped knot vector are uniformly spaced.

3. CODING SCHEME

In the following a powerful yet simple coding scheme for NURBS patches is presented. It consists in coding the NURBS parameter instead of coding the geometric shape of the surface directly. The knot vectors and control points are handled independently but in a similar manner. The basic building blocks are prediction, quantization and entropy coding.

In the following it is assumed that all knot vectors are of the form $\{0, 0, u_2, \ldots, u_{r-2}, 1, 1\}$, without any loss of generality. As previously mentioned, the first and last knots can be freely modified and thus u_0 set to the value of u_1, and u_r to the one of u_{r-1}. In addition, a linear reparametrization does not modify the control points, the degree of the B-Spline functions, or the geometric shape of the surface.[11] Using a linear reparametrization that maps $[u_0, u_r]$ to $[0,1]$ yields the desired knot vector form. Note that the NURBS surface does change parametrically, but it is of no importance to our application.

Likewise, it is also assumed, without loss of generality, that all weights $w_{i,j}$ of a NURBS patch are in the interval $(0,1]$ and that the largest one is 1. From Eqn. (4) it is trivial to see that multiplying all the weights by a non-zero constant does not modify the NURBS surface (strictly speaking the surface is modified in projective space, but its projection to Euclidean space remains unchanged). Thus, dividing all weights by the largest one yields the desired form.

3.1. Prediction and quantization

From the above brief overview of NURBS, it is clear that knot vectors possess some structure. Likewise, control points also possess some structure since they describe objects in 3D space. In order to reduce the redundancy of the data to code, we use a predictive coding scheme, namely differential pulse coded modulation (DPCM). For NURBS, there is unfortunately no clear relationship between the distortion on the knot values and/or control point coordinates and the actual distortion of the surface itself. A conservative approach has thus been taken, where the maximum knot and control point distortion is guaranteed, given a quantization step size. This is achieved by the use of a uniform scalar quantizer.

3.1.1. Knot vectors

As previously mentioned, knots define the B-Spline functions and their multiplicity affects the surface continuity. It is thus important to preserve the knot multiplicity in the coding. To this end, a knot vector is decomposed into a *break vector* $U' \equiv \{u'_i\}$ and a *multiplicity map* $U^m \equiv \{u^m_i\}$. The break vector contains the values of the knot vector, in the same order, but disregarding their multiplicity (i.e. each value appears only once, even if it is a multiple knot). The multiplicity map contains the multiplicity of each break value, minus one. For example, the knot vector $U = \{0, 0, 0, 0.25, 0.25, 0.5, 0.75, 0.75, 1, 1, 1\}$ is decomposed in the break vector $U' = \{0, 0.25, 0.5, 0.75, 1\}$ and the multiplicity map $U^m = \{2, 1, 0, 1, 2\}$. We denote by $r' + 1$ the number of elements in the break vector.

The multiplicity map is losslessly encoded as explained in Sec. 3.2. The break vector is coded using DPCM and uniform scalar quantization, prior to entropy coding. The order of the corresponding B-Spline function as well as the number of knots is coded as overhead information in the bitstream header.

Since break vectors are always a strictly increasing sequence and most knot vectors are uniform or close to, one can expect the difference between consecutive break values to remain constant. This is used as the DPCM predictor. The prediction error ϵ_i can thus be expressed as

$$\epsilon_i = (u'_{i+1} - \hat{u}'_i) - (\hat{u}'_i - \hat{u}'_{i-1}) = u'_{i+1} - 2\hat{u}'_i + \hat{u}'_{i-1} \qquad i = 0, \ldots, r' - 2 \tag{5}$$

where $\hat{u}'_i$ denotes the reconstructed break values, as would be seen by the decoder (using the decoded values avoids drift due to quantization error accumulation).

The prediction error is quantized with a mid-rise uniform scalar quantizer of step size Δ_k. The quantization index is thus $\zeta_i = \langle \epsilon_i / \Delta_k \rangle$, where $\langle \cdot \rangle$ denotes the rounding operator. The break values are decoded as

$$\hat{u}'_i = \zeta_{i-1}\Delta_k + 2\hat{u}'_{i-1} - \hat{u}'_{i-2} \qquad i = 1, \ldots, r' - 1 \tag{6}$$

The first and last break values are implicit: $\hat{u}'_0 = 0$ and $\hat{u}'_{r'} = 1$. In order to compute ϵ_0 the coder and decoder must agree in the value of $\hat{u}'_{-1}$. Following the fact that most break vectors are uniform, this value is set to $\hat{u}'_{-1} = -1/r'$, so that ϵ_0 is minimized in such common cases. As for any DPCM coder with a uniform scalar quantizer, the coding error for break values is bounded by $\Delta_k/2$, as $|u'_i - \hat{u}'_i| = |\epsilon_{i-1} - \zeta_{i-1}\Delta_k| \leq \Delta_k/2$.

Figure 2 shows the histogram (calculated over 85 samples) of the DPCM prediction error, for the non-uniform break vectors of the scissors model. From the figure it is clear that the scheme produces a skewed distribution that is amenable to entropy coding.

Although effective, this prediction scheme cannot guarantee that the decoded break vector will be a sequence of strictly increasing values. If the quantization step size is too large, it can happen that $\hat{u}'_i > \hat{u}'_{i+1}$ for some i, leading to illegal break and knot vectors. An upper bound on Δ_k that guarantees proper coding is $\min\{u'_{i+1} - u'_i\}$, since that implies $u'_i + \Delta_k < u'_{i+1}$ and

$$\hat{u}'_i \leq u'_i + \Delta_k/2 < u'_{i+1} - \Delta_k/2 \leq \hat{u}'_{i+1} \Rightarrow \hat{u}'_i < \hat{u}'_{i+1} \text{ for all } i$$

However, this upper bound is not tight and larger values of Δ_k can still produce properly coded break vectors. In practice, the coder must check that the reconstructed break values form a strictly increasing sequence, and reduce the quantization step size otherwise. This is fairly simple since a coder must always calculate the reconstructed break values in order to calculate the prediction. As it is shown in Sec. 4, the bitrate necessary to properly code knot vectors is very low, and thus using a fine quantization step size is not a problem in itself.

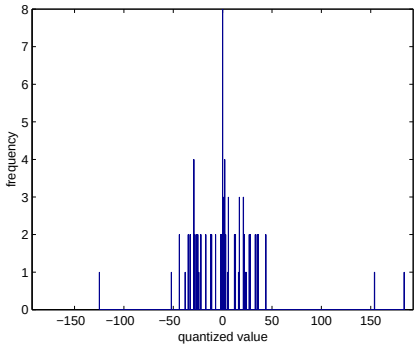


Figure 2. Histogram of the quantized prediction error for the eight (out of 14) non-uniform break vectors of the scissors model.

3.1.2. Control points

NURBS control points can be expressed in terms of their affine coordinates and their associated weights or in terms of their homogeneous coordinates. Since the control net approximates the surface and weights are used to adjust the distance to a control point, affine coordinates (which are the control points in three-space) possess better structure than homogeneous ones. In fact the variations of the weights induce large variations on the homogeneous coordinates making them difficult to predict. This is not the case for the affine coordinates. For this reason we code the affine coordinates and their associated weights, instead of the homogeneous ones.

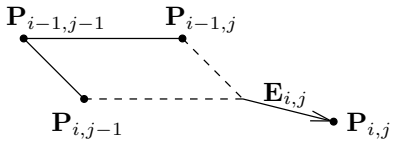


Figure 3. Parallelogram prediction rule

The DPCM predictor used for the affine coordinates is the parallelogram rule which has been successfully used in the compression of vertex coordinates of 3D triangular meshes.[3] This rule predicts that given three known vertices, the fourth will form a parallelogram, as is shown in figure 3. The weights are also predicted in a similar fashion, but in 2D. Using vector notation for the control point coordinates and denoting as $\mathbf{E}_{i,j}$ the prediction error for coordinates and as $\delta_{i,j}$ the one for weights, one obtains:

$$\begin{aligned} \mathbf{E}_{i,j} &= \mathbf{P}_{i,j} - \hat{\mathbf{P}}_{i-1,j} - \hat{\mathbf{P}}_{i,j-1} + \hat{\mathbf{P}}_{i-1,j-1} \qquad i = 0,\ldots,n \\ \delta_{i,j} &= w_{i,j} - \hat{w}_{i-1,j} - \hat{w}_{i,j-1} + \hat{w}_{i-1,j-1} \qquad j = 0,\ldots,m \end{aligned} \tag{7}$$

where $\hat{\mathbf{P}}_{i,j}$ and $\hat{w}_{i,j}$ denote the reconstructed values of the control point coordinates and weights, respectively, as would be seen by the decoder.

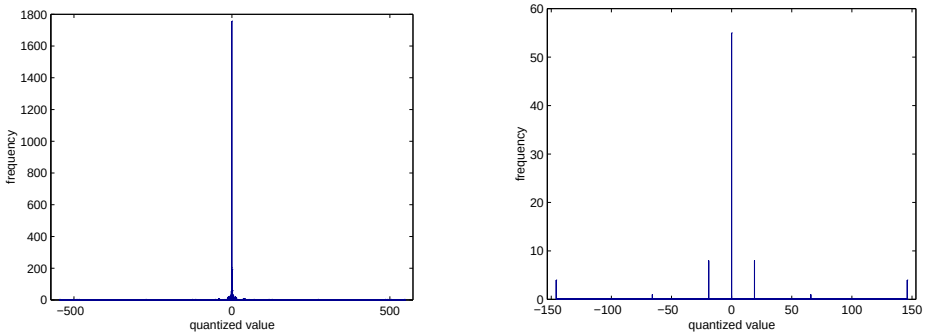
The prediction errors $\mathbf{E}_{i,j}$ and $\delta_{i,j}$ are quantized using two different midrise uniform scalar quantizers, of step size Δ_c and Δ_w, respectively. The prediction error $\mathbf{E}_{i,j}$ is however normalized by the decoded value $\hat{D}$, of the maximum overall dynamic range D of the x, y and z coordinates. The quantization indices are thus $\mathbf{Q}_{i,j} = \langle \mathbf{E}_{i,j}/(\hat{D}\Delta_c) \rangle$ and $\xi_{i,j} = \langle \delta_{i,j}/\Delta_w \rangle$, and the maximum error for the dequantized coordinate and weight values are $\hat{D}\Delta_c/2$ and $\Delta_w/2$, respectively. The weights are, as previously mentioned, already normalized to the $(0, 1]$ range and require no further scaling. The control point coordinates and weights are reconstructed as

$$\begin{aligned} \hat{\mathbf{P}}_{i,j} &= \mathbf{Q}_{i,j}\hat{D}\Delta_c + \hat{\mathbf{P}}_{i-1,j} + \hat{\mathbf{P}}_{i,j-1} - \hat{\mathbf{P}}_{i-1,j-1} \\ \hat{w}_{i,j} &= \xi_{i,j}\Delta_w + \hat{w}_{i-1,j} + \hat{w}_{i,j-1} - \hat{w}_{i-1,j-1} \end{aligned} \tag{8}$$

A proper initial value is needed for $\hat{\mathbf{P}}_{-1,-1}$ to initialize the predictor. To avoid large values of $\mathbf{Q}_{0,0}$, which could hinder entropy coding, we set $\hat{\mathbf{P}}_{-1,-1} = \hat{\mathbf{P}}_{\text{init}}$, which is an approximation of $\mathbf{P}_{0,0}$, coded as overhead information in exponent-mantissa representation. The precision of the representation is adjusted implicitly to guarantee a small magnitude for $\mathbf{Q}_{0,0}$. The values of $\hat{\mathbf{P}}_{i,-1}$ and $\hat{\mathbf{P}}_{-1,j}$ are also considered to be $\hat{\mathbf{P}}_{\text{init}}$. Thus the control points along the topmost row are predicted as their left neighbor and those along the leftmost column as their top neighbor.

For weights the inexistent values $w_{i,-1}$ and $w_{-1,j}$ are considered to be 1. Since, in general, most of the weights are equal to one, this works reasonably well, without requiring the use of a special value as is the case for the control point coordinates.

Fig. 4 shows the histograms of the quantized coordinate and weight prediction error for the control points of the scissors model, calculated over 3006 and 81 values, respectively. As it can be seen, the distribution of the values is very skewed, which is well adapted to an efficient entropy coding.



(a) (b)

Figure 4. Histogram of the quantized control point prediction error for all seven patches of the scissors model: (a) coordinate values $Q_{i,j}$; (b) weight values $\xi_{i,j}$. For the weights, only the truly rational patch is considered (one out of seven).

3.2. Entropy coding

The quantized values ζ_i, $\mathbf{Q}_{i,j}$ and $\xi_{i,j}$, as well as the knot multiplicity map U^m, are entropy coded using the MQ adaptive binary arithmetic coder, which is used in the JPEG 2000[13] and JBIG-2[14] standards. This arithmetic coder is very similar to the more widely known QM coder used in the JPEG[15] standard.

Since many NURBS patches use uniform and/or clamped knot vectors, it is beneficial to signal those explicitely. In fact, uniform knot vectors are entirely determined by their number of elements and their clamped or un-clamped type. Similarly, uniform break vectors require only the coding of their associated multiplicity map in addition to what is required by uniform knot vectors. Therefore, clamped and uniform knot vector flags are coded, followed by an eventual uniform break vector flag. Each flag type is coded with an independent context of the arithmetic coder.

In the case where a knot vector is non-uniform, its multiplicity map U' needs to be coded. The entropy coder in this case is similar to that used for DPCM coding in JPEG,[15] which is a modified version of the method devised by Langdon.[16] The basic idea is to select the correct statistics by signalling the $\log_2$ bin of the u'_i value, and then coding the magnitude bits with the corresponding context. Similarly, in the case where a break vector is non-uniform the quantized break values need to be coded, as explained below.

A simple, yet effective, bitplane arithmetic coder is used to compress the quantized values of break vectors and control points. The values are coded in their sign-magnitude representation. The same coder is used for break values and control point coordinates and weights. The coding proceeds from the most significant to the least significant bitplane, where the maximum number of bitplanes is derived from quantization step sizes.

For typical data the maximum number of bitplanes is larger than the number of bitplanes occupied by the largest magnitude and thus a considerable amount of leading most significant bitplanes are all zero. The number of these is explicitely signalled as a sequence of zeros terminated by a one, encoded with a separate context, thus reducing the number of bitplanes to code. For each coefficient, the most significant bits are coded with another context until the first one bit is encountered. Then the remaining bits (i.e. the magnitude refinement bits) are coded with another context. For non-zero values the sign is coded seperately with yet another context.

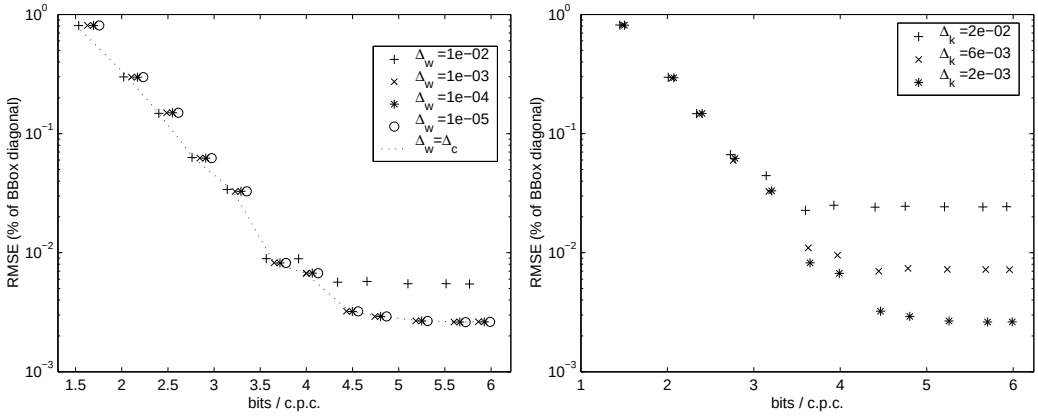
4. EXPERIMENTAL RESULTS

The rate distortion performance of the proposed NURBS coding scheme has been evaluated. The distortion between the coded and original surfaces is measured by means of the Hausdorff distance. Let $d(p, S')$ be the Euclidean distance from a point p on surface S to the closest point on surface S' (i.e the Hausdorff distance). The one-sided L^2 distance between S and S', $d(S, S')$, is given by

$$d(S, S') = \left(\frac{1}{area(S)} \int_{p \in S} d(p, S')^2 \mathrm{d}p \right)^{1/2}$$

This distance is clearly asymmetric. The two-sided distance is defined as $\max\{d(S, S'), d(S', S)\}$, which is symmetric. Evaluating this two-sided L^2 distance directly on the NURBS surface is an extremely difficult task. Here we have tessellated the NURBS surfaces as very fine triangular meshes, and the distance between these is measured by the Metro tool.[17] In the following, the L^2 error is expressed relative to the length of the model's bounding box diagonal, and the rate as the number of bits per control point coordinate (bits/c.p.c.).

In order to vary the bitrate of a coded NURBS, three quantization step sizes (Δ_k, Δ_c and Δ_w) can be adjusted independently. It is clear that, in general, reducing one quantization step size, without modifying the others, will increase the bitrate and reduce the distortion. However, the optimum setting for these three parameters that minimizes the distortion for a given bitrate is not well determined. Nevertheless, as is shown below, some simple rules can be applied to find a semi-optimal trade-off between the different parameters.

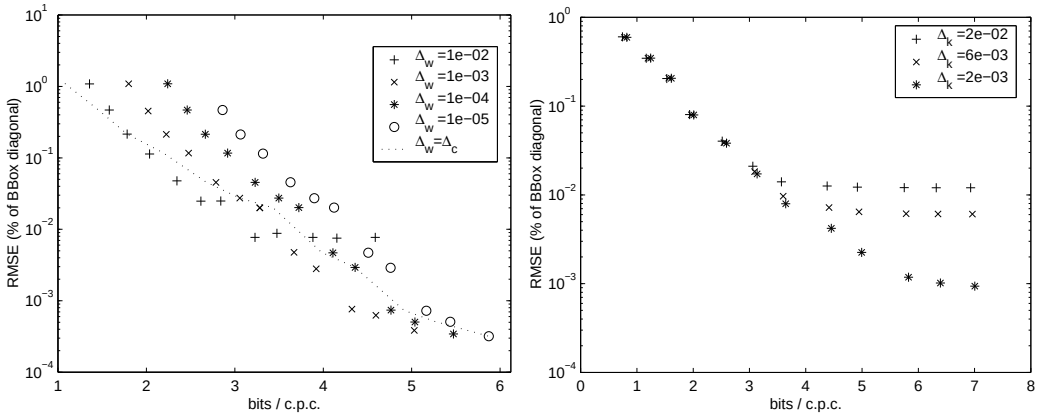


(a) (b)

Figure 5. Rate-distortion curves for the goblet model. (a) Curves for fixed $\Delta_k = 2 \cdot 10^{-3}$ and different Δ_w, for which Δ_c is varied to obtain different bitrates. (b) Curves for $\Delta_w = \Delta_c$, and various Δ_k.

Fig. 5a shows various rate-distortion curves for the goblet model, where the knot quantization step size Δ_k is fixed to some small value. The dotted line shows the rate-distortion curve when $\Delta_w = \Delta_c$. As it can be seen, $\Delta_w = \Delta_c$ is close to optimal for all bitrates, as opposed to some fixed Δ_w value. Furthermore, decreasing Δ_w below some value (in this case $2 \cdot 10^{-3}$) does not decrease the distortion by any noticeable amount, at any bitrate, although it decreases the coding efficiency. Finally, one can also observe that, beyond some bitrate (in

this case 4 bits/c.p.c.), almost no reduction in distortion occurs. That is, the system "saturates". Fig. 5b shows the rate-distortion curves for various values of Δ_k, where $\Delta_w = \Delta_c$. One can observe that the aforementioned "saturation" is, as it could be expected, due to the value of Δ_k: beyond some bitrate, it is necessary to decrease Δ_k to obtain a noticeable reduction in distortion. Finally, one can also see that the increase in bitrate incurred by reducing Δ_k tenfold is fairly small. This can be explained by the fact that the number of knot values is normally small when compared to the number of control point coordinates and weights and that uniform knot or break vectors are efficiently coded. Therefore, using a value of Δ_k that is smaller than optimal will not adversely affect the compression efficiency.



(a) (b)

Figure 6. (a) Coke model's rate-distortion curves for fixed $\Delta_k = 2 \cdot 10^{-3}$ and different Δ_w, for which Δ_c is varied to obtain different bitrates. (b) Scissors model's rate-distortion curves for $\Delta_w = \Delta_c$, and various Δ_k.

Fig. 6a shows the same results as Fig. 5a, but for the coke model. In this case, weight coding represents a significant amount of the total bitrate and thus the tradeoff between Δ_c and Δ_w is more involved. The $\Delta_w = \Delta_c$ setting is not near-optimal anymore, although it still remains a reasonable simplification. For the coke model the setting of Δ_k has virtually no influence on the coding efficiency. This is because only four of the 42 break vectors are non-uniform and require explicit coding of the quantized values.

Fig. 6b shows results for the scissors model. Similar remarks apply. However, for this model, the setting of Δ_w has a very small effect on the coding efficiency. This is because there is only one truly rational patch and thus weight coding represents only a small fraction of the overall bitrate.

	ASCII	ASCII	binary	binary
	bits/c.p.c.	bytes	bits/c.p.c.	bytes
goblet	13.53	1279	10.11	956
coke	11.10	8885	10.63	8510
scissors	15.45	5961	13.15	5075

Table 1. Compressed size of original models, in ASCII and binary (32 bit floating point). Compressed with `gzip`, maximum level.

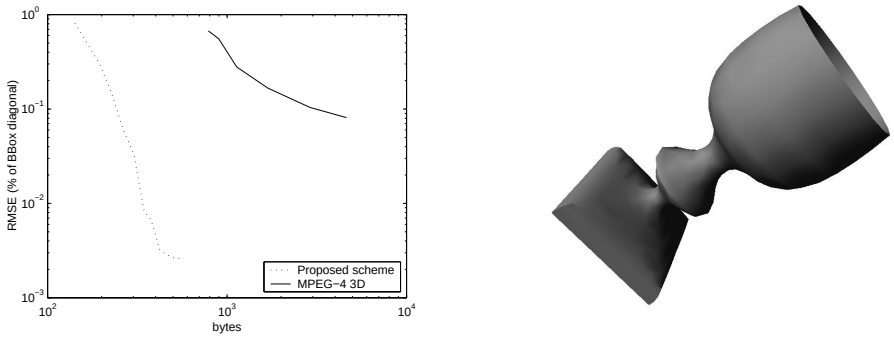
As a basis for comparison, table 1 shows the file sizes of the original NURBS models, in ASCII and binary form, compressed with the popular general purpose `gzip` compressor. By comparing these numbers to the results shown in Figs. 5 and 6, it is clearly seen that the proposed coding scheme is very advantageous, while remaining simple. For an L^2 error of 1 in 10^4, around three times better compression than `gzip`'ed binary is achieved. For an error of 1 in 10^3 that factor increases to 3.5 or even 6, depending on the model. Fig. 7 shows

the rendered decoded models, where no visual differences are visible with respect to the original ones. The proposed scheme achieves, for visually lossless results, a compression factor four or more times larger than the lossless compression obtained by `gzip`'ing the binary files.

(a) (b) (c)

Figure 7. Coded NURBS models: (a) goblet, 2.4 bits/c.p.c., L^2 error $14.8 \cdot 10^{-4}$; (b) coke, 2.3 bits/c.p.c., L^2 error $4.8 \cdot 10^{-4}$; (c) scissors 2.5 bits/c.p.c., L^2 error $4.0 \cdot 10^{-4}$;

Finally, Fig. 8 compares the proposed NURBS coding scheme with compressed polygonal meshes obtained by tessellation of the goblet model. The mesh compression used is that of MPEG-4,[5] namely the 3DMC reference software of Feb. 21, 2001. It is clearly seen that compressed NURBS are much more compact than polygonal meshes, even in the case of coarse ones. The proposed scheme, besides providing more than 5 times better compression, provides much better visual quality for comparable L^2 errors. In fact, surface normals are implicit in NURBS, whereas they must be estimated when rendering the compressed polygonal meshes. MPEG-4 allows to code the "true" normals of a model, but that increases even more the difference in coding gain with respect to the proposed scheme.



(a) (b)

Figure 8. Comparison of NURBS vs. polygonal meshes, on the goblet model. (a) Rate-distortion behavior of MPEG-4 compressed polygonal meshes and the proposed NURBS coding scheme. (b) Visual results for MPEG-4 on the goblet model with 612 vertices and 1192 triangles, compressed size is 1686 bytes and the L^2 error is $16.7 \cdot 10^{-4}$ with respect to the original NURBS model.

5. CONCLUSIONS AND FUTURE WORK

In this paper we propose a method for lossy compression of NURBS models generated by 3D geometric modeling systems, such as those used in CAD and virtual character generation. The method is effective yet simple. It

uses DPCM with ad-hoc predictors, scalar quantization and arithmetic coding. The use of a uniform scalar quantizer guarantees a maximum bound for the error on the various NURBS parameters. The rate-distortion characteristics of the method have been evaluated on various models and insight has been given as to how to get a good tradeoff between the different quantization step sizes. Comparisons against lossless compression of the NURBS models and MPEG-4 based compression of triangular meshes have also been performed. The results show that the proposed method is efficient even at visually lossless settings. In addition, the results also compare very favorably to what can be obtained by compression of triangular meshes derived from the same models. Furthermore, the NURBS based compression retains the full description of the model, which is otherwise lost if converted to triangular meshes.

Although the proposed method is effective, two main subjects deserving further work are worth mentioning. In the current setting, heuristics have been used to derive a proper tradeoff between the different quantization parameters that achieves good rate-distortion performance. However, for some models the rate-distortion obtained by following such heuristics is not as close to the optimal as could be desired, as was demonstrated in the previous section. The problem stems from the fact that the relation between the distortion of the NURBS parameters and the NURBS surface is not well established. A scheme that could integrate some measure or approximation of the surface distortion could improve the rate-distortion efficiency.

Another problem that has not been shown here but which is of high importance is that the adjacency of NURBS patches is not taken into account. Often, complex surface shapes are obtained by making several NURBS patches be adjacent. However, the distortion introduced by the coding scheme can make the (originally) adjacent patches not adjacent anymore. As a consequence, visible cracks appear in the decoded model. Note that this is a general problem of NURBS. Even if no compression is involved, tesellation of the model into polygons can also introduce cracks. Nevertheless, a solution to this problem should be found that encodes the border of adjacent patches in an integrated way and avoids the introduction of cracks.

ACKNOWLEDGMENTS

The NURBS models used to evaluate the proposed scheme were created with the Alpha_1 geometric modeling system at the Computer Science Department, University of Utah.

REFERENCES

1. A. Khodakovsky, P. Schröder, and W. Sweldens, "Progressive geometry compression," in *Proc. of the International Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 2000*, pp. 271–278, New Orleans, LA, USA, July 2000.
2. J. S. Choi, Y. H. Kim, H.-J. Lee, I.-S. Park, M. H. Lee, and C. Ahn, "Geometry compression of 3-D mesh models using predictive two-stage quantization," *IEEE Trans. on Circuits and Systems for Video Technology*, **10**, pp. 312 – 322, Mar. 2000.
3. C. Touma and C. Gotsman, "Triangle mesh compression," in *Proc. of the 24th Conference on Graphics Interface (GI-98)*, pp. 26–34, San Francisco, CA, USA, June 1998.
4. ISO/IEC, *ISO/IEC 14496-2:1999: Information technology — Coding of audio-visual objects — Part 2: Visual*, Dec. 1999.
5. ISO/IEC, *ISO/IEC 14496-2:1999/Amd 1:2000: Visual extensions*, Aug. 2000.
6. M. Gopi and D. Manocha, "Simplifying spline models," *Computational Geometry*, **14**, pp. 67–90, Nov. 1999.
7. L. Piegl, "On NURBS: a survey," *IEEE Computer Graphics and Applications*, **11**, pp. 55–71, Jan. 1991.
8. H. Grahn, T. Volk, and H. J. Wolters, "Nurbs in VRML," in *VRML 2000*, Monterey, CA, USA, Feb. 2000.
9. ISO/IEC, *ISO/IEC 14772-1:1998 Information technology — Computer graphics and image processing — The Virtual Reality Modeling Language — Part 1: Functional specification and UTF-8 encoding*, June 1998.
10. G. Farin, *Curves and surfaces for computer aided geometric design*, Academic Press, London, 4th ed., 1996.
11. L. Piegl and W. Tiller, *The NURBS Book*, Springer-Verlag, Berlin, 2nd ed., 1997.

12. G. E. Farin, *NURBS: from projective geometry to practical use*, A K Peters Ltd., Natick, Massachusetts:, 2nd ed., 1999.
13. ISO/IEC JTC 1/SC 29/WG 1, *ISO/IEC FDIS 15444-1: Information technology — JPEG 2000 image coding system: Core coding system [WG 1 N 1890]*, Sept. 2000.
14. ISO/IEC, *ISO/IEC 11544:1993 Information technology — Coded representation of picture and audio information — Progressive bi-level image compression*, Mar. 1993.
15. W. B. Pennebaker and J. L. Mitchell, *JPEG: Still Image Data Compression Standard*, Van Nostrand Reinhold, New York, 1992.
16. G. G. Langdon, "Compression of multilevel signals," Patent Nr. 4,749,983, United States, June 1988.
17. P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: measuring error on simplified surfaces," *Computer Graphics Forum*, **17**, pp. 167–174, June 1998.