Compression of NURBS Surfaces with Error Evaluation

Yoshiyuki Furukawa
Department of Environmental and Ocean Engineering, The University of Tokyo
furu@nakl.t.u-tokyo.ac.jp

Hiroshi Masuda
Research into Artifacts, Center for Engineering, The University of Tokyo
masuda@race.u-tokyo.ac.jp

Abstract

NURBS surfaces are supported by most of 3D geometric modeling systems. In such an environment, geometry compression for NURBS data is very useful for collaborative work on networks. However, very few compression methods have been reported so far. This paper proposes a compression method for 3D models with NURBS surfaces. In our method, a NURBS surface is encoded using its boundaries and additional data. At first, an interpolating surface is calculated using the boundary curves extracted from the original surface. Then, the differences between the original surface and the interpolating one are calculated. Difference data are optionally used, and they are represented in two types: distances only, and distances and orientations. One type of difference data is used for encoding according to the surface smoothness and the accuracy required by receivers. Finally, the boundary curves and the difference data are compressed using the discrete cosine transform (DCT). On the receiver side, they are decoded in the reverse order, and the 3D model is reconstructed. In our method, the compression ratio and the accuracy of each surface can be controlled by using the types of difference data and the quality parameters of DCT. We implemented and evaluated our method by several example data. The result shows our method can easily control the trade-off between the compression ratio and accuracy, and achieve a good compression rates.

1 Introduction

The growth of 3D CAD systems and CG modelers makes the number of 3D data increase. 3D data is frequently exchanged through network for various applications such as collaborative works. In such an environment, geometry compression is required to exchange the huge and detailed data. Some geometry compression for polygonal models have been proposed, but most of 3D geomeric modeling systems support free-form surfaces as well as polygonal models. This is mainly due to the fact that CAD users in engineering fields strongly require NURBS data in order to keep interoperability with other application systems. Therefore, we believe that geometry compression for free-form surfaces is very useful in practical use, although very few compression methods for NURBS have been reported.

We propose a new compression method for 3D models with NURBS surfaces. In our approach, a NURBS surface is represented by its boundary curves and difference data for correction. A base surface is created by interpolating the boundary curves, and it is corrected by using difference data. Difference data is represented in multiple ways, and one of which is selected by taking the trade-off between the quality and the data size into account. Then, the boundary curves and the difference data are compressed using the discrete cosine transform (DCT).

The rest of this paper is structured as follows. In Section 2, we will present related works. Section 3 shows an overview of our compression method. We will describe, in Section 4, a method of compressing boundary curves and interpolating surfaces, and in Section 5, a method of representing difference data and compressing them. Section 6 presents experimental results. We will conclude in Section 7 with summary.

2 Related works

We will overview research works related to compression of polygonal mesh surfaces and parametric surfaces.

The research of Deering[1] is well known as a pioneer work of geometry compression. In his method, vertices of a triangular mesh are indexed, and duplication is significantly reduced by using a coordinate buffer. In addition, he reduced geometry data by representing vertex coordinates by discrete values represented in polar coordinate values. Taubin and Rossignac[2] encoded a triangular mesh model using triangle strips and vertex trees. Other compression methods for triangular mesh models have been reported by Gumhold and Straßer[3], Rossignac[4], and so on. Hoppe[5] proposed a method for progressive data transfer by applying polygonal simplification operations in reverse order. In this method, polygonal data are detailed gradually. Some other progressive methods [6]-[10] have been reported.

Although many compression methods for polygonal models have been proposed, reports for free-form surfaces are relatively few. DeVore, et.al.[11] transformed implicit surfaces using wavelet transformation, and encoded their coefficients. Masuda, et.al.[12] used a similar approach for parametric surfaces using DCT, which is included in JPEG compression. Both are lossy compression and reduce data

size by transforming coordinates for surfaces into the frequency domain, and quantizing coefficients so that a high frequency domain is measured coarsely. In the case of a single surface, there is an advantage in the elimination of high frequency domain, since it is difficult to perceive the difference from original surfaces. However, when smoothly connected multiple surfaces are compressed, the remarkable gap can be observed at the boundaries. This is called the Mach-band effect, which means that human eyes are very sensitive for discontinuity. Since each surface is quantized in frequency domain, the continuity is not always reserved in space domain. Wakita, et.al.[13] proposed an efficient compression method by approximating models as their own representation. This approach is convenient for browsing 3D data. However, it is not sufficient when data are used in various engineering purposes, because many engineering applications use very limited types of surfaces.

3 Approach to compression of NURBS Surfaces

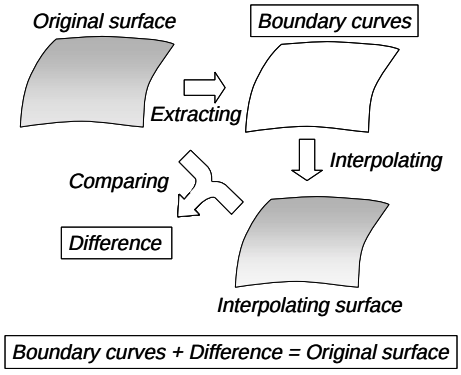


Figure 1. Concept of this method

Figure 1 shows our basic approach, which consists of two steps. In the first step, boundary curves of original parametric surfaces are extracted. They are encoded and transferred as a part of compressed data. In the receiver's side, transferred data are used to generate interpolating surfaces. In the second step, difference data between interpolating surfaces and the original surfaces are calculated. Two types of difference data are prepared; one maintains the difference only by distances, and the other does by distances and directions. We use DCT as the encoding method, which enables to control data size by DCT parameters.

In addition, types of difference data makes it possible to control the balance between accuracy and data size according to the environment or purpose of receivers. In our method, encoded surface data are classified in three cases as shown in Figure 2. One is to represent surfaces only by boundary curves. In this case, the original surface is approximated by the interpolation surface, and difference data is not used. The second case is a surface that is adjusted by displacements of control points in the normal direction. We

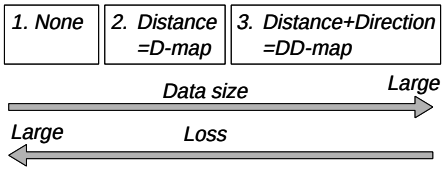


Figure 2. Three cases of correction using difference data

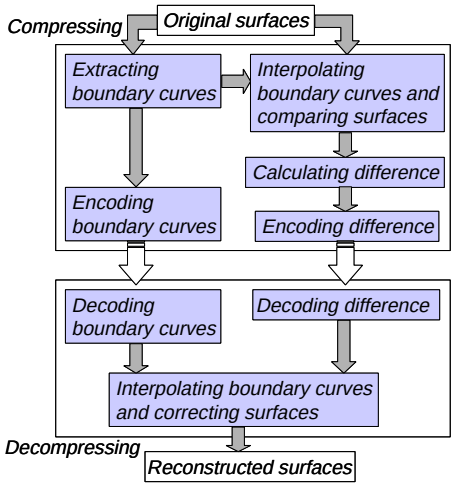


Figure 3. Flow of this method

call this displacement matrix as D-map. The third one is a surface to be adjusted by both displacements and directions of control points, which is called as DD-map in this paper.

Figure 3 shows processes for our compression method. It consists of a compression part and a decompression part. In compression part, the original data are converted into a combination of boundary curves and difference data. They are encoded using DCT compression and transmitted. The receiver decodes each of the data for generating interpolating surfaces and difference data, and then reconstructs NURBS surfaces by combining them.

4 Boundary curves and surface interpolation

4.1 Extraction of boundary curves

In this paper, we represent a NURBS surface $\mathbf{S}$ as

$$\mathbf{S}(u,v) = \frac{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{ij}\mathbf{P}_{ij}}{\sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)w_{ij}}, \qquad (1)$$

where $\mathbf{P}_{ij}$ is a control point, w_{ij} is a weight, p and q are degrees, $(n+1)$ and $(m+1)$ are numbers of control points, and N is a basis function of B-spline. Knot vectors are expressed as follows:

$$\begin{aligned} \mathbf{U} &= \{u_i | 0 \le u_i \le 1; i = 0, \cdots, n+p+1\}, \\ \mathbf{V} &= \{v_j | 0 \le v_j \le 1; j = 0, \cdots, m+q+1\}. \end{aligned} \qquad (2)$$

The following is used to represent a NURBS curve.

$$\mathbf{C}(u) = \frac{\sum_{i=0}^{n} N_{i,p}(u) w_i \mathbf{P}_i}{\sum_{i=0}^{n} N_{i,p}(u) w_i} \tag{3}$$

When a surface $\mathbf{S}$ is given, four boundary curves $\mathbf{C}_i$(i=$0, \cdots, 3$) are extracted. They are described as follows:

$$\begin{aligned} \mathbf{C}_0(u) = \mathbf{S}(u,0)\ ,\ \mathbf{C}_1(v) = \mathbf{S}(0,v)\ , \\ \mathbf{C}_2(u) = \mathbf{S}(u,1)\ ,\ \mathbf{C}_3(v) = \mathbf{S}(1,v)\ . \end{aligned} \tag{4}$$

4.2 Compression of a boundary curve

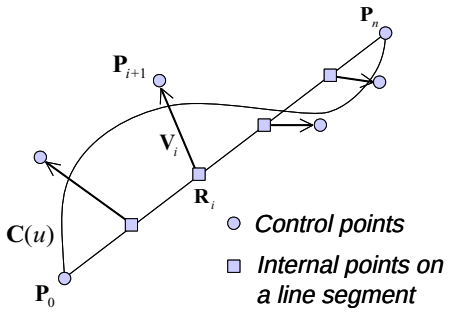


Figure 4. Representation of a boundary curve for compression

In the process of curve compression, each control point is measured by difference from an internal point of a line segment, as shown in Figure 4.

When a boundary curve $\mathbf{C}$ is given and corresponding control points are $\mathbf{P}_i$ (i=$0, \cdots, n$), an internal point $\mathbf{R}_i$ (i=$0, \cdots, n$-2) is obtained by equally dividing a line segment using:

$$\mathbf{R}_i = \left(1 - \frac{i+1}{n}\right)\mathbf{P}_0 + \frac{i+1}{n}\mathbf{P}_n\ . \tag{5}$$

We represent a sequence of difference vectors as

$$\mathbf{V}_i \equiv (V_i^x, V_i^y, V_i^z) = \mathbf{P}_{i+1} - \mathbf{R}_i\ . \tag{6}$$

Similarly, we obtain a sequence of weights as

$$V_i^w = 1 - w_{i+1}\ . \tag{7}$$

Then, we apply the discrete cosine transform (DCT) to all sequences of V_i^α (α=x, y, z, w;i=$0, \cdots, n$-2), and convert them into coefficients in the frequency domain. Coefficients are calculated by the following equation. This conversion is reversible and lossless.

$$D_k^\alpha = \sqrt{\frac{2}{n-2}} C_k \sum_{i=0}^{n-2} V_i^\alpha \cos\frac{(2i+1)k\pi}{2(n-2)}\ , \tag{8}$$

$$C_k = \begin{cases} \frac{1}{\sqrt{2}} & (k=0) \\ 1 & (k \neq 0) \end{cases}\ . \tag{9}$$

Then, coefficients are quantized and rounded into a bits. The bit legth a is selected so that distances from the original control points are less than tolerances specified by a user.

$$\begin{aligned} \bar{D}_k^\alpha &= Round\left(\frac{D_k^\alpha}{Q_k}\right)\ , \\ Q_k &= \frac{(1+k)D_0^\alpha}{2^a}\ . \end{aligned} \tag{10}$$

4.3 Surface interpolation

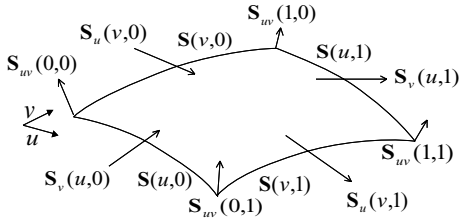


Figure 5. Conditions of interpolation

For generating a surface by interpolating boundary curves, we use bicubic blend Coons surfaces. A bicubic blend Coons surface is generated from boundary curves $\mathbf{S}(u,0)$, $\mathbf{S}(0,v)$, $\mathbf{S}(u,1)$, $\mathbf{S}(1,v)$, cross-boundary derivatives $\mathbf{S}_v(u,0)$, $\mathbf{S}_u(0,v)$, $\mathbf{S}_v(u,1)$, $\mathbf{S}_u(1,v)$, and twist vectors $\mathbf{S}_{uv}(0,0)$, $\mathbf{S}_{uv}(0,1)$, $\mathbf{S}_{uv}(1,0)$, $\mathbf{S}_{uv}(1,1)$. Figure 5 shows these conditions.

For applying the method, cross-boundary derivatives have to be determined. We approximate them as cubic Bezíer curves. By determining twist vectors from the network of boundary curves, we can calculate each cross-boundary derivative.

Therefore, an interpolating surface is calculated as

$$\mathbf{S}^I(u,v) = \mathbf{S}_1^I(u,v) + \mathbf{S}_2^I(u,v) - \mathbf{S}_3^I(u,v)\ , \tag{11}$$

where $\mathbf{S}_1^I$ and $\mathbf{S}_2^I$ are cubic Bezíer blend surfaces in the v and u direction respectively, and $\mathbf{S}_3^I$ is a bicubic Bezíer surface.

5 Correction of interpolating surfaces

5.1 Representaions of difference data

The difference data is calculated as directions and distances between the interpolating surface $\mathbf{S}^I$ and the original surface $\mathbf{S}$, as shown in Figure 6. These are represented as a matrix form, which is reffered to as a DD-map. A DD-map V_{ij}^α (α=x, y, z, w;i=$0, \cdots, n$-2;j=$0, \cdots, m$-2) consists of control points and weights as:

$$\mathbf{V}_{ij} \equiv (V_{ij}^x, V_{ij}^y, V_{ij}^z) = \mathbf{P}_{(i+1)(j+1)} - \mathbf{P}^I_{(i+1)(j+1)}\ , \tag{12}$$

$$V_{ij}^w = w_{(i+1)(j+1)} - w^I_{(i+1)(j+1)}\ . \tag{13}$$

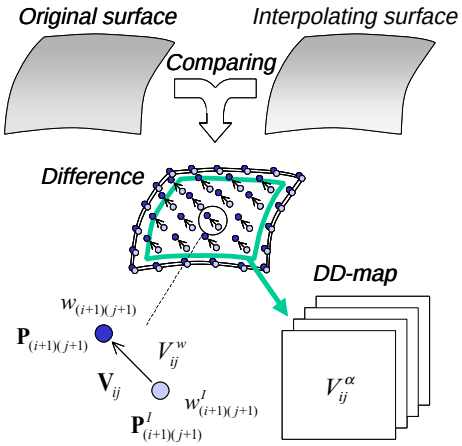


Figure 6. Definition of DD-map

The other type of difference data is a D-map, which maintains only distances that are measured from known positions and directions. As shown in Figure 7, each position is determined as a grid point in a uv-parameter space, and the direction as the normal vector at the point. This matrix form is reffered to as a D-map. Since a D-map cannot precisely represent the original surface, we calculate the optimal set of distance values, which minimizes the sum of square distances.

When the optimal surface is represented as

$$\mathbf{S}^A(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)\mathbf{P}_{ij}^A \ , \tag{14}$$

each control point can be represented as

$$\mathbf{P}_{ij}^A = \mathbf{P}_{ij}^I + d_{mi+j}\mathbf{n}_{ij} \ , \tag{15}$$

where d_{mi+j} is a distance between correspondent control points and $\mathbf{n}_{ij}$ can be represented as

$$\mathbf{n}_{ij} = \frac{\mathbf{S}_u^I(\hat{u}_i,\hat{v}_j)\times\mathbf{S}_v^I(\hat{u}_i,\hat{v}_j)}{|\mathbf{S}_u^I(\hat{u}_i,\hat{v}_j)\times\mathbf{S}_v^I(\hat{u}_i,\hat{v}_j)|} \ . \tag{16}$$

Grid points in uv-space are determined using knot vectors $\mathbf{U}$, $\mathbf{V}$ of $\mathbf{S}^I$ as

$$\begin{aligned} \hat{u}_i &= \tfrac{1}{p}\sum_{k=i+1}^{p} u_k \ , \\ \hat{v}_j &= \tfrac{1}{q}\sum_{l=j+1}^{q} v_l \ . \end{aligned} \tag{17}$$

A D-map is obtained by minimizing the following objective function.

$$E = \sum_{k=0}^{n}\sum_{l=0}^{m} |\mathbf{S}^A(\hat{u}_k,\hat{v}_l) - \mathbf{S}(\hat{u}_k,\hat{v}_l)|^2 \tag{18}$$

By calculating $\frac{\partial E}{\partial d_{mi+j}} = 0$ (i=0,$\cdots$,n;j=0,$\cdots$,m), the following equation is obtained.

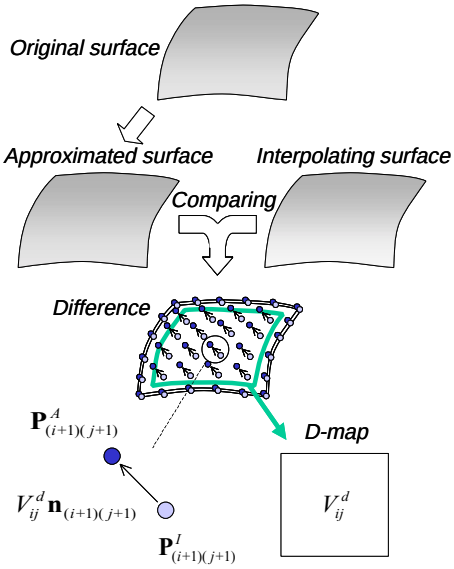


Figure 7. Definition of D-map

$$\begin{aligned} &\sum_{k=0}^{n}\sum_{l=0}^{m}\sum_{s=0}^{n}\sum_{t=0}^{m} F_{ijstkl}d_{ms+t}\mathbf{n}_{ij}\cdot\mathbf{n}_{st} \\ &\qquad = \sum_{k=0}^{n}\sum_{l=0}^{m} N_{i,p}(\hat{u}_k)N_{j,q}(\hat{v}_l)\mathbf{G}_{kl}\cdot\mathbf{n}_{st} \ , \end{aligned} \tag{19}$$

where

$$\begin{aligned} F_{ijstkl} &= N_{i,p}(\hat{u}_k)N_{j,q}(\hat{v}_l)N_{s,p}(\hat{u}_k)N_{t,q}(\hat{v}_l) \ , \\ \mathbf{G}_{kl} &= \mathbf{S}(\hat{u}_k,\hat{v}_l) - \mathbf{S}^I(\hat{u}_k,\hat{v}_l) \ . \end{aligned} \tag{20}$$

Then, we can obtain the distances as

$$\mathbf{d} = \mathbf{A}^{-1}\mathbf{B} \ , \tag{21}$$

by using the following matrices.

$$a_{mi+j,ms+t} = \sum_{k=0}^{n}\sum_{l=0}^{m} F_{ijstkl}\mathbf{n}_{ij}\cdot\mathbf{n}_{st}$$

$$\mathbf{A} = \begin{bmatrix} a_{0,0} & \cdots & a_{0,ms+t} & \cdots & a_{0,nm} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{mi+j,0} & \cdots & a_{mi+j,ms+t} & \cdots & a_{mi+j,nm} \\ \vdots & \ddots & \vdots & \ddots & \vdots \\ a_{nm,0} & \cdots & a_{nm,ms+t} & \cdots & a_{nm,nm} \end{bmatrix} \tag{22}$$

$$\mathbf{d} = [d_0\cdots d_{ms+t}\cdots d_{nm}]^T \tag{23}$$

$$\begin{aligned} b_{mi+j} &= \sum_{k=0}^{n}\sum_{l=0}^{m} N_{i,p}(\hat{u}_k)N_{j,q}(\hat{v}_l)\mathbf{G}_{kl}\cdot\mathbf{n}_{st} \\ \mathbf{B} &= [b_0\cdots b_{mi+j}\cdots b_{nm}]^T \end{aligned} \tag{24}$$

Finally, a D-map V_{ij}^d (i=0,$\cdots$,n-2;j=0,$\cdots$,m-2) is obtained as

$$V_{ij}^d = d_{m(i+1)+j+1} \ . \tag{25}$$

5.2 Coding correction data

D-maps and DD-maps are represented as matrices, thus they can be compressed by two dimensional DCT. DCT transforms matrices V_{ij}^{α} (α=x, y, z, w, d;i=$0, \cdots, n$-2;j=$0, \cdots, m$-2) to D_{kl}^{α} (k=$0, \cdots, n$-2;l=$0, \cdots, m$-2) as follows:

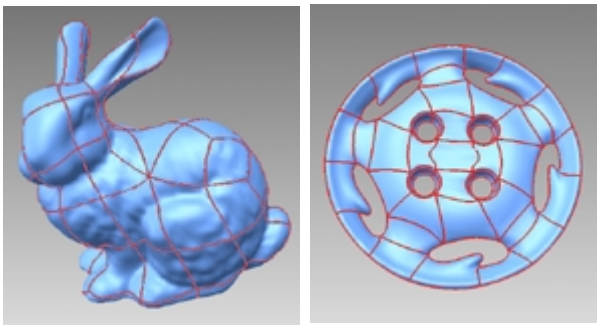
$$D_{kl}^{\alpha} = \sqrt{\frac{2}{n-2}}\sqrt{\frac{2}{m-2}} C_k C_l \sum_{i=0}^{n-2}\sum_{j=0}^{m-2} \left\{ V_{ij}^{\alpha} \cos\frac{(2i+1)k\pi}{2(n-2)} \cos\frac{(2j+1)l\pi}{2(m-2)} \right\}, \tag{26}$$

where C_k and C_l are using (9). Then, D_{kl}^{α} is quantized into a bits as follows:

$$\bar{D}_{kl}^{\alpha} = Round\left(\frac{D_{kl}^{\alpha}}{Q_{kl}}\right), \quad Q_{kl} = \frac{(1+k+l)D_{00}^{\alpha}}{2^a}. \tag{27}$$

Finally, curve data and correction data, which are described as a set of integers, are represented as text data, and encoded by a text compression tool.

6 Experimental results



(a) bunny (b) wheel

Figure 8. Example data

We implemented our compression method and evaluated using two example data shown in Figure 8. Both data consist of 63 bicubic NURBS surfaces. In our experiment, the surfaces were compressed using several tolerances. The tolerances were given as absolute errors measured in 3D Euclidean space. Compression rates were calculated as ratios of the encoded data size to the original ASCII data size. In compression processes, types of correction data and values of quantization parameter were selected to minimize data size.

Experimental results are shown in Table 1 and Table 2. In both results, good compression rates are achieved as large tolerances are given. Figure 9 shows renderings of each reconstructed model.

7 Conclusion

This paper proposed a new compression method for exchanging 3D data with NURBS surfaces through network. In the method, a NURBS surface is represented as boundary curves and difference data. The boundary curves and the difference data are encoded by DCT compression. Additionally, We defined three types of difference data and a quantization parameter of DCT compression. By implementing and evaluating our method, we achived good compression rates and to control the quality and the data size.

References

[1] M. Deering, "Geometry compression", *Proceedings of ACM SIGGRAPH'95*, 1995, pp. 13-20.

[2] G. Taubin, J. Rossignac, "Geometry compression through topological surgery", *ACM Transaction of Graphics*, Vol. 17, No. 2, 1998, pp. 84-115.

[3] S. Gumhold, W. Straßer, "Real time compression of triangle mesh connectivity", *Proceedings of ACM SIGGRAPH'98*, 1998, pp. 133-140.

[4] J. Rossignac, "Edgebreaker: Connectivity compression for triangle meshes", *IEEE Transactions on Visualization and Computer Graphics*, Vol. 5, No. 1, 2000, pp. 47-61.

[5] H. Hoppe, "Progressive meshes", *Proceedings of ACM SIGGRAPH'96*, 1996, pp. 99-108.

[6] J. Li, C.C. Kuo, "Progressive compression of 3D graphic models", *Proceedings of IEEE ICMCS'97*, 1997, pp. 135-142.

[7] G. Taubin, A. Guéziec, W. Horn, F. Lazarus, "Progressive forest split compression", *Proceedings of ACM SIGGRAPH'98*, 1998, pp. 123-132.

[8] R. Pajarola, J. Rossignac, "Compressed progressive meshes", *IEEE Transactions on Visualization and Computer Graphics*, Vol. 6, No. 1, 2000, pp. 79-93.

[9] A. Khodakovsky, P. Schröder, W. Sweldens, "Progressive geometry compression", *Proceedings of ACM SIGGRAPH2000* , 2000, pp. 271-278.

[10] R. Balan, G. Taubin, "3D mesh geometry filtering algorithms for progressive transmission schemes", *Computer-Aided Design*, Vol. 32, 2000, pp. 825-846.

[11] R.A. DeVore, J. Bjorn, B.J. Lucier, "Surface compression", *Computer Aided Geometric Design*, Vol. 9, 1992, pp. 219-239.

[12] H. Masuda, R. Ohbuchi, M. Aono, "Frequency domain compression and progressive transmission of parametric surfaces (in Japanese)", *Journal of IPSJ*, 1999, pp. 1188-1195.

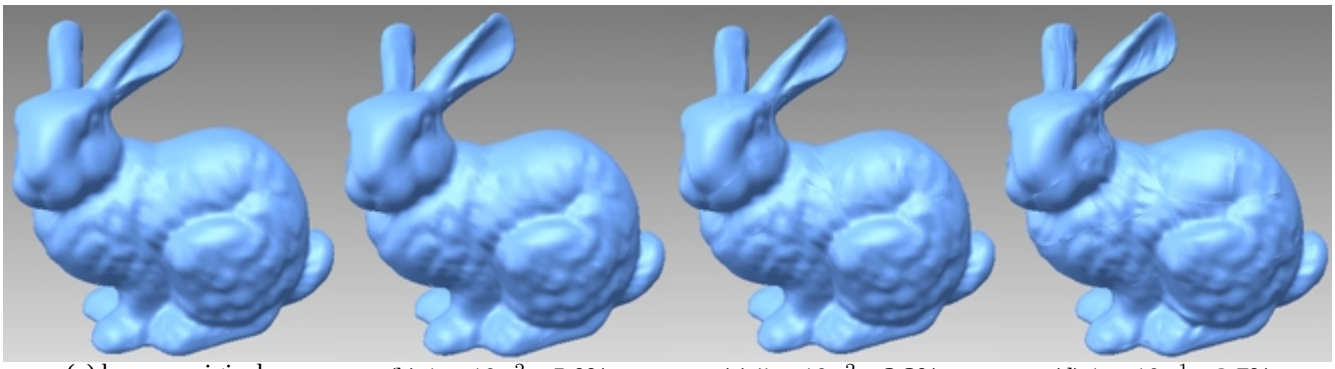
[13] A. Wakita, M. Yajima, T. Harada, H. Toriya,H. Chiyokura, "XVL: A compact and qualified 3D representation with lattice mesh and surface for the internet", *Proceedings of ACM VRML2000*, 2000, pp. 45-51.

Table 1. Result: bunny

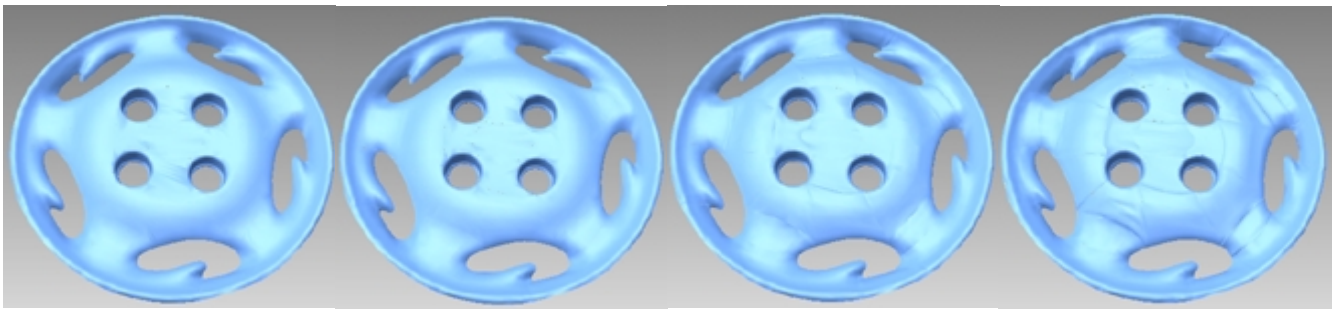
Tolerance	Data Size ASCII	Encoded	Compression rate	Number of each type None	D-map	DD-map
(Original)	553.1kB	163.8kB	29.6%			
1×10^{-4}	125.7kB	52.3kB	9.5%	0	0	63
1×10^{-3}	103.0kB	40.2kB	7.3%	0	0	63
1×10^{-2}	79.8kB	27.7kB	5.0%	0	4	59
5×10^{-2}	56.3kB	18.5kB	3.3%	1	25	37
1×10^{-1}	43.9kB	14.7kB	2.7%	5	34	24

Table 2. Result: wheel

Tolerance	Data Size ASCII	Encoded	Compression rate	Number of each type None	D-map	DD-map
(Original)	598.5kB	196.0kB	32.8%			
1×10^{-3}	129.0kB	53.0kB	8.9%	0	0	63
1×10^{-2}	105.1kB	40.0kB	6.7%	0	0	63
1×10^{-1}	71.2kB	24.3kB	4.0%	0	25	38
5×10^{-1}	44.0kB	15.1kB	2.5%	20	29	24
1×10^{-0}	39.0kB	13.5kB	2.3%	33	23	7



(a) bunny: original (b) 1×10^{-2} 5.0% (c) 5×10^{-2} 3.3% (d) 1×10^{-1} 2.7%



(e) wheel: original (f) 1×10^{-1} 4.0% (g) 5×10^{-1} 2.5% (h) 1×10^{0} 2.3%

Figure 9. Reconstructed models: tolerances and compression rates