EURASIP Journal on Applied Signal Processing 2004:14, 2142–2152
© 2004 Hindawi Publishing Corporation

Watermarking Algorithms for 3D NURBS Graphic Data

Jae Jun Lee
Institute of New Media and Communications (INMC), School of Electrical Engineering, Seoul National University, Seoul 151-744, Korea
Email: jazzon@ispl.snu.ac.kr

Nam Ik Cho
Institute of New Media and Communications (INMC), School of Electrical Engineering, Seoul National University, Seoul 151-744, Korea
Email: nicho@snu.ac.kr

Sang Uk Lee
Institute of New Media and Communications (INMC), School of Electrical Engineering, Seoul National University, Seoul 151-744, Korea
Email: sanguk@ipl.snu.ac.kr

Received 14 March 2003; Revised 10 June 2004

Two watermarking algorithms for 3D nonuniform rational B-spline (NURBS) graphic data are proposed: one is appropriate for the steganography, and the other for watermarking. Instead of directly embedding data into the parameters of NURBS, the proposed algorithms embed data into the 2D virtual images extracted by parameter sampling of 3D model. As a result, the proposed steganography algorithm can embed information into more places of the surface than the conventional algorithm, while preserving the data size of the model. Also, any existing 2D watermarking technique can be used for the watermarking of 3D NURBS surfaces. From the experiment, it is found that the algorithm for the watermarking is robust to the attacks on weights, control points, and knots. It is also found to be robust to the remodeling of NURBS models.

Keywords and phrases: watermark, 3D image, nonuniform rational B-spline (NURBS), steganography, data hiding.

1. INTRODUCTION

With the development of computing environments, 3D models became widely used and are being produced in great numbers. 3D models are usually represented by mesh, nonuniform rational B-spline (NURBS), or voxel. Among these models, mesh is quite widely used because many studies on the mesh have already been performed, and also because the scanned 3D data are naturally the sampling points of surfaces. However, the mesh representation has drawbacks in that it requires large amount of data, and it cannot represent mathematically rigorous curves and surfaces. Unlike mesh, the NURBS describes 3D models by using mathematical formula. The data size for the NURBS is remarkably smaller than that for the mesh because the surface can be represented by only a few parameters. Also, the NURBS is smooth in nature so that the smoothness of NURBS is restricted only by hardware resolution. Hence, the NURBS is used in CAD and other areas which need high precision, and it is also used in animation because the motion of an object can be realized by successively adjusting some of the parameters.

Although the amount of 3D multimedia data is dramatically increasing, there has not been much discussion on the watermarking of 3D models, especially on the 3D NURBS models. For example, Ohbuchi et al. proposed an algorithm that embeds information into the knot equations by knot reparameterization [1]. This algorithm has merits that the resulting NURBS model has exactly the same shape as the original one, and that the number of knots and control points are also unchanged. But the embedded information can be detected only when one has the original model because it is embedded into the coefficients of the function that reparameterizes the original knot vector. Moreover, the maximum number of embedded data is only three, which is the degree of freedom of bilinear function, and a slight modification of the knots and control points makes it impossible to detect the information.

In this paper, we propose two watermarking algorithms for 3D NURBS, one is suitable for the steganography (for secret communication between trusting parties) and the other for the robust watermarking. In the proposed algorithm, a virtual NURBS model is first generated from the original

one. Instead of embedding information into the parameters of NURBS data as in the existing algorithm, the proposed algorithms extract several 2D images from the 3D virtual model and apply the 2D watermarking methods.

In the steganography algorithm, 3D virtual model is first sampled in each of u and v directions, where u and v are parameters of NURBS. That is, a sequence of $\{u, v\}$ is generated, where the number of elements is limited to be less than that of control points. Then, three 2D virtual images are extracted, the pixels of which are the distances from the sample points to the x, y, and z plane, respectively. The watermark is embedded into these 2D images, which leads to the modification of control points of NURBS. As a result, the original model is changed by the watermark data as much as the quantity of embedded data. But the data size of the NURBS model is preserved because there is no change in the number of knots and control points. For the extraction of embedded information, modified virtual sample points are first acquired by the matrix operation of basis functions in accordance with the $\{u, v\}$ sequence. Even if the third party has the original NURBS model, the embedded information cannot be acquired without $\{u, v\}$ sequence as a key, which is a good property for the steganography.

The second algorithm is suitable for the robust watermarking. This algorithm also samples the 3D virtual model. But the difference from the steganography algorithm is that the number of sampled points is not limited by the number of control points of the original NURBS model. Instead, the sequence $\{u, v\}$ is chosen so that the sampling interval in the physical space is kept constant. This makes the model robust against the attacks on knot vectors, such as knot insertion, removal, and so forth. The procedure of making 2D virtual images is the same as the steganography algorithm. Then, the watermarking algorithms for 2D images are applied to these virtual images, and a new NURBS model is made by the approximation of watermarked sample points. The watermarks in the coordinate of each sample point are distorted within the error bound by approximation. But such distortion can be controlled by the strength of embedded watermarks and the magnitude of error bound. Since the points are not sampled in the physical space (x-,y-,z-coordinate) but in the parametric space (u-,v-coordinate), the proposed algorithm for watermarking is also found to be robust against the attacks on the control points that determine the model's transition, rotation, scaling, and projection. A preliminary version of this paper has appeared in [2].

This paper is organized as follows. In Section 2, the NURBS model is briefly reviewed. In Section 3, the watermarking applications considered are introduced and the proposed algorithms are described. Implementation of the algorithms and experimental results are presented in Section 4, followed by Section 5 presenting some conclusions.

2. NURBS CURVES AND SURFACES

For the explanation of the proposed algorithms in later sections, we first briefly review the definition and notation of NURBS. More details on the NURBS can be found in [3, 4].

A NURBS surface $\mathbf{S}(u, v) = (x(u, v), y(u, v), z(u, v))$ of degree p in the u direction and degree q in the v direction is defined by a bivariate function of the form

$$\mathbf{S}(u, v) = \sum_{i=0}^{n} \sum_{j=0}^{m} R_{i,j}(u, v)\mathbf{P}_{i,j}, \tag{1}$$

where

$$R_{i,j}(u, v) = \frac{N_{i,p}(u)N_{j,q}(v)w_{i,j}}{\sum_{k=0}^{n} \sum_{l=0}^{m} N_{k,p}(u)N_{l,q}(v)w_{k,l}}. \tag{2}$$

In (1), $\{\mathbf{P}_{ij}\}$ form a bidirectional control net, and $\{w_i\}$ in (2) are the weights. $\{N_{i,p}(u)\}$ are the ith B-spline basis function of degree p, which can be defined recursively as

$$\begin{aligned} N_{i,0}(u) &= \begin{cases} 1 & \text{if } u_i \le u < u_{i+1}, \\ 0 & \text{otherwise,} \end{cases} \\ N_{i,p}(u) &= \frac{u - u_i}{u_{i+p} - ui} N_{i,p-1}(u) \\ &\quad + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u) \end{aligned} \tag{3}$$

on a nonperiodic and nonuniform knot vector

$$U = \{\underbrace{a, \ldots, a}_{p+1}, u_{p+1}, \ldots, u_{r-p-1}, \underbrace{b, \ldots, b}_{p+1}\}, \tag{4}$$

where $u_i \le u_{i+1}$, $i = 0, \ldots, r-1$, and $r = n + p + 1$. $\{N_{j,q}(v)\}$ are analogously defined as $\{N_{i,p}(u)\}$.

The equation for the NURBS curve can be derived from the surface of (1) by fixing one of the variables u or v. To be precise, isoparametric curves $\mathbf{C}_{u_0}(v)$ and $\mathbf{C}_{v_0}(u)$ of NURBS surface $\mathbf{S}(u, v)$ trace the trajectory curve for the fixed u_0 and v_0, respectively, and they intersect at a surface point $\mathbf{S}(u_0, v_0)$. For the simplicity of explanation, all the weights are set to one. Then $\mathbf{C}_{u_0}(v)$ is defined as

$$\begin{aligned} \mathbf{C}_{u_0}(v) &= \mathbf{S}(u_0, v)\,|_{w_{i,j}=1, \forall i,j} \\ &= \sum_{j=0}^{m} N_{j,q}(v) \left(\sum_{i=0}^{n} N_{i,p}(u_0)\mathbf{P}_{i,j} \right) \\ &= \sum_{j=0}^{m} N_{j,q}(v)\mathbf{Q}_j(u_0), \end{aligned} \tag{5}$$

where

$$\mathbf{Q}_j(u_0) = \sum_{i=0}^{n} N_{i,p}(u_0)\mathbf{P}_{i,j}. \tag{6}$$

Analogously

$$\mathbf{C}_{v_0}(u) = \sum_{i=0}^{n} N_{i,p}(u)\mathbf{Q}_i(v_0), \tag{7}$$

where

$$\mathbf{Q}_i(v_0) = \sum_{j=0}^{m} N_{j,q}(v_0)\mathbf{P}_{i,j}. \tag{8}$$

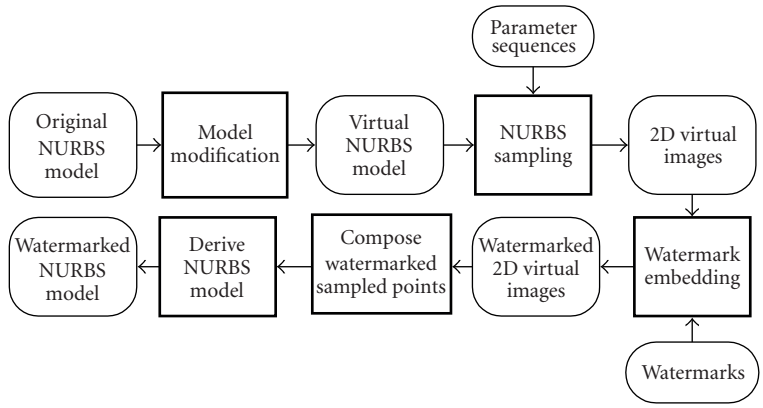

Figure 1: Block diagram of watermarking process using model sampling.

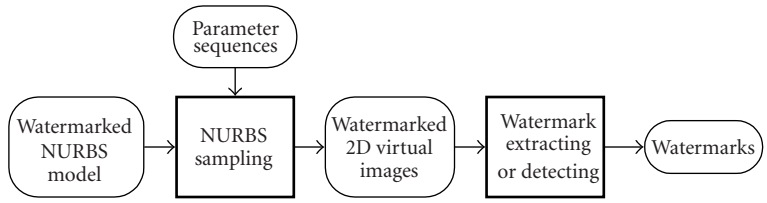

Figure 2: Block diagram of extracting or detecting process.

3. WATERMARKING NURBS CURVES AND SURFACES BY MODEL SAMPLING

3.1. Overview of the algorithm

There are many categories in watermarking, such as steganography, watermarking, data hiding, fingerprinting, and so on. In this paper, we focus only on two categories of watermarking, which are steganography and watermarking. The definitions are slightly different depending on the literature, and in this paper we refer to [5]. In general, steganography refers to the technique that allows secret communication, usually by embedding or hiding secret information in the other unsuspected data. Steganographic methods generally rely on the assumption that the existence of the covert communication is unknown to the third parties and are mainly used in secret point-to-point communication between the trusting parties. As a result, the steganographic methods are usually not robust, that is, the hidden information cannot be recovered after some data manipulation. Watermarking, as opposed to the steganography, has the additional notion of robustness against attacks. Even if the existence of the hidden information is known and the algorithmic principle of the watermarking method is public, it is difficult for an attacker to destroy the embedded watermark. Steganography and watermarking are thus more complementary rather than competitive approaches.

The main idea of the proposed algorithms is to extract some informative 2D virtual images from the surface of 3D objects with the (u, v) sampling sequence as a key, and apply any existing 2D watermarking algorithm. In order to extract 2D virtual images from the 3D NURBS model, we first make a virtual NURBS model $\mathbf{S}_{\text{virtual}}(u, v)$ (which will be explained in Section 3.2). Then, three 2D virtual images $x(u, v)$, $y(u, v)$, and $z(u, v)$ are extracted from the $\mathbf{S}_{\text{virtual}}(u, v)$, where x, y, and z represent the coordinates in the physical space. 2D watermarking techniques are applied to these images, and then the pixel values of watermarked images form new coordinates of the watermarked surface. The outline of the watermarking process is described in Figure 1.

Extracting or detecting watermarks is the reverse of embedding process. Three 2D virtual images are extracted by the same parameter sequences $\{u'_\alpha\}$, $\{v'_\beta\}$ used in the watermarking process. Then, according to the watermarking application (steganography or watermarking), we extract watermarks and check whether they are the same as the one that we embedded, or detect watermarks and check if the watermarks exist. The outline of extracting or detecting process is described in Figure 2.

3.2. Algorithm for steganography

In this subsection, we present the algorithm for the steganography. In this paper, we use the bracket $[\cdot]$ to denote a vector or a matrix, and the brace $\{\cdot\}$ to denote a set. The purpose of the proposed algorithm is to embed some data into the surface $\mathbf{S}(u, v)$ defined in (1). But instead of using $\mathbf{S}(u, v)$, we use the virtual surface $\mathbf{A}(u, v)$ defined as

$$\begin{aligned}\mathbf{A}(u,v) &\triangleq \mathbf{S}(u,v)|_{w_{i,j}=1,\,\forall i,j} \\ &= \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)\mathbf{P}_{i,j},\end{aligned} \tag{9}$$

where $\{\mathbf{P}_{i,j}\}$ are the same control points as the original model. That is, we use a surface with all the weights set to 1 because it can then be represented in a matrix form. This also gives robustness to the attacks on weights, as will be discussed later. Note that $\mathbf{S}(u,v)$ can be easily reconstructed from $\mathbf{A}(u,v)$ by giving the weights. The surface $\mathbf{A}(u,v)$ can be written in the matrix form as

$$\mathbf{A}(u,v) = [N_{e,p}(u)]^T [\mathbf{P}_{e,f}][N_{f,q}(v)], \tag{10}$$

where $0 \le e \le n$ and $0 \le f \le m$. Note that $[N_{e,p}(u)]^T$ is a $1 \times (n+1)$ row vector, $[\mathbf{P}_{e,f}]$ is an $(n+1) \times (m+1)$ matrix of control points, and $[N_{f,q}(v)]$ is an $(m+1) \times 1$ column vector. If we sample $(n+1) \times (m+1)$ points from the surface of (10) by using $n+1$ parameters in the range of u-directional knot vector U, and $m+1$ in the v-directional knot vector V, we can make a matrix of samples $[\mathbf{A}(u'_\alpha, v'_\beta)]$ as

$$[\mathbf{A}(u'_\alpha, v'_\beta)] = \begin{pmatrix} \mathbf{A}(u'_0, v'_0) & \mathbf{A}(u'_0, v'_1) & \cdots & \mathbf{A}(u'_0, v'_m) \\ \mathbf{A}(u'_1, v'_0) & \mathbf{A}(u'_1, v'_1) & \cdots & \mathbf{A}(u'_1, v'_m) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{A}(u'_n, v'_0) & \mathbf{A}(u'_n, v'_1) & \cdots & \mathbf{A}(u'_n, v'_m) \end{pmatrix} \tag{11}$$

for $\alpha = 0, \ldots, n$ and $\beta = 0, \ldots, m$, where $\{u'_\alpha\}$, $\{v'_\beta\}$ are user-chosen parameter sequences in the direction of u and v. From (10), $\mathbf{A}(u'_\alpha, v'_\beta)$ can be decomposed as

$$[\mathbf{A}(u'_\alpha, v'_\beta)] = [\mathbf{N}_{e,p}(u'_\alpha)]^T [\mathbf{P}_{e,f}][\mathbf{N}_{f,q}(v'_\beta)], \tag{12}$$

where $[\mathbf{N}_{e,p}(u'_\alpha)] = [N_{e,p}(u'_0)\ N_{e,p}(u'_1)\ \cdots\ N_{e,p}(u'_n)]$, which is an $(n+1) \times (n+1)$ square matrix, and $[\mathbf{N}_{f,q}(v'_\beta)] = [N_{f,q}(v'_0)\ N_{f,q}(v'_1)\ \cdots\ N_{f,q}(v'_m)]$, which is an $(m+1) \times (m+1)$ square matrix.

In the watermarking algorithm, the watermark is embedded into the virtual surface $\mathbf{A}(u,v)$ instead of $\mathbf{S}(u,v)$, as will be explained in Section 3.3. In the case of steganography, we need to define another virtual surface $\mathbf{B}(u,v)$, which is of course closely related with $\mathbf{A}(u,v)$ as

$$\begin{aligned}[\mathbf{B}(u'_\alpha, v'_\beta)] = {} & [\mathbf{A}(u'_\alpha, v'_\beta)] + \gamma [\mathbf{P}_{e,f}][\mathbf{N}_{f,q}(v'_\beta)] \\ & + \delta [\mathbf{N}_{e,p}(u'_\alpha)]^T [\mathbf{P}_{e,f}] + \gamma\delta [\mathbf{P}_{e,f}],\end{aligned} \tag{13}$$

where γ and δ are real numbers. This can also be written as

$$\begin{aligned}&[\mathbf{B}(u'_\alpha, v'_\beta)] \\ &\quad = [[\mathbf{N}_{e,p}(u'_\alpha)] + \gamma \mathbf{I}_{n+1}]^T [\mathbf{P}_{e,f}][[\mathbf{N}_{f,q}(v'_\beta)] + \delta \mathbf{I}_{m+1}].\end{aligned} \tag{14}$$

The reason for defining another virtual surface $\mathbf{B}(u,v)$ as shown above is that the matrices in (14) can be made invertible by appropriately selecting γ and δ, whereas $[\mathbf{A}(u'_\alpha, v'_\beta)]$ in (12) is not generally invertible. The invertibility is necessary, in the case of steganography, for embedding and extracting information.

For simplicity, we denote $\mathbf{B}(u'_\alpha, v'_\beta)$ as $\mathbf{B}_{\alpha,\beta}$. Note that $\mathbf{B}_{\alpha,\beta}$ for the given (u'_α, v'_β) is a point in the 3D space, and thus $\{\mathbf{B}_{\alpha,\beta}\}$ can be considered as three 2D images with the variables u and v. More specifically, $\mathbf{B}_{\alpha,\beta} = (x(u'_\alpha, v'_\beta), y(u'_\alpha, v'_\beta), z(u'_\alpha, v'_\beta))$, where $x(u'_\alpha, v'_\beta)$, $y(u'_\alpha, v'_\beta)$, and $z(u'_\alpha, v'_\beta)$ are 2D images. By embedding watermark data $\{\mathbf{X}_{\alpha,\beta}\}$ into each image new images $\{\hat{\mathbf{B}}_{\alpha,\beta}\}$ are generated as

$$\hat{\mathbf{B}}_{\alpha,\beta} = \mathbf{B}_{\alpha,\beta} + \mathbf{X}_{\alpha,\beta}. \tag{15}$$

Assume that the modified model for $\hat{\mathbf{B}}(u,v)$ and the embedded data model for $\mathbf{X}(u,v)$ are the same as the original virtual one, that is, $\mathbf{B}(u,v)$, except for the coordinates of control points. Then, they can also be represented in the form of (14) as

$$\begin{aligned}[\hat{\mathbf{B}}_{\alpha,\beta}] &= [\hat{\mathbf{B}}(u'_\alpha, v'_\beta)] \\ &= [[\mathbf{N}_{e,p}(u'_\alpha)] + \gamma \mathbf{I}_{n+1}]^T [\hat{\mathbf{P}}_{e,f}][[\mathbf{N}_{f,q}(v'_\beta)] + \delta \mathbf{I}_{m+1}],\end{aligned} \tag{16}$$

$$\begin{aligned}[\mathbf{X}_{\alpha,\beta}] &= [\mathbf{X}(u'_\alpha, v'_\beta)] \\ &= [[\mathbf{N}_{e,p}(u'_\alpha)] + \gamma \mathbf{I}_{n+1}]^T [\tilde{\mathbf{P}}_{e,f}][[\mathbf{N}_{f,q}(v'_\beta)] + \delta \mathbf{I}_{m+1}],\end{aligned} \tag{17}$$

where $[\hat{\mathbf{P}}_{e,f}]$ and $[\tilde{\mathbf{P}}_{e,f}]$ are the control point matrices. Hence, the relationship between the models in (15) can be represented by their control point matrices as

$$[\hat{\mathbf{P}}_{e,f}] = [\mathbf{P}_{e,f}] + [\tilde{\mathbf{P}}_{e,f}]. \tag{18}$$

Thus, for the given original control points $[\mathbf{P}_{e,f}]$ and watermarked 2D images $\mathbf{X}_{\alpha,\beta}$, the control points of watermarked NURBS model can be found as

$$\begin{aligned}[\hat{\mathbf{P}}_{e,f}] = {} & [\mathbf{P}_{e,f}] + \left[[\mathbf{N}_{e,p}(u'_\alpha)] + \gamma \mathbf{I}_{n+1}\right]^T \Big]^{-1} [\mathbf{X}_{\alpha,\beta}] \\ & \times [[\mathbf{N}_{f,q}(v'_\beta)] + \delta \mathbf{I}_{m+1}]^{-1}.\end{aligned} \tag{19}$$

We can always find proper γ and δ which make $[[\mathbf{N}_{e,p}(u'_\alpha)] + \gamma \mathbf{I}_{n+1}]^T$ and $[[\mathbf{N}_{f,q}(v'_\beta)] + \delta \mathbf{I}_{m+1}]$ to be nonsingular matrices. The proof of this is shown in Appendix A. Finally, the watermarked surface $\mathbf{S}_{\mathrm{wm}}(u,v)$ can be expressed as

$$\mathbf{S}_{\mathrm{wm}}(u,v) = \sum_{i=0}^{n} \sum_{j=0}^{m} R_{i,j}(u,v) \hat{\mathbf{P}}_{i,j}. \tag{20}$$

When we extract the embedded data or watermarks, the original model is required. The real numbers γ, δ and the parameter sequences $\{u'_\alpha\}$, $\{v'_\beta\}$ play the role of keys for watermarking. For the extraction of watermark, we first derive the matrices of the control points, $[\hat{\mathbf{P}}_{i,j}]$ from $\mathbf{S}_{\mathrm{wm}}(u,v)$ and $[\mathbf{P}_{i,j}]$ from $\mathbf{A}(u,v)$. Then we can obtain the matrix of the control points $[\tilde{\mathbf{P}}_{i,j}]$ of the embedded data model $\mathbf{X}(u,v)$ by subtracting $[\mathbf{P}_{i,j}]$ from $[\hat{\mathbf{P}}_{i,j}]$ as in (18). Now we derive the matrices of the basis function of the embedded data model $\mathbf{X}(u,v)$, that is, $[[\mathbf{N}_{e,p}(u'_\alpha)] + \gamma \mathbf{I}_{n+1}]^T$ and $[[\mathbf{N}_{f,q}(v'_\beta)] + \delta \mathbf{I}_{m+1}]$ by using the keys γ, δ, and $\{u'_\alpha\}$, $\{v'_\beta\}$. Then, by using (17), we can

obtain the embedded data $\{\mathbf{X}_{\alpha,\beta}\}$. Even if the watermarked NURBS model is slightly modified, the matrix of the control points $[\hat{\mathbf{P}}_{i,j}]$ from the watermarked NURBS model $\mathbf{S}_{\mathrm{wm}}(u,v)$ is distorted as well. Then, it is difficult to extract the perfect watermark data $\{\mathbf{X}_{\alpha,\beta}\}$ by using the distorted matrix of the control points $[\hat{\mathbf{P}}_{i,j}]$.

3.3. Algorithm for watermarking

Before describing the algorithm, this subsection first summarize, the assumptions and conditions for the watermarking of NURBS surfaces. For watermarking robust to the attacks, we have to find the exact or almost exact watermarked positions on the surface even after the attacks. Since the NURBS model is comprised of three components—weights, control points, and knot vectors—we assume that these three components are the targets of the attacks.

As stated previously, we use the virtual surface $\mathbf{A}(u,v)$ in (9) instead of the original surface $\mathbf{S}(u,v)$. Hence, it is naturally robust to the attacks on the weights. In the case of the attacks on the control points, the attacker can arbitrarily change the positions and/or the number of the control points. But in modifying the position of control points, it is assumed that the deviation of the control points has certain limit in that the attacker would normally want to preserve the original shape. Among the methods which change the number of the control points while preserving the overall shape of the model, many methods can also change the knot vectors. When the knot vectors are modified, we cannot use any of the parameter sequences because they are related with the knot vectors. In this case, we have to be able to find watermarked positions of the model without parameter sequences. Hence, we must extract parameter sequences according to the innate shape of the model, not from the user-chosen parameter sequences as in the steganography algorithm. For this purpose, we choose u, v values which generate uniformly spaced grid points of the model in the real space. Uniformly spaced grid points of the model mean the equally spaced points on isoparametric curves. We denote the sequences of u and v as $\{u'_\alpha\}$ and $\{v'_\beta\}$, respectively. Then, the watermarking becomes robust to the series of attacks on the knot vector, such as modification of the number of control points, knot insertion, knot removal, or reparameterization of knot vectors. Also, unlike the steganography algorithm, the number of parameter sequences, which is related to the number of sampled points, is not restricted to the number of control points of the model.

The procedure of deciding $\{u'_\alpha\}$ and $\{v'_\beta\}$, which results in the uniform surface sampling, is composed of two procedures. Procedure 1 (see Algorithm 1) extracts the parameter sequence that generates equally spaced points on the curve. We get the parameter sequence that makes all the adjacent points to have the same distance within the error bound by assuming that parameter value increases almost linearly with the distance between two adjacent points. The detailed explanation is shown in Appendix B. Procedure 2 (see Algorithm 2) extracts the parameter sequences $\{u'_\alpha\}$ and $\{v'_\beta\}$ which obtain equally spaced grid points on the surface.

Begin

(1) Initialize u'_i

$$u'_i \longleftarrow u_p + \left(\frac{u_{m-p} - u_p}{k}\right) i \quad (i = 0, \ldots, k).$$

(2) Repeat

(a) Compute the distances between successive points and the average of these

$$d_i = \left|\mathbf{C}(u'_i) - \mathbf{C}(u'_{i-1})\right| \quad (i = 1, \ldots, k)$$

$$\overline{d} = \frac{\sum_{i=1}^{k} d_i}{k}.$$

(b) Break if $|\overline{d} - d_i|/\overline{d} \le$ tolerance for all i.

(c) Recompute u parameters using linear interpolation.

For each i $(i = 1, \ldots, k-1)$,
Find the integer j $(j \in [1,k])$ which makes the condition

$$\sum_{l=1}^{j-1} d_l < \overline{d} \le \sum_{l=1}^{j} d_l.$$

Then, assign $\widehat{u'_i}$ as

$$\widehat{u'_i} \longleftarrow u'_{j-1} + \frac{u'_j - u'_{j-1}}{d_j}\left(\overline{d} \cdot i - \sum_{l=1}^{j-1} d_l\right).$$

(d) Update u parameters as

$$u'_i \longleftarrow \widehat{u'_i} \quad (i = 0, \ldots, k). \tag{21}$$

(e) Go to (a).

End

Algorithm 1

Procedure 2 uses the isometric curves of the surface, while Procedure 1 is required to get the parameter sequences in each curve. The detailed procedure is as follows.

Procedure 1. Compute u parameters (u'_i, where $i = 0, \ldots, k$) to obtain $(k+1)$ equally spaced points on a curve $\mathbf{C}(u)$ as in Algorithm 1.

Procedure 2. Compute u, v parameters (u'_i, v'_j, where $i = 0, \ldots, k$, $j = 0, \ldots, l$) to obtain $(k+1) \times (l+1)$ equally spaced points on a surface $\mathbf{S}(u,v)$ as shown in Algorithm 2.

By using $\{u'_\alpha\}$ and $\{v'_\beta\}$ derived from the above procedures, sampled points $\{\mathbf{A}_{\alpha,\beta}\}$ can be decomposed into three 2D images as in the steganography algorithm. These images are watermarked by 2D watermarking techniques, which are denoted by $\{\hat{\mathbf{A}}_{\alpha,\beta}\}$. And to check the errors for the values between each parameter sequences $\{u'_\alpha\}$, $\{v'_\beta\}$ in a new NURBS model, we define other parameter sequences as

$$\begin{aligned} \{u'_{\alpha+1/2}\} &= \left\{\frac{u'_0 + u'_1}{2}, \ldots, \frac{u'_{k-1} + u'_k}{2}\right\}, \\ \{v'_{\beta+1/2}\} &= \left\{\frac{v'_0 + v'_1}{2}, \ldots, \frac{v'_{l-1} + v'_l}{2}\right\}. \end{aligned} \tag{22}$$

Begin
(1) Compute the averaged v parameters $(\overline{v'}_j)$ using uniform u parameters.
  (a) For each i $(i = 0, \ldots, k)$,

$$u_i'' = u_p + \left(\frac{u_{r-p} - u_p}{k}\right) i \quad (i = 0, \ldots, k).$$

  Compute v parameters $(v_j'|_{u_i''} : v_j'$ given $u_i'')$ to obtain $(l+1)$ equally spaced points on a curve $\mathbf{C}_{u_i''}(v)$ (using Procedure 1).
  (b) $\overline{v'}_j = \sum_{\text{all } u_i''} v_j'|_{u_i''}/(k+1)$.
(2) Compute averaged u parameters $(\overline{u'}_i)$ using uniform v parameters.
  (Analogous to the above procedure.)
(3) Compute v parameters (v_j') to obtain $(l+1)$ equally spaced points using averaged u parameters.
  (a) For each i $(i = 0, \ldots, k)$,
  Compute v parameters $(v_j'|_{\overline{u'}_i})$ for obtaining $(l+1)$ equally spaced points on a curve $\mathbf{C}_{\overline{u'}_i}(v)$ (using Procedure 1).
  (b) $v_j' = \sum_{\text{all } \overline{u'}_i} v_j'|_{\overline{u'}_i}/(k+1)$.
(4) Compute u parameters (u_i') to obtain $(k+1)$ equally spaced points using average v parameters.
  (Analogous to (3).)
End

Algorithm 2

Then, we make a new NURBS model that approximates all the watermarked points $\{\hat{\mathbf{A}}_{\alpha,\beta}\}$ within a specified error bound E and $\check{E}$, using the least control points and degrees as possible. During this procedure, initial parameter sequences $\{u_\alpha'\}$ and $\{v_\beta'\}$ are slightly updated. The detailed algorithm is summarized as follows.

Procedure 3. Approximate $\{\hat{\mathbf{A}}_{\alpha,\beta}\}$ within a specified error E using a surface $\mathbf{S}_{\text{wm}}(u, v)$ as presented in Algorithm 3.

In detecting or extracting the watermarks, the parameter sequences $\{u_\alpha'\}$, $\{v_\beta'\}$ are required to be used as the key. In the case where the sequences are not available due to the attacks on the knots, we can generate the parameter sequences by using Procedure 2.

4. EXPERIMENTAL RESULTS

In the experiment, we use three models, "Head," "Pumpkin," and "Lion,"[1] where Head and Pumpkin are single NURBS models. The Head has 27×31 control points and degree 3 in u, v direction and the Pumpkin has 9×22 control points and degree 3 in u, v direction (Figure 3). The Lion is a model which is composed of 51 NURBS patches (Figure 4).

[1] http://www.cs.unc.edu/~hoff/research/nurbs/lionsamp.html.

Begin
(1) Initialize knot vectors U, V.
(2) Interpolate $\{\hat{\mathbf{A}}_{\alpha,\beta}\}$ with the surface $\mathbf{S}'(u, v)$ with the degree 1 in the u, v direction, where

$$\mathbf{S}'(u, v) = \sum_{i=0}^{k'} \sum_{j=0}^{l'} N_{i,p'}(u) N_{j,q'}(v) \mathbf{P}'_{i,j},$$

that is, $\hat{\mathbf{A}}_{\alpha,\beta} = \mathbf{S}'(u'_\alpha, v'_\beta)|_{k' \leftarrow k, l' \leftarrow l, p' \leftarrow 1, q' \leftarrow 1}$.
(3) Initialize $\{\mathbf{e}_{\alpha,\beta}\}$ which is defined as

$$\mathbf{e}_{\alpha,\beta} \triangleq ||\hat{\mathbf{A}}_{\alpha,\beta} - \mathbf{S}'(u'_\alpha, v'_\beta)||.$$

(4) Initialize $\{\check{\mathbf{e}}_{\alpha+1/2,\beta}\}$, $\{\check{\mathbf{e}}_{\alpha,\beta+1/2}\}$, and $\{\check{\mathbf{e}}_{\alpha+1/2,\beta+1/2}\}$ which are defined as

$$\check{\mathbf{e}}_{\alpha,\beta} \triangleq ||\mathbf{S}(u'_\alpha, v'_\beta) - \mathbf{S}'(u'_\alpha, v'_\beta)||.$$

(5) While $p' < p$ and $q' < q$,
  (a) Remove as many knots as possible while satisfying

$$\mathbf{e}_{\alpha,\beta} \le E, \qquad \check{\mathbf{e}}_{\alpha+1/2,\beta} \le \check{E},$$
$$\check{\mathbf{e}}_{\alpha,\beta+1/2} \le \check{E}, \qquad \check{\mathbf{e}}_{\alpha+1/2,\beta+1/2} \le \check{E}.$$

  (b) Rearrange the range of the parameter indices for each basis function.
  (c) Increase knot multiplicity by one, which in turn increases the degree by one, that is, $p' \leftarrow p' + 1$, $q' \leftarrow q' + 1$.
  (d) Update the control point $\{\mathbf{P}'_{i,j}\}$ of $\mathbf{S}'(u, v)$ so as to approximate $\{\hat{\mathbf{A}}_{\alpha,\beta}\}$ in least square sense [6], that is,

$$\{\mathbf{P}'_{i,j}\} \longleftarrow \underset{\{\mathbf{P}'_{i,j}\}}{\operatorname{argmin}} \left(\sum_{i=0}^{k} \sum_{j=0}^{l} ||\hat{\mathbf{A}}_{i,j} - \mathbf{S}'(u'_i, v'_j)||^2 \right).$$

  (e) Update u'_α, v'_β, that is,
  $u'_\alpha, v'_\beta \leftarrow \operatorname{argmin}_{u,v}(\|\hat{\mathbf{A}}_{\alpha,\beta} - \mathbf{S}'(u, v)\|)$.
  (f) Update $\{\mathbf{e}_{\alpha,\beta}\}$.
(6) Assign $\mathbf{S}'(u, v)$ to $\mathbf{S}_{\text{wm}}(u, v)$.
End

Algorithm 3

We first compare the number of data that can be embedded into the NURBS model in the case of steganography application. In the conventional algorithm [1], we can embed information into only 6 places per NURBS patch, since only three places per a direction are allowed and there are two directions u, v. But, in the proposed steganography algorithm, we can embed information into all the pixels of the three virtual images. Therefore, the proposed algorithm can embed information into more places of surfaces than the conventional algorithm can [1]. Table 1 shows the comparison of the number of the places where the information is embedded in each model.

In the experiment on the watermarking application, the single-NURBS models (Head and Pumpkin) are sampled using 128×128 and 64×64 points, respectively (Figure 5); and all the patches that constitute the NURBS model Lion are

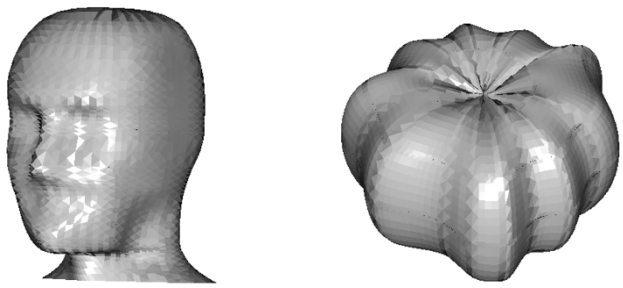

Figure 3: Head and Pumpkin models.

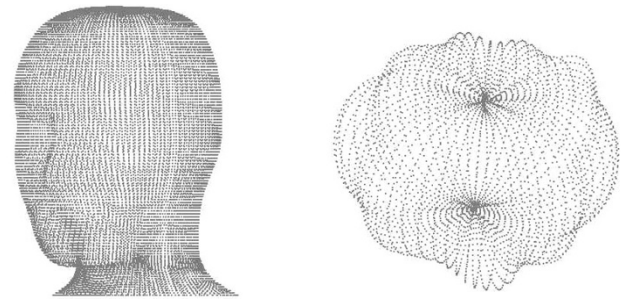

Figure 5: The sampled points of Head and Pumpkin.

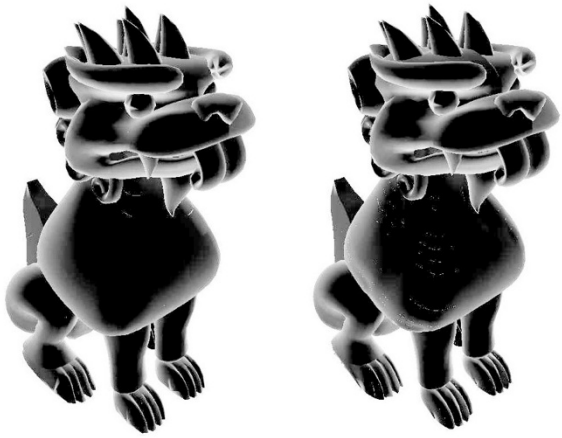

Figure 4: Original Lion and watermarked Lion models.

Table 1: Ratios of the number of data that can be embedded.

Model	Proposed : Conventional
Head	418.5 : 1
Pumpkin	99.0 : 1
Lion	21.3 : 1

appropriately sampled according to the number of control points of each patch. We employ DCT domain watermarking algorithm in [7, 8] as the 2D watermarking method for the virtual images (Figure 6). The details of the watermarking algorithm is introduced in Appendix C. Table 2 shows the numbers of control points in the original and the watermarked NURBS model. Even the largest number of the control points in the watermarked NURBS model is not greater than the number of the sampled points. In the worst case, the number of control points in the watermarked NURBS model is about 20 times larger than that of original, but its data size is not considerable.

There is no generally accepted perceptual measure that shows how much the modified NURBS model is different from the original one. Hence, we define two measures as the distance between two NURBS surfaces. The first one is inspired by [9] which measures the difference between two mesh surfaces. The first, equally spaced points $\{p\}$ are sampled from the original NURBS surface $\mathbf{S}$, and also $\{p'\}$ from the watermarked NURBS surface $\mathbf{S}_{\text{wm}}$. From the experiment, it is found that the 4 times larger number of samples than the original control points is enough to measure the distance in most cases. Following this procedure, we can get range data which have the overall shape of a NURBS model. Given a point p and a surface $\mathbf{S}$, we define a distance $e(p, \mathbf{S})$ as

$$e(p, \mathbf{S}) = \min_{p' \in \mathbf{S}} d(p, p'), \tag{23}$$

where $d(\cdot, \cdot)$ is the Euclidean distance between two points. The one-sided distance from $\mathbf{S}$ to $\mathbf{S}_{\text{wm}}$ is defined as

$$E_1(\mathbf{S}, \mathbf{S}_{\text{wm}}) = \max_{p \in \mathbf{S}} e(p, \mathbf{S}_{\text{wm}}). \tag{24}$$

Note that this definition of distance is not symmetric [9]. That is, there exist surfaces such that $E_1(\mathbf{S}, \mathbf{S}_{\text{wm}}) \neq E_1(\mathbf{S}_{\text{wm}}, \mathbf{S})$. Hence, we define the difference D_1 between the original NURBS surface $\mathbf{S}$ and the watermarked NURBS surface $\mathbf{S}_{\text{wm}}$ to be a two-sided distance as

$$D_1(\mathbf{S}, \mathbf{S}_{\text{wm}}) = \max\left(E_1(\mathbf{S}, \mathbf{S}_{\text{wm}}), E_1(\mathbf{S}_{\text{wm}}, \mathbf{S})\right). \tag{25}$$

And the other one is "root mean square error (RMSE)" between $\{p\}$ and $\{p'\}$. First, we define the one-sided distance from $\mathbf{S}$ to $\mathbf{S}_{\text{wm}}$ as

$$E_2(\mathbf{S}, \mathbf{S}_{\text{wm}}) = \sqrt{\frac{\sum_{p \in \mathbf{S}} \{e(p, \mathbf{S}_{\text{wm}})\}^2}{n}}, \tag{26}$$

where n is the number of samples. Like E_1, this definition of distance is not symmetric. Thus, the difference D_2 between the original NURBS surface $\mathbf{S}$ and the watermarked NURBS surface $\mathbf{S}_{\text{wm}}$ is defined as shown below:

$$D_2(\mathbf{S}, \mathbf{S}_{\text{wm}}) = \max\left(E_2(\mathbf{S}, \mathbf{S}_{\text{wm}}), E_2(\mathbf{S}_{\text{wm}}, \mathbf{S})\right). \tag{27}$$

If all the conditions except for the size of the model are equal, the differences D_1 and D_2 are proportional to the bounding box diagonal of each model.

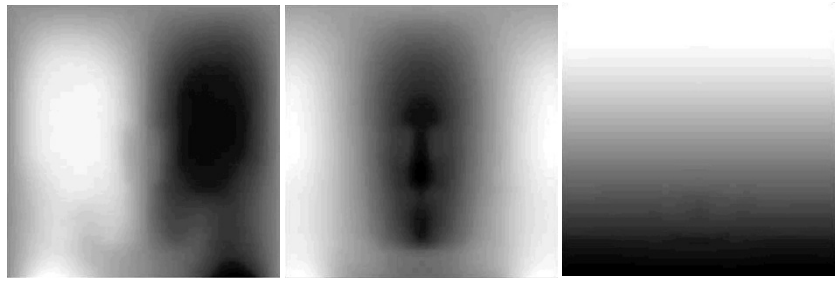

Figure 6: The 2D virtual images of Head (x, y, and z images in order).

Table 2: Numbers of control points.

Model	Original	Watermarked
Head	27×31	$\leq 128 \times 128$
Pumpkin	9×22	$\leq 64 \times 64$

The watermark strength is set to 1 after the pixel values in the virtual 2D images are mapped into the range of $[0, 256)$. When removing knots (Procedure 3(5a)), we update error bounds by using the equations derived in [10]. Table 3 shows the experimental results of the watermark strength versus $D_1(\mathbf{S}, \mathbf{S}_{\text{wm}})$ and $D_2(\mathbf{S}, \mathbf{S}_{\text{wm}})$ for finding the just noticeable difference (JND). Subjectively, $D_1(\mathbf{S}, \mathbf{S}_{\text{wm}}) = 5.06717 \times 10^{-6}$ in Head model and 1.83746×10^{-4} in Pumpkin model are the JND based on our experiments, where the bounding box diagonal of the Head model is 3.527 and that of the Pumpkin model is 172.413. But for the case of $D_2(\mathbf{S}, \mathbf{S}_{\text{wm}})$, it is difficult to find the JND because the watermark strength and the difference D_2 are not much correlated with each other.

In order to evaluate the robustness of watermarks, we detect the watermarks in the watermarked model under the condition that the watermark strength is below JND. We choose three attacks that preserve the overall shape of the model. One is the control point modification by affine transformation. This attack does not affect the watermarks at all because the affine-transformed model begets the affine-transformed sampled points, and the coordinates of these sampled points generate the same 2D images as the original ones. Hence, by using DCT domain watermarking, the effect of the affine-transformation attack disappears. Another attack is the knot vector modification, such as knot insertion, knot removal, or knot reparameterization. In this case, it is difficult to find the sampled points of the watermarked image by using u, v parameter sequences. But we can find points close to the original ones by using Procedure 2 in Section 3, because the initial u, v parameter sequences are chosen to generate equally spaced sampled points. In the experiment, we derive the watermarked images by Procedure 2 without knot vectors. The last attack is the surface approximation. We make two assumptions in this attack. One is that the number of attacker's samples is less than that of control points of the watermarked model because the larger number of sampled points generally means the higher probability that the sampled points have a portion of watermarks. The other assumption is that the surface is equally sampled because the attacker wants to preserve the outline of the model. In this attack, each model is sampled to get a quarter of the original sampled points and the error bound E and $\check{E}$ is set to be 1% of the diagonal of the bounding box. In the case of the second and the last attack, new parameter sequences are used instead of the original ones. Table 4 shows the detection results. In spite of the condition that the false alarm probability is below 10^{-16}, all the watermarks are detected among 10^7 experiments. The false alarm probability in these experiments is specified in Appendix D. The difference $D_1(\mathbf{S}, \mathbf{S}_{\text{wm}})$ is found to be under the JND in each model.

5. CONCLUSIONS

In this paper, we have proposed two algorithms for the watermarking of 3D NURBS surface. One is suitable for the steganography, and the other for the watermarking. These algorithms extract 2D images from 3D NURBS model, the pixels of which represent the coordinates of surfaces for the given parameters. The watermark is embedded into 2D images by using any existing 2D watermarking algorithms.

In the case of steganography algorithm, a virtual model is generated by setting all the weights to 1 and adding some terms to ensure the invertibility of matrices that constitute the NURBS model. With this virtual model and the parameter sequences, we derive linear equations to get the control points of the watermarked model. Since we can embed information into each pixel of three images, the number of data that can be embedded is much greater than that of the conventional algorithm. And since we need original model and parameter sequences as keys, the security is reinforced. The data size is also preserved.

In the case of watermarking algorithm, the numbers of parameter sequences are not restricted to the numbers of the control points. Instead, the values of the parameter sequences are chosen to generate equally spaced points on the surface. Since the virtual model for the watermarking is also generated with all the weights set to be 1, the proposed algorithm is robust to the attacks on weight. And since the virtual images are extracted not by x-, y-, and z-coordinate values, but by u, v parameter sequences, the proposed algorithm is robust to the attacks on the control points. Also, since we can regenerate parameter sequences by using Procedures 1 and 2, it is also robust to the attacks on the knot vectors.

Table 3: Watermark strength versus $D_1(\mathbf{S}, \mathbf{S}_{wm})$ and $D_2(\mathbf{S}, \mathbf{S}_{wm})$.

Watermark strength	Head		Pumpkin	
	$D_1(\mathbf{S}, \mathbf{S}_{wm})$	$D_2(\mathbf{S}, \mathbf{S}_{wm})$	$D_1(\mathbf{S}, \mathbf{S}_{wm})$	$D_2(\mathbf{S}, \mathbf{S}_{wm})$
0.50	3.80393×10^{-6}	1.40559×10^{-6}	1.65935×10^{-4}	6.98483×10^{-5}
0.75	4.56005×10^{-6}	1.50105×10^{-6}	1.65195×10^{-4}	6.29352×10^{-5}
1.00	4.96078×10^{-6}	1.49049×10^{-6}	1.72109×10^{-4}	6.78075×10^{-5}
1.25	5.06717×10^{-6} (JND)	1.49825×10^{-6}	1.81887×10^{-4}	7.28989×10^{-5}
1.50	5.97003×10^{-6}	1.57601×10^{-6}	1.83746×10^{-4} (JND)	6.43491×10^{-5}
1.75	6.23372×10^{-6}	1.62320×10^{-6}	1.90399×10^{-4}	6.66394×10^{-5}

Table 4: Resilience against three attacks.

Model ($D_1(\mathbf{S}, \mathbf{S}_{wm})$)	Attack	False alarm probability	Watermark detection
Head (4.96078×10^{-6})	Control points modification	Below 10^{-16}	100%
	Knot vector modification	Below 10^{-16}	100%
	Surface approximation	Below 10^{-16}	100%
Pumpkin (1.72109×10^{-4})	Control points modification	Below 10^{-16}	100%
	Knot vector modification	Below 10^{-16}	100%
	Surface approximation	Below 10^{-16}	100%
Lion (2.38715×10^{-4})	Control points modification	Below 10^{-16}	100%
	Knot vector modification	Below 10^{-16}	100%
	Surface approximation	Below 10^{-16}	100%

APPENDICES

A. PROOF OF EXISTENCE OF NONSINGULAR MATRIX

Lemma A.1. *Let* $\mathbf{A}$ *be an* $(n \times n)$ *square matrix,* $\mathbf{I}$ *an* $(n \times n)$ *identity matrix, and* x *a scalar. Then, for all* $\mathbf{A}$ *and* $\mathbf{I}$*, there exists* x *which makes* $\mathbf{A} + x\mathbf{I}$ *to be nonsingular.*

Proof. Since $\mathbf{A}$ and $\mathbf{I}$ are $(n \times n)$ square matrices, the determinant of $\mathbf{A} + x\mathbf{I}$ is the nth polynomial of x. Hence the number of solutions which make the determinant of $\mathbf{A} + x\mathbf{I}$ zero is finite (at most n). Therefore, we can always find a proper x that makes $\mathbf{A} + x\mathbf{I}$ to be nonzero. □

B. DETAILED EXPLANATION OF PROCEDURE 1

Like Figure 7, the parameter value u' increases monotonously with the distance d. In this procedure, we assume that parameter value u' increases almost linearly between two adjacent points. Let the expectation of u'_i be $\widehat{u'_i}$. Then we can derive $\widehat{u'_i}$ as

$$\widehat{u'_i} \longleftarrow u'_{j-1} + \frac{u'_j - u'_{j-1}}{d_j}\left(\overline{d} \cdot i - \sum_{l=1}^{j-1} d_l\right) \quad \left(\Longleftrightarrow u'_j - u'_{j-1} : \widehat{u'_i} - u'_{j-1} = d_j : \left(\overline{d} \cdot i - \sum_{l=1}^{j-1} d_l\right)\right). \tag{B.1}$$

We denote the distance between successive points and the average of these after the nth iteration by $d_i^{(n)}$ and $\overline{d}^{(n)}$. Then

$$d_i^{(1)} = \mathbf{C}(\widehat{u'_i}). \tag{B.2}$$

And by (B.1), $\sum_{l=1}^{i} d_l^{(1)}$ approaches $\overline{d} \cdot i$, that is,

$$\sum_{l=1}^{i} d_l^{(1)} \longrightarrow \overline{d} \cdot i. \tag{B.3}$$

After the nth iteration of this procedure,

$$\sum_{l=1}^{i} d_l^{(n)} \longrightarrow \overline{d}^{(n-1)} \cdot i. \tag{B.4}$$

Therefore, for the large integer n,

$$\sum_{l=1}^{i} d_l^{(n)} \simeq \overline{d}^{(n)} \cdot i \tag{B.5}$$

since $\overline{d}^{(n-1)} \simeq \overline{d}^{(n)}$, which means

$$d_i^{(n)} \longrightarrow \overline{d}^{(n)}. \tag{B.6}$$

By this procedure, we can make the distances between successive points approach the average of these.

C. 2D DCT DOMAIN WATERMARKING

The virtual image is divided into 8×8 blocks and each block is transformed by DCT; and the watermark is added to the

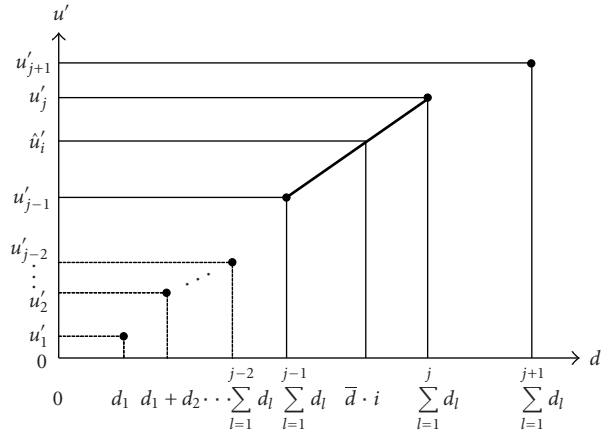

Figure 7: Example of Procedure 1.

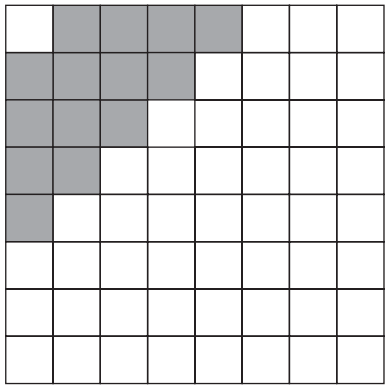

Figure 8: AC components where the watermark is inserted.

AC components as $c' = c + \alpha w$, where c is the original AC component and c' is the watermarked one. The target AC components are shown in Figure 8. One bit is inserted into one 8×8 block. If the bit is 1, the watermark is added as $c' = c + \alpha w$; else $c' = c - \alpha w$. The watermark w is the uniform random sequence of 1 or -1 and the seed value of the random sequence is also used as a key. The α is a parameter to control the watermark strength.

In the extraction, we need the original virtual image obtained from the original 3D model. The original and the watermarked image are normalized and divided into blocks and transformed by DCT as done in the embedding. For each block, the extracted watermark is obtained by the subtraction of AC components as $w^* = (c' - c)/\alpha$. The normalized correlation β is calculated with w and w^* for each 8×8 block [8] as

$$\beta = \frac{\sum_i \left(w_i^* - \overline{w_i^*}\right)\left(w_i - \overline{w_i}\right)}{\sqrt{\sum_i \left(w_i^* - \overline{w_i^*}\right)^2}\sqrt{\sum_i \left(w_i - \overline{w_i}\right)^2}}, \tag{C.1}$$

where $\overline{w}$ is the mean of w. If the inserted bit was 1, β is close to 1 and if the bit was 0, β is close to -1.

D. THE FALSE ALARM PROBABILITY

The probability of false alarm is the probability of declaring "watermark is detected" although there is no watermark in the 3D model. Let H_0 be the event that the model is not watermarked; and the probability of false alarm P_F is calculated as

$$P_F = \Pr(\rho > \lambda | H_0), \tag{D.1}$$

where ρ is the mean of the normalized correlations in (C.1) and λ is a detection threshold. Hence, the pdf $p(\rho|H_0)$ must be obtained in order to calculate P_F. To decide the distribution of $p(\rho|H_0)$, 10 000 ρ are calculated with the randomly generated sequence w and the extracted w^* for each model and each attack; and the inserted message is always bit 1 for each block because only the detection of the watermark is considered here. According to the χ^2 test, the $p(\rho|H_0)$ can be considered as Gaussian. Because $p(\rho|H_0)$ is Gaussian, P_F can be easily calculated given the threshold λ. During the computations, the highest precision was 10^{-16} for the numerical integration of the pdf. Hence, the probability below this limit cannot be calculated. In the experiments, the P_F is below 10^{-16}.

REFERENCES

[1] R. Ohbuchi, H. Masuda, and M. Aono, "A shape-preserving data embedding algorithm for NURBS curves and surfaces," in *Proc. Computer Graphics International (CGI '99)*, pp. 180–187, Canmore, Alberta, Canada, June 1999.

[2] J. J. Lee, N. I. Cho, and J. W. Kim, "Watermarking for 3D NURBS graphic data," in *Proc. IEEE Workshop on Multimedia Signal Processing (MMSP '02)*, pp. 304–307, St. Thomas, Virgin Islands, USA, December 2002.

[3] L. Piegl and W. Tiller, *The NURBS Book*, Springer-Verlag, New York, NY, USA, 2nd edition, 1997.

[4] L. Piegl, "On NURBS: a survey," *IEEE Computer Graphics and Applications*, vol. 11, no. 1, pp. 55–71, 1991.

[5] F. Hartung and M. Kutter, "Multimedia watermarking techniques," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1079–1107, 1999.

[6] C. De Boor, *A Practical Guide to Splines*, Springer-Verlag, New York, NY, USA, 1998.
[7] J. R. Hernandez and F. Perez-Gonzalez, "Statistical analysis of watermarking schemes for copyright protection of images," *Proceedings of the IEEE*, vol. 87, no. 7, pp. 1142–1166, 1999.
[8] I. J. Cox, J. Kilian, F. T. Leighton, and T. Shamoon, "Secure spread spectrum watermarking for multimedia," *IEEE Trans. Image Processing*, vol. 6, no. 12, pp. 1673–1687, 1997.
[9] P. Cignoni, C. Rocchini, and R. Scopigno, "Metro: measuring error on simplified surfaces," *Computer Graphics Forum*, vol. 17, no. 2, pp. 167–174, 1998.
[10] W. Tiller, "Knot-removal algorithms for NURBS curves and surfaces," *Computer-Aided Design*, vol. 24, no. 8, pp. 445–453, 1992.

Jae Jun Lee was born in Seoul, Korea, in May 1979. He received the B.S. and M.S. degrees in electrical engineering in 2001 and 2003, respectively, from Seoul National University, Seoul, Korea. He joined the NHN Corporation, Seoul, Korea, in 2003, where he is currently working on the wireless Internet. His research fields of interest are in signal processing and watermarking.

Nam Ik Cho received the B.S., M.S., and Ph.D. degrees in control and instrumentation engineering from Seoul National University, Seoul, Korea, in 1986, 1988, and 1992, respectively. From 1991 to 1993, he was a Research Associate at the Engineering Research Center for Advanced Control and Instrumentation, Seoul National University. From 1994 to 1998, he was with the University of Seoul, as an Assistant Professor of electrical engineering. He joined the School of Electrical Engineering, Seoul National University, in 1999, where he is currently an Associate Professor. His research interests include speech, image, video signal processing, and adaptive filtering.

Sang Uk Lee received the B.S. degree from Seoul National University, Seoul, Korea, in 1973, M.S. degree from Iowa State University, Ames, in 1976, and the Ph.D. degree from the University of Southern California, Los Angeles, in 1980, respectively, all in electrical engineering. In 1980–1981, he was with the General Electric Company, Lynchburg, Va, and worked for the development of digital mobile radio. From 1981 to 1983, he was a member of technical staff, M/A-COM Research Center, Rockville, Md. In 1983, he joined the Department of Control and Instrumentation Engineering, Seoul National University, where he is currently a Professor at the School of Electrical Engineering. His research fields of interests are in signal processing, 3D image processing, and computer vision.