# Watermarking 3D Polygonal Meshes in the Mesh Spectral Domain

[1]Ryutarou Ohbuchi ohbuchi@acm.org
[2]Shigeo Takahashi takahashis@acm.org
[1]Takahiko Miyazawa k7184@kki.yamanashi.ac.jp
[1]Akio Mukaiyama k7186@kki.yamanashi.ac.jp

[1] Computer Science Department, Yamanashi University, 4-3-11 Takeda, Yamanashi-shi, Japan.
[2] The University of Tokyo, Graduate School of Arts and Sciences, Department of General Systems Studies.

*Abstract*

Digital watermarking embeds a structure called watermark into the target data, such as image and 3D polygonal models. The watermark can be used, for example, to enforce copyright and to detect tampering. This paper presents a new robust watermarking method that adds watermark into a 3D polygonal mesh in the mesh's spectral domain. The algorithm computes spectra of the mesh by using eigenvalue decomposition of a Laplacian matrix derived only from connectivity of the mesh. Mesh spectra can be obtained by projecting coordinates of vertices onto the set of eigenvectors. A watermark is embedded by modifying the magnitude of the spectra. Watermarks embedded by using this method are resistant to similarity transformation, random noise added to vertex coordinates, mesh smoothing, and partial resection of the meshes.

*Keywords: Geometric modeling, information security, information hiding, graph Laplacian, mesh spectra.*

## 1. Introduction

Watermarking adds structures called watermark to various target data objects so that information encoded in the watermark are added to the target data. The watermark must not interfere with the intended purposes of the target object (e.g., viewing for a 2D image data object) and that the watermark should ideally be inseparable from the target object. Embedded watermark can be used to enforce copyright, add comments, detect tampering, or to identify rightful purchasers of the data.

Most of the previous research on watermarking has been concentrating on watermarking "classical" object data types, such as text, 2D still image, 2D movie, or audio data. Numerous papers on watermarking these data objects have been published (see, for example, a book edited by Katzenbeisser, et al. [8]). Recently, an increased popularity and importance of three-dimensional (3D) data objects, such as VRML, MPEG4 and various 3D geometric CAD data has prompted investigation of techniques to watermark 3D models. Most of the watermarking algorithms for 3D geometric models that has been published so far targeted geometric shapes of 3D polygonal meshes [10, 5, 11, 6, 12, 13, 3, 17, 14, 16], while a few have targeted parametric curves and surfaces that are often found in 3D geometric CAD models. Others have targeted movement of 3D models, that is, MPEG 4 facial animation parameters [5] or attributes of 3D polygonal models [12].

In this paper, we propose a watermarking algorithm that embeds data into shapes of 3D polygonal meshes. The algorithm employs spectral decomposition of 3D polygonal mesh shape for watermarking. The watermark produced by the algorithm is robust against similarity transform (i.e., rotation, translation, and uniform scaling). In addition, the watermark can be made resistant against such operations as mesh smoothing ("low-pass filtering" of 3D shapes), random noise added to vertex coordinates, resection of a part of the model, as well as to mesh simplification.

The rest of this paper is structured as follows. In Section 2, the definition of the spectral analysis of polygonal meshes will be followed by the description of our watermarking algorithm based on the analysis. In Section 3, we will present experimental results, followed by a survey on related work in Section 4. We will conclude this paper with summary and future work in Section 5.

## 2. The Spectral Domain Watermarking Algorithm

The watermarking algorithm presented in this paper adds a watermark to the shape of a given 3D polygonal mesh whose shape is defined by its vertex connectivity and vertex coordinates. The watermark is added in a transformed domain. Here, the transformation is *mesh spectral analysis*. Mesh spectral analysis was first employed by Karni and Gotsman [7] for lossy compression of vertex coordinates of polygonal meshes. Mesh spectra is computed from a *Laplacian* matrix derived from connectivity of a polygonal mesh [1, 2].

The watermarking method proposed in this paper embeds information into the mesh shape by modifying its mesh spectral coefficients. An inverse transformation converts the watermarked spectral coefficients back into original mesh whose vertex coordinates are slightly altered. The method is a private watermarking method, meaning that the watermark extraction requires both the watermarked mesh and the original, non-watermarked mesh. The watermark extraction is performed in the mesh spectral domain, comparing spectral coefficients of the watermarked and original meshes.

### 2.1 Spectral Analysis of Polygonal Meshes

The spectrum of a polygonal mesh is computed from connectivity and coordinates of vertices of the mesh. Computation of mesh spectra involves eigenvalue decomposition of a *Laplacian matrix*, which is derived only from the connectivity of the mesh vertices. The decomposition produces a sequence of eigenvalues and a corresponding sequence of eigenvectors of the matrix.

Approximately, smaller eigenvalues correspond to lower spatial frequencies, and larger eigenvalues correspond to higher spatial frequencies. Eigenvectors and spectral coefficients of the smaller eigenvalues represent global shape features, while eigenvectors and spectral coefficients of the larger eigenvalues represent local or detail shape features. Projecting the coordinate of a vertex onto a normalized eigenvector produces a mesh spectral coefficient of the vertex.

There are several different definitions for the mesh Laplacian matrix [1, 2]. We employed a definition by Bollabás [2]. Bollabás calls it a *combinatorial Laplacian* or *Kirchhoff* matrix. The Kirchhoff matrix $\mathbf{K}$ is defined by the following formula;

$$\mathbf{K} = \mathbf{D} - \mathbf{A} \tag{1}$$

$\mathbf{D}$ is a diagonal matrix whose diagonal element $\mathbf{D}_{ii} = d_i$ is a degree (or valence) of the vertex $i$, while $\mathbf{A}$ is an adjacency matrix of the polygonal mesh whose elements $a_{ij}$ are defined as below;

$$a_{ij} = \begin{cases} 1, & \text{if vertices } i \text{ and } j \text{ are adjacent;} \\ 0, & \textit{oterwise.} \end{cases} \tag{2}$$

Karni and Gotsman [7] used another definition of mesh Laplacian $\mathbf{L} = \mathbf{I} - \mathbf{HA}$ for their mesh compression. In their formula, $\mathbf{H}$ is a diagonal matrix whose diagonal element $\mathbf{H}_{ii} = 1/d_i$ is the reciprocal of the degree of the vertex $i$ and $\mathbf{A}$ is the adjacency matrix as the Kirchhoff matrix above.

Compared to $\mathbf{L}$, which is a semi-symmetric matrix, $\mathbf{K}$ is a real-symmetric matrix so that eigenvalue decomposition on $\mathbf{K}$ is somewhat easier to compute. Furthermore, our experiments on compressing vertex coordinates of polygonal meshes using Karni's method [7] showed that the matrices $\mathbf{K}$ and $\mathbf{L}$ exhibited nearly identical properties. We thus decided to use $\mathbf{K}$ for our experiments on watermarking.

A polygonal mesh $M$ having $n$ vertices produces a Kirchhoff matrix $\mathbf{K}$ of size $n \times n$, whose eigenvalue decomposition produces $n$ eigenvalues $\lambda_i$ and $n$ $n$-dimensional eigenvectors $\mathbf{w}_i$ $(1 \le i \le n)$. Projecting each component of the vertex coordinate $\mathbf{v}_i = (x_i, y_i, z_i)$ $(1 \le i \le n)$ separately onto the $i$-th normalized eigenvectors $\mathbf{e}_i$

$$\mathbf{e}_i = \mathbf{w}_i / \|\mathbf{w}_i\| \qquad (1 \le i \le n) \tag{3}$$

produces $n$ mesh spectral coefficient vectors $\mathbf{r}_i = (r_{s,i}, r_{t,i}, r_{u,i})$ $(1 \le i \le n)$. The subscripts $s$, $t$, and $u$ denote orthogonal coordinate axes in the mesh-spectral domain corresponding to the spatial axes x, y, and z. To invert the transformation, multiplying $\mathbf{e}_i$ with $\mathbf{r}_i$ and summing over $i$ recovers original vertex coordinates.

$$\begin{aligned} (x_1, x_2, ..., x_n)^T &= r_{s,1}\mathbf{e}_1 + r_{s,2}\mathbf{e}_2 + \cdots + r_{s,n}\mathbf{e}_n, \\ (y_1, y_2, ..., y_n)^T &= r_{t,1}\mathbf{e}_1 + r_{t,2}\mathbf{e}_2 + \cdots + r_{t,n}\mathbf{e}_n, \\ (z_1, z_2, ..., z_n)^T &= r_{u,1}\mathbf{e}_1 + r_{u,2}\mathbf{e}_2 + \cdots + r_{u,n}\mathbf{e}_n. \end{aligned} \tag{4}$$

### 2.2 Embedding Watermark

The watermarking algorithm of this paper embeds watermark by modifying mesh spectral coefficients derived by using the Kirchhoff matrix. For each spectral axis $s$, $t$, and $u$, a mesh spectra is an ordered set of numbers, that are, spectral coefficients.

Various watermarking algorithms have adopted the principles of *spread spectrum communication* to modulate ordered sets of numbers for watermarking [8]. Properties of spread spectrum communication include low spectral power density and robustness against additive random noise. Both of these two properties are considered advantageous to watermarking, since they correspond to low perceptibility and high noise resistance of watermarks. Our watermarking algorithm presented in this paper also employs a spread-spectrum approach similar to that of Hartung et al [5] to modulate the sequence of numbers obtained by using the mesh spectral analysis.

In the algorithm, the data to be embedded is an m-dimensional bit vector $\mathbf{a} = (a_1, a_2, ..., a_m)$, in which each bit takes values $\{0,1\}$. Each bit $a_j$ is duplicated by *chip rate* $c$ to produce a watermark symbol vector $\mathbf{b} = (b_1, b_2, ...b_{mc})$, $b_i \in \{0,1\}$ of length $m \cdot c$;

$$b_i = a_j, \quad j \cdot c \le i < (j+1) \cdot c \tag{5}$$

Repeatedly embedding the same bit $c$ times increases resistance of the watermark against additive random noise. Averaging the detected signal by $c$ times upon watermark detection reduces the effect of the additive random noise.

The bit vector $\mathbf{b}_i$ is converted to another vector $\mathbf{b}' = (b'_1, b'_2, ...b'_{mc})$ $b'_i \in \{-1,1\}$ by the following simple mapping;

$$b'_i = \begin{cases} -1, & \text{if } b_i = 0; \\ 1, & \text{if } b_i = 1. \end{cases} \tag{6}$$

Let us now consider modulating spectral coefficients of one of the spectral axes $s$. Modulation processes for the other two spectral axes are identical. Let $r_{s,i}$ be the $i$-th spectral coefficient prior to watermarking corresponding to the spectral axis $s$, $p_i \in \{-1,1\}$ be the *pseudo-random number sequence* (PRNS) generated from a known stego-key $k_w$, and $\alpha$ $(\alpha > 0)$ be the modulation amplitude. Watermarked $i$-th spectral coefficient $\hat{r}_{s,i}$ is computed by the following formula;

$$\hat{r}_{s,i} = r_{s,i} + b'_i \cdot p_i \cdot \alpha \tag{7}$$

The extraction algorithm requires the same stego-key, which is a seed for the PRNS used for the embedding, for extraction. Key distribution can be achieved by using a public-key cryptography, for example.

Performing the same to $t$ and $u$ components of the spectrum produces a set of watermarked set of spectral coefficients $\hat{\mathbf{r}}_i = \left(\hat{r}_{s,i}, \hat{r}_{t,i}, \hat{r}_{u,i}\right)$. Inverse-transforming the set of spectral coefficients back into the domain of vertex coordinates $\hat{\mathbf{v}}_i = (\hat{x}_i, \hat{y}_i, \hat{z}_i)$ by using the formula (4) produces a watermarked polygonal mesh.

### 2.3 Extracting Watermark

As a private watermark, watermark extraction of the algorithm described in this paper is non-blind, that is, the extraction requires a *cover-mesh* (i.e., an original mesh without watermark) as well as a *stego-mesh* (i.e., watermarked, and possibly degraded, mesh).

The extraction starts with realignment of the cover-mesh $M$ and the stego-mesh $\hat{M}$. To realign meshes, for each mesh, a coarse approximation of its shape is reconstructed from the first (lowest-frequency) 5 spectral coefficients. Then, a set of eigenvectors is computed from a $3\times3$ covariance matrix derived from the reconstructed shape [4]. A comparison of the two sets of eigenvectors realigns the meshes $M$ and $\hat{M}$.

Each of the realigned meshes is applied with spectral decomposition to produce spectral coefficients $r_{s,i}$ for $M$ and $\hat{r}_{s,i}$ for $\hat{M}$. Multiplying the difference $(\hat{r}_{s,i} - r_{s,i})$ with the same PRNS as is used for the embedding, which is generated from the shared seed $k_w$, and summing the result over $c$ times produces correlation sum for the spectral axis $s$. Summing the correlation sums over all three of the spectral axes produces the overall correlation sum $q_j$;

$$\begin{aligned} q_j &= \frac{1}{3} \sum_{l \in \{s,t,u\}} \sum_{i=j\cdot c}^{(j+1)\cdot c-1} (\hat{r}_{l,i} - r_{l,i}) \cdot p_i \\ &= \frac{1}{3} \sum_{l \in \{s,t,u\}} \sum_{i=j\cdot c}^{(j+1)\cdot c-1} b'_i \cdot \alpha \cdot p_i^2 \end{aligned} \tag{8}$$

If the PRNSs for the embedding and extraction are synchronized, and if disturbances applied to the vertex coordinates of $\hat{M}$ (e.g., additive random noise) are negligible,

$$q_j = c \cdot \alpha \cdot b'_i \tag{9}$$

where $q_j$ takes one of the two values $\{-\alpha c, \alpha c\}$. Since $\alpha$ and $c$ are always positive, simply testing for the signs of $q_j$ recovers the original message bit sequence $a_j$,

$$a_j = sign(q_j) \tag{10}$$

The string $a_j$ can easily be converted to the original message bit sequence $b_i$ by applying an inverse of the mapping of the formula (6).

### 2.4 Mesh Partitioning

Our current implementation performs eigenvalue decomposition by first applying similarity transformation of the matrix using the Housholder method, followed by an iterative method. This approach to eigenvalue decomposition performs well for the meshes of size up to a few hundred vertices. Two problems arise if we try to process a larger mesh. First, computational time for the eigenvalue decomposition on a typical PC becomes more than desirable for typical watermarking applications. Second, numerical stability of the decomposition becomes increasingly questionable.

Our approach to watermarking a larger mesh (e.g., of size $10^4$-$10^7$ vertices) is to partition the mesh into smaller sub-meshes of manageable size (e.g., about 500 vertices) so that watermark embedding and extraction is performed individually within each reasonably sized sub-mesh. This approach is essentially what Karni and Gotsman employed for their polygonal mesh geometry compression [7].

Mesh partitioning has an additional benefit. The watermark can be made resistant against resection by embedding the same copy of information repeatedly into multiple sub-meshes. The watermark can be extracted after a partial resection if the remaining mesh contains at least one sub-mesh that is not affected by the resection.

The untouched sub-mesh can be used for mesh realignment and watermark extraction.

We have implemented a simple mesh-partitioning algorithm that can incorporate human intentions. To partition a mesh, we first specify "feature vertices" manually so that the feature vertices are approximately at the center of desired sub-meshes and that the feature vertices are distributed roughly evenly on the original mesh. Each sub-mesh is then grown around a feature vertex by incrementally expanding the sub-mesh region around the feature vertices by following connectivity of the mesh vertices. The incremental expansion of the sub-meshes halts when the original mesh is fully partitioned.

Mesh spectral analysis uses a Laplacian matrix defined by vertices (and their connectivity) that belong to a sub-mesh, ignoring edges connecting vertices on the boundaries of sub-meshes. Since watermark embedding and extraction requires exactly the same mesh partitioning, feature vertices used for the partitioning of the original cover-mesh must be saved together with the cover mesh to be used for extraction.

### 2.5 Watermarking Parameters

**Chip rate** $c$

In case of watermarking 2D still images or 2D movies, the chip rate $c$ can be quite large, since there are at least a few tenths of thousands of numbers to be manipulated for watermarking. Since many polygonal meshes often have as few as a few hundreds vertices, the chip rate $c$ can only be a moderate number, e.g., 5 or 10. For example, if we were to embed a 32 bit number, the maximum chip rate is 7 for the tiger model shown in Figure 1 which has 254 vertices.

Given a number of vertices, a higher chip rate means lower data capacity for the watermark. Also, as we will discuss in Section 3.3, a higher chip rate increases some aspects of robustness while it decreases others.

**Modulation amplitude** $\alpha$

The modulation amplitude $\alpha$ is computed as a fraction of the largest of the axis-aligned bounding box (AABB) of the target mesh. The $\alpha$ is chosen by the user so that it is small enough to preserve appearance of the model while large enough to withstand as much disturbances (e.g., additive random noise and mesh smoothing) as possible.

**Spectrum allocation**

A mesh having $n$ vertices produces $n$ eigenvalues, $n$ eigenvectors and 3 sets of $n$ spectral coefficients, allowing $n$ bits to be embedded. All of the $n$ spectral coefficients are not used for embedding, however. As described in Section 2.3, 5 eigenvectors corresponding to smallest 5 eigenvalues are used to realign stego- and cover- meshes. Thus, at most ($n$-5) (higher-frequency) coefficients are available for embedding watermark information.

Utilization of these ($n$-5) coefficients depends on which one of the robustness properties is to be emphasized. For example, in order to increase robustness against random noise added to the vertex coordinates, the chip rate $c$ needs to be maximized. On the other hand, if we were to emphasize robustness against smoothing, we need to use only the spectral coefficients corresponding to smaller eigenvalues (i.e., lower frequencies.)

## 3. Experiments and results

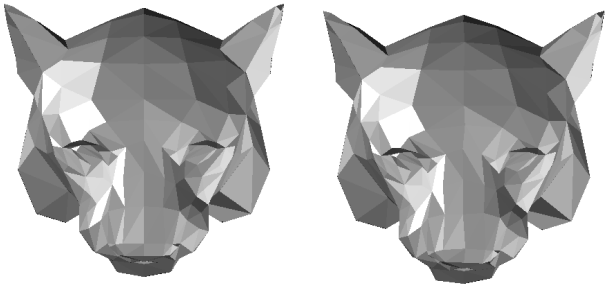
We implemented the algorithm described above using C++, OpenGL graphics API, and fltk (available from http://www.fltk.org), which is a graphical user interface toolkit for the OpenGL.

### 3.1 Watermark Perceptibility

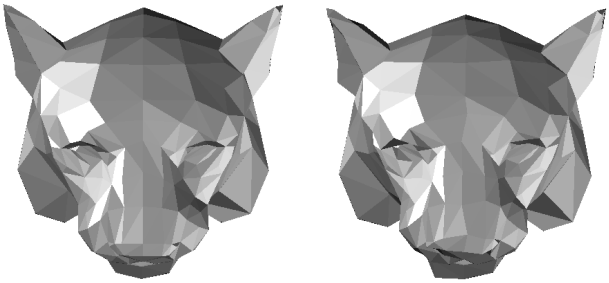
Figure 1 compares perceptibility of watermark in case of the tiger model. Figure1a is a VRML model having 254 vertices and 504 faces without mesh partitioning.

Three other figures show meshes watermarked with three different combinations of modulation amplitude $\alpha$ and the chip rate $c$; $\alpha = 0.002$, $c = 7$ (Figure 1b), $\alpha = 0.005$, $c = 1$ (Figure 1c), and $\alpha = 0.005$, $c = 7$ (Figure 1d). Change in shape is hardly noticeable at $\alpha = 0.002$, $c = 7$. Shape change is hardly noticeable at $\alpha = 0.005$, $c = 1$, but higher chip rate $c = 7$ at the same amplitude made the shape change perceptible.

The perceptibility of a watermark depends not only on the amplitude $\alpha$ and the chip rate $c$, by also to the spectral band employed for the watermark. For example, modifications of the lower frequency coefficients are less noticeable than those of the higher frequency coefficients. Perceptibility also depends on such other factors as the target mesh size and shape. Furthermore, the perceptibility of the watermark embedded in a 3D polygonal mesh depends on the way it is viewed. For example, the perceptibility of the watermark may be affected by the rendering method (e.g., wire-frame or shaded surface rendering), camera parameters, lighting and pixel interpolation method, photometric properties of the mesh (e.g., color, texture, etc.), and many other factors. Thus, the evaluation of perceptibility of watermarks embedded in the shapes of 3D polygonal meshes is likely be more involved than that of 2D still or movie images.



(a) Before watermarking. (b) $\alpha = 0.002$, $c = 7$.



(c) $\alpha = 0.005$, $c = 1$. (d) $\alpha = 0.005$, $c = 7$.

Figure 1. Appearances of the "tiger" model (253 vertices, 504 faces) watermarked with multiple watermark embedding amplitudes $\alpha$ and chip rate $c$ are compared.

### 3.2 Mesh Partitioning and Computation time

Mesh partitioning enables our algorithm to handle meshes much larger than possible without the partitioning.

Figure 5a shows a mesh model of bunny3 containing 2218 vertices and 4432 triangles. In Figure 6a, it is partitioned into 6 sub-meshes. The sizes of sub-meshes in this example ranged from 334 to 494 vertices. Table 1 shows the computation time required for watermarking meshes both with and without mesh partitioning. Even a model with 1197 vertices (bunny2) took more than 30 minutes to process, which is not very useful. However, with partition, the time to process the bunny3 model was reduced to less than 7 minutes. (The bunny meshes of various complexity found in Table 1 and in Figure 6 are created from the bunny mesh obtained from the Stanford University by applying a mesh simplification algorithm.)

Table 1. Computation time for watermark embedding by using a Pentium III 866MHz PC.

| Models | Number of sub-meshes | No. of vertices | No. of faces | Execution time |
|---|---|---|---|---|
| tiger | 1 | 254 | 504 | 20s |
| bunny1 | 1 | 646 | 1288 | 5m21s |
| distcap | 1 | 686 | 1368 | 6m19s |
| bunny2 | 1 | 1197 | 2390 | 33m16s |
| tiger | 3 | 254 | 504 | 2s |
| bunny2 | 4 | 1197 | 2390 | 2m11s |
| bunny2 | 3 | 1197 | 2390 | 3m50s |
| bunny3 | 6 | 2218 | 4432 | 6m34s |

Mesh partitioning reduces computation time and increases robustness against resection. On the other hand, number of embeddable bits, robustness against additive random noise, and robustness against mesh smoothing are reduced. The number of partitioned meshes must be selected with these conflicting factors in mind.

### 3.3 Robustness

We experimentally evaluated the robustness of watermarks produced by using our watermarking algorithm against various disturbances.

Figure 5a and 5e show cover-meshes (*i.e.*, the original meshes) used for the experiments, the bunny3 mesh (2218 vertices) and the distcap mesh (686 vertices). In Figure 5b and 5f, these two meshes are watermarked with $\alpha = 0.005$ and $c = 10$, producing stego-meshes with almost unnoticeable changes in shapes. For the watermarking, the bunny3 mesh was partitioned into 6 submeshes of approximately equal size. The smaller distcap mesh was watermarked without partitioning.

**Similarity Transformation**

Watermarks embedded by using this method are robust against similarity transformation, for the transformation can be identified and inverted by using the method described in Section 2.3.

In Figure 6b, partitioned and watermarked bunny3 mesh was first subjected to resection of the ears, followed by scaling and rotation. The algorithm was able to extract the watermark from the resected and transformed mesh.

**Additive Random Noise**

As discussed in Section 2.5, allocation of available spectral coefficients depends on several different robustness requirements, e.g., robustness against additive random noise and robustness against smoothing. An increase in chip rate $c$ makes the watermark more resistant to random noise added to vertex coordinates. However, the use of highest-frequency band for embedding that resulted from the high chip rate would reduce the watermark's robustness against mesh smoothing.

Figure 5c and Figure 5g show, respectively, the partitioned bunny3 and the (non-partitioned) distcap meshes whose vertex coordinates are added with uniform random noise. The watermarking was embedded using the watermark amplitude factor $\alpha = 0.005$, while

the noise is added using the amplitude factor $\alpha = 0.007$. The watermarks could still be extracted despite the visible degradations of quality in both meshes.

Figure 2 plotted, for the partitioned bunny3 mesh, the bit error rate for the chip rate varying from 1 to 10. The watermark modulation amplitude and the noise amplitude, respectively, are fixed at $\alpha = 0.005$ and $\alpha = 0.007$. As expected, the bit error rates decreased as the chip rate increased. Averaging by the chip error rate $c$ decreased the effect of additive random noise.

The bit error rate is defined, in this paper, as the number of correctly detected bits divided by the number of bits embedded. (The bit error rate of 0.5 means no correlation.) For the experiment shown in Figure 2 and in Figure 4, we embedded 3 different bit patterns of the length 32 bits; all 0, all 1, and random. To compute the bit error rate, we repeated, for each parameter (e.g., a given chip rate in the case of Figure 2), the embedding-extraction run 10 times using each of the data pattern. For example, obtaining the bit error rate for the chip rate $c = 2$ in Figure 2 required 30 embedding-extraction runs. The additive random noise was different every run.

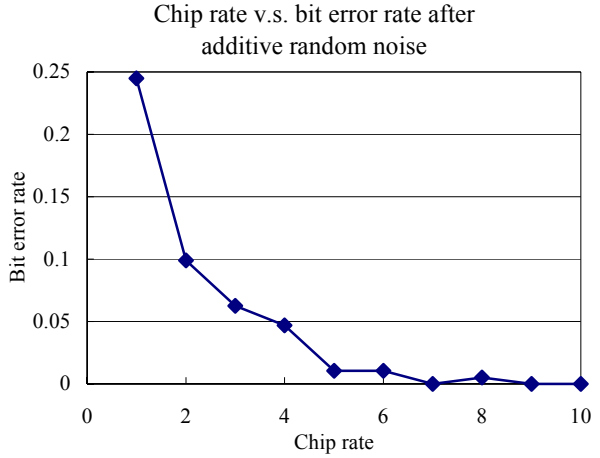



Figure 2. An increase in chip rate decreases bit error rate when random noise is added to the vertex coordinates.

## Mesh smoothing

Polygonal meshes may be subjected to smoothing for fairing or for destroying watermarks. It may thus be important to make the watermark resistant against mesh smoothing as far as the smoothing does not degrade shape of the mesh too much.

Figure 3 plots the difference in amplitudes of the mesh spectral coefficients before and after an application of the Taubin's smoothing filter [15]. Larger changes in amplitudes of coefficients are concentrated in the higher "frequency" bands of the spectrum. This seems to imply the possibility of watermarks resistant against mesh smoothing if we employ only low frequency bands of the spectrum for the watermarking.

Figure 5d and Figure 5h show, respectively, models resulted from an application of Taubin's smoothing filter to the watermarked meshes of Figure 5b and Figure 5f. The smoothing made the bunny3 model smoother and the distcap model deformed. Despite the changes, the watermarks embedded into these models could be extracted.

Figure 4 plots the relationship of watermark modulation amplitude and bit error rate after mesh smoothing for two different chip rates of $c = 1$ and $c = 10$. The bit error rate fell as the watermark modulation amplitude increased. Also the watermark embedded with maximum chip rate of 10 is less robust against smoothing for it used all the spectral bands available, including the higher frequency band that is vulnerable to smoothing.

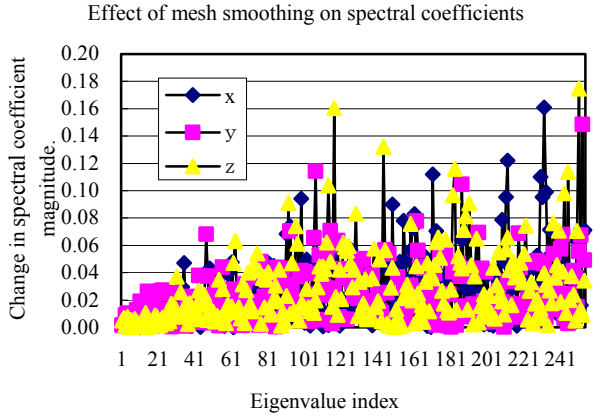



Figure 3. Effects of mesh smoothing on spectral coefficients.

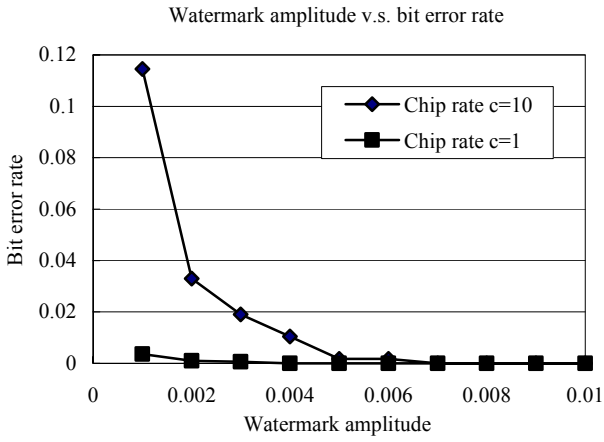
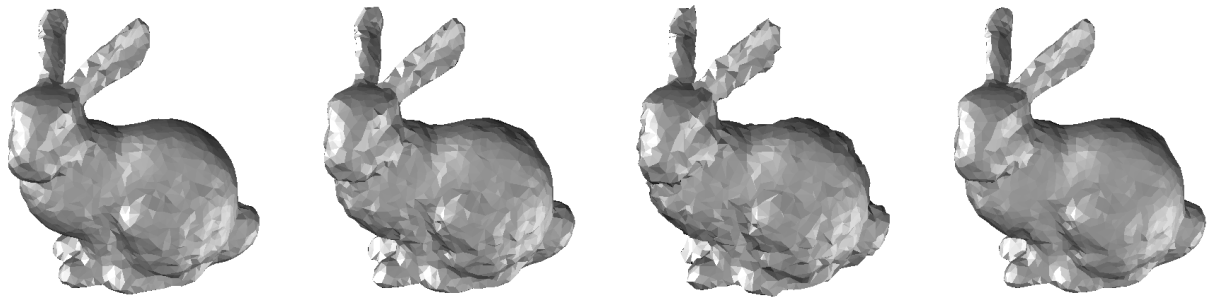
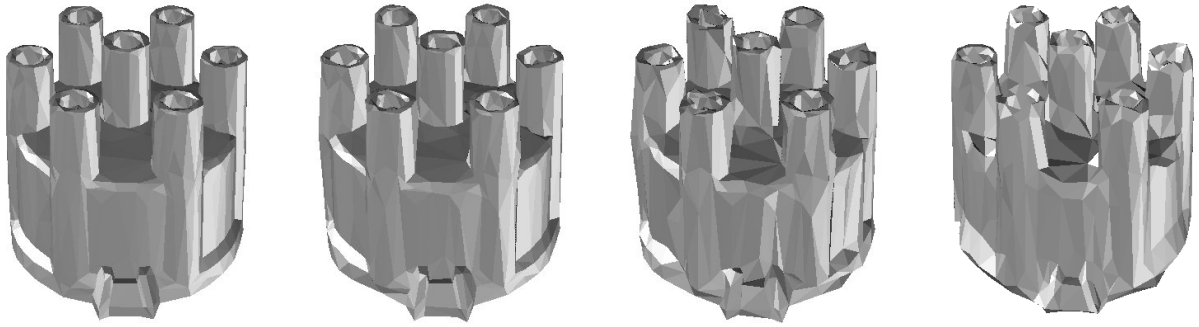



Figure 4. Extraction bit error rate after the model is subjected to the Taubin's smoothing.

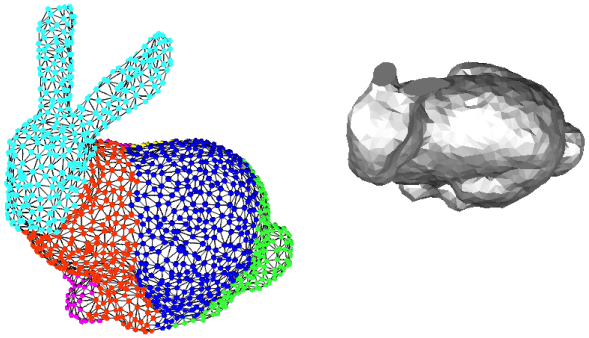


(a) Original (b) Watermarked. (c) Additive random noise. (d) Mesh smoothing.



(e) Original (f) Watermarked. (g) Additive random noise. (h) Mesh smoothing.

Figure 5. The watermarks are resistant against additive random noise and mesh smoothing, among other disturbances. The bunny3 mesh in (a) above that has 2218 vertices was partitioned into 6 sub-meshes for the watermarking and extraction.



(a) The bunny3 mesh partitioned. (b) Partially resected, scaled, and rotated.

Figure 6. The bunny3 mesh model is partitioned into 6 sub-meshes (a). Watermark was extracted after partial resection followed by similarity transformation (b).

**Mesh resection**

The watermark embedded by using mesh partitioning can be made resistant against partial resection of the mesh. Figure 6a shows the bunny3 model partitioned into 6 sub-meshes. After it is watermarked (Figure 5b), its ears are resected, and a similarity transformation is applied (Figure 6b). The watermark could still be extracted from the model shown in Figure 6b. The watermark was extracted by using the sub-mesh near the tail of the bunny model, which was not affected by the resection.

**Combinations of the Operations Above**

Even if a watermark is robust against any one of the operations listed above, combinations of the operations could destroy watermark embedded by using the method described in this paper.

For example, in case of the distcap model of Figure 5e, similarity transformation alone or one pass of mesh smoothing alone left the watermark intact. However, if a watermarked model is smoothed and then applied with similarity transformation, the watermark could not be recovered. In this case, alignment of the cover-mesh and the stego-mesh failed during extraction. This is due partially to the distcap mesh's highly symmetric (axis symmetric) shape, which made the alignment after shape degradation quite difficult. In fact, the result was different in the model of the tiger; one application of mesh smoothing followed by similarity transformation left the watermark intact. Note that if the shape of a mesh is extremely symmetric (i.e., a sphere with a regular triangulation), mere rotation of the mesh may prevent realignment of meshes necessary for proper extraction of watermarks.

## 4. Related Work

The paper by Karni and Gotsman [7] on 3D polygonal mesh compression is the only paper known to the authors that employed the mesh spectra for the global approximation of vertex coordinates of 3D polygonal meshes.

The authors are aware of only one published works by Kanai, et al [6] that employed a wavelet transform for watermarking 3D polygonal meshes. Their algorithm first decomposes a 3D polygonal mesh by using lazy wavelets induced on 3D polygonal meshes. They then modified wavelet coefficients to embed a watermark. Their watermark withstands affine transformation, partial resection and is resistant against random noise added to vertex coordinates. As a limitation, their method requires the mesh to have 1-to-4 subdivision connectivity. Our method, on the other hand, is not limited to meshes having 1-to-4 subdivision connectivity.

Praun and Hoppe [14] developed a robust watermarking algorithm that modifies target meshes by using *ad hoc* spatial kernels to modify shapes of 3D polygonal meshes. Their watermark withstands similarity transformation and is robust against random noise added to vertex coordinates. It is also resistant to mesh simplification. Their method, however, appears to require quite complex target mesh since the spatial kernels used to modify mesh shapes required rather large support (in terms of number of meshes). We speculate that their method would find it difficult to embed 40 bit data into a small, manually tuned VRML model of having 600 faces. (The meshes used in the examples of Praun and Hoppe's paper [14] are of sizes ranging from 13K to 65K polygons.) Compared to their method, our method is able to embed information into meshes that are much less complex than their method would require. Our method, on the other hand, currently can't withstand mesh simplification.

## 5. Conclusion and Future Work

We presented a new watermarking algorithm that embeds data into shapes of 3D polygonal meshes. The algorithm employs mesh spectra as the feature to be modified for watermarking. The mesh spectra are computed by projecting vertex coordinate of the mesh onto eigenvectors of the mesh Laplacian matrix, which is defined by using the connectivity of the mesh. The method is a private watermark, requiring both an original mesh and a watermarked mesh for extraction.

The watermark produced by the algorithm is robust against similarity transform (i.e., rotation, translation, and uniform scaling). In addition, the watermark has been shown to resist such operations as mesh smoothing ("low-pass filtering" of 3D shapes), random noise added to vertex coordinates, and resection of a part of the mesh. The method also has a relatively high information density, being able to embed tens of bits of information into a small mesh having only a few hundred vertices.

Any operation that alters connectivity of meshes could destroy watermarks embedded by using the method described in this paper. We intend to make our method robust against operations that alter connectivity but try to preserve shape. Examples of such operations are edge flipping (with shape preserving some criteria) and mesh simplification. Our proposed approach is to resample the geometry of the watermarked and modified (e.g., by mesh simplification) mesh by using the connectivity of its original. In fact, this is the approach employed by Praun and Hoppe [14].

The algorithm must also be improved so that it can handle a larger mesh, e.g., on the order of 10 k to 100 k vertices. It may be possible to achieve this by improving the mesh-partitioning algorithm.

In the future, we would like to investigate geometric-feature based mesh partitioning and watermarking. Adapting watermarking parameters to each sub-mesh having its own geometric features might allow more robust yet less perceptible watermarks. Such partitioning could also reduce computational costs of watermarking by allowing the embedding algorithm to choose sub-meshes having salient geometric features for watermarking targets.


### Acknowledgements

Mesh data of the rabbit model is obtained from Stanford University, and that of the tiger head is obtained from Avalon archive at Viewpoint Datalabs. The first author is supported in part by grants from the Ministry of Education, Culture, Sports, Science and Technology (MECSST) of Japan (Grant No. 12680432), the Nakayama Hayao Foundation, and the Million, Inc. The second author is supported in part by grants from the MESSC of Japan (Grant No. 12780185) and from the Kayamori Foundation of Information Science Advancement.


### References


[1] N. Biggs, *Algebraic Graph Theory* (2nd Ed.). Cambridge University Press, 1993.

[2] B. Bollobás, *Modern Graph Theory*, Springer, 1998.

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

[4] Gottschalk, S., Lin, M.C., Manocha, D., OBBTree: A Hierarchical Structure for Rapid Interference

Detection, *Proc. SIGGRAPH '96*, pp. 171-180, 1996.

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

[6] S. Kanai, H. Date, and T. Kishinami, Digital Watermarking for 3D Polygons using Multiresolution Wavelet Decomposition, Proc. *Sixth IFIP WG 5.2 GEO-6,* pp. 296-307, Tokyo, Japan, December 1998. (http://minf.coin.eng.hokudai.ac.jp/members/kanai/wm1-geo6.pdf)

[7] Zachi Karni, Craig Gotsman, Spectral Compression of Mesh Geometry, Proc. *SIGGRAPH 2000*, pp. 279-286, July 2000, New Orleans, U.S.A.

[8] S. Katzenbeisser, F. A. P. Petitcolas, *Digital Watermarking*, Artech House, London, 2000.

[9] G. Karypis and V. Kumar, MeTis: A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-reducing Orderings of Sparse Matrices. Version 4.0, Univ. of Minnesota, Dept. of Comp. Sci., 1998. Available at: http://wwwusers.cs.umn.edu/~karypis/metis/metis.html

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

[11] R. Ohbuchi, H. Masuda, and M. Aono, Watermarking Three-Dimensional Polygonal Models Through Geometric and Topological Modifications, pp. 551-560, *IEEE JSAC*, May 1998.

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

[13] Ryutarou Ohbuchi, Hiroshi Masuda, and Masaki Aono, A Shape-Preserving Data Embedding Algorithm for NURBS Curves and Surfaces, Proc. *Computer Graphics International '99*, pp. 180-177, Canmore, Canada, June 7-11, 1999.

[14] Emil Praun, Hugues Hoppe, Adam Finkelstein, Robust Mesh Watermarking, Proc. *SIGGRAPH '99*, pp. 49-56, Aug. 1999.

[15] G. Taubin, A Signal Processing Approach to Fair Surface Design, Proc. *ACM SIGGRAPH '95*, pp. 351-358, 1995.

[16] M. G. Wagner, Robust Watermarking of Polygonal Meshes, Proc. *Geometric Modeling & Processing 2000*, pp. 201-208, Hong Kong, April 10-12, 2000.

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