EUROGRAPHICS 2002 / G. Dettrakis and H.-P. Seidel
(Guest Editors)

Volume 21 (2002), Number 3

A Frequency-Domain Approach to Watermarking 3D Shapes

[1]Ryutarou Ohbuchi [1]Akio Mukaiyama [2]Shigeo Takahashi

ohbuchi@acm.org k7186@kki.yamanashi.ac.jp takahashis@acm.org

[1] Computer Science Department, Yamanashi University, 4-3-11 Takeda, Kofu-shi, Japan

[2] The University of Tokyo, Graduate School of Arts and Sciences, Department of General Studies.

Abstract

This paper presents a robust watermarking algorithm with informed detection for 3D polygonal meshes. The algorithm is based on our previous algorithm [22] that employs mesh-spectral analysis *to modify mesh shapes in their transformed domain. This paper presents extensions to our previous algorithm so that (1) much larger meshes can be watermarked within a reasonable time, and that (2) the watermark is robust against connectivity alteration (e.g., mesh simplification), and that (3) the watermark is robust against attacks that combine similarity transformation with such other attacks as cropping, mesh simplification, and smoothing. Experiment showed that our new watermarks are resistant against mesh simplification and remeshing combined with resection, similarity transformation, and other operations..*

Keywords:
geometric modeling, polygonal meshes, mesh spectral analysis, copyright protection, information hiding, steganography.

1. Introduction

Watermarking adds structures called watermarks to various target data objects so that information encoded in the watermark is 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 the watermark should ideally be inseparable from the target object. Embedded watermarks can be used to enforce copyright, add comments, detect tampering, or to identify rightful purchasers of the data. Please refer to books on watermarking for definitions of the terms (e.g., blind *vs.* informed detection) and other information on digital watermarking [16, 7].

While most of the effort on digital watermarking has concentrated on such media data types as audio, still image, and movie images, increased popularity and importance of three-dimensional (3D) data objects, for example, Web3D, MPEG4 as well as various 3D geometric CAD data has prompted investigation of techniques to watermark 3D models. 3D polygonal meshes have been the most popular targets for watermarking [18, 13, 19, 20, 24, 3, 28, 27, 4, 29]. These methods alter either vertex coordinate or vertex connectivity of the meshes in a way that the modification do not interfere with the use of the model, i.e., viewing by a human observer using a 3D model viewer. A few other methods tried to watermark 3D geometric CAD models defined by using parametric surfaces without any change in shape, a property required by most of the CAD applications [21]. 3D scene animation data [11, 17] and attributes (e.g., vertex coordinates) of 3D polygonal models have been targeted for watermarking.

In the field of image watermarking, a majority of the watermarking algorithms published depends on some form of transformation, e.g., wavelet or Fourier transformations. This is because transformed domain techniques offer various advantages. For example, by modifying the spatial frequency band human beings are not very sensitive to, a watermark embedded in an image can be made less visible. Or, by targeting the coarse shape features, the watermarks embedded are less susceptible to low-pass filtering or additive random noise. In addition, by doing so, watermarks become harder to remove since coarse shape features are often essential to the target content data.

Transformed domain techniques have found uses in the field of watermarking 3D shapes. Kanai, *et al.* [13] is the first to apply a transformed-domain watermarking approach on 3D meshes. It is a robust, blind-detection watermarking algorithm that works in the mesh's wavelet-transformed domain. Kanai's 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 watermarks are resistant against affine transformation, partial resection, and random noise added to vertex coordinates, and other attacks. As a limitation, their method requires the mesh to have 1-to-4 subdivision connectivity.

Praun and Hoppe [24] reported an informed-detection, robust mesh-watermarking algorithm that works in a transformed domain but is applicable to polygonal meshes having arbitrary vertex connectivity. Praun's method modified the shape of the mesh by using a spatial kernel to embed information in the "low-frequency" component of the shape. Their watermarks are resistant against similarity transformation, smoothing, additive random noise, and other attacks. In addition, their watermarks are resistant against mesh simplification and other operations that preserve shape but modify vertex connectivity, by recreating the connectivity of the *reference* (*i.e.*, original) mesh on the *watermarked* (and possibly attacked) mesh by means of mesh alignment followed by resampling.

Yin *et al*. [29] reported an informed-detection, robust mesh-watermarking algorithm that works in a transformed domain. It is based on a multiresolution decomposition of polygonal mesh shapes developed by Guskov *et al* [8] that separates a mesh into detail and coarse feature sequences essentially by repeatedly applying local smoothing combined with shape difference. This watermarking method has shown good robustness property similar to the method proposed by Praun *et al* [24]. In addition, the watermarking algorithm integrates nicely with the other signal processing tools developed by Guskov *et al.* [8].

We have previously presented an informed-detection, robust mesh-watermarking algorithm [22] that embeds message bits by deforming the "low-frequency" components of the shape by using the *mesh spectral analysis* proposed by Karni and Gotsman [15]. As with Praun's and Yin's methods, our previous method is capable of watermarking meshes having arbitrary vertex connectivity. It has a high information payload, thanks to the orthogonal decomposition employed. Resiliency property of the algorithm is similar to Praun's method, except for the connectivity change; if connectivity of the mesh is altered due to remeshing, mesh simplification, and other operations, watermarks fail. This is because the mesh spectral analysis depends on the connectivity of the mesh. The last, but not the least problem is that the high computational cost of the numerical method used for the spectral analysis precluded the analysis and thus watermarking of mesh domains having more than a few thousand vertices.

This paper proposes methods to improve our previous watermarking algorithm [22]. The main contributions of our work presented in this paper are summarized as follows:

(1) **Resiliency Against Connectivity Alterations**: Our watermarks are resilient against such common mesh-connectivity altering operations as mesh simplification and remeshing. We solved this issue by remeshing based on the connectivity of the original mesh.

(2) **Resiliency Against Combined Attacks:** Our watermarks are resilient against attacks that combine cropping with geometric transformation, mesh simplification, smoothing, and other interferences. We employed the careful alignment of possibly cropped meshes based on subsets or "patches" of the mesh

(3) **Performance Improvement**: We have improved the computational efficiency of the spectral analysis more than tenfold by adopting the Arnoldi method for eigenvalue decomposition. We can now analyze and watermark meshes having tens of thousands of vertices as a single domain.

The rest of this paper is structured as follows. In Section 2, we will present our new mesh-watermarking algorithm based on mesh spectral analysis, followed in Section 3 by several experimental results. We will conclude this paper with summary and conclusion in Section 4.

2. The Watermarking Algorithm

The watermarking algorithm described in this paper is an improvement of our previous algorithm [22]. Our previous algorithm watermarks a 3D polygonal mesh by modifying the coarse shape features or "low-frequency component" of the shape. The frequency decomposition is performed by using a technique called *mesh spectral analysis* proposed by Karni and Gotsman [15] that can also be seen as a *principal component analysis* of the shape. To extract a watermark, the algorithm compares the shape of the *reference* (i.e., original) mesh with the *watermarked* (and possibly attacked) mesh in their mesh spectral domain.

Our new algorithm described in this paper still follows our previous method [22] but with the following modifications.

(1) **Efficient eigenanalysis:** We improved the efficiency of the numerical method employed for the eigenvalue decomposition, an operation crucial

to the mesh spectral analysis. The use of the Arnoldi method improved the performance more than 10 times. The algorithm now is able to analyse a much larger mesh region for shape features to be modified for watermarking.

(2) **Per-patch alignment:** We introduced the *per-patch* mesh alignment, instead of per-model alignment used in [21]. The per-patch alignment method enabled extraction of watermarks after cropping followed by geometric transformation. Here, a patch is a subset of the original mesh modified for watermarking, and a mesh usually has multiple watermarking patches.

(3) **Connectivity recovery:** We introduced the remeshing step to recover the exact mesh connectivity of the reference mesh on the watermark mesh whose connectivity is altered. A connectivity alteration may occur due, for example, to mesh simplification and remeshing.

In terms of the computational efficiency, the first improvement above achieved a speed up of more than 10 times over the previous method. We can now analyze and watermark a salient feature having more than a few thousand vertices in a densely sampled mesh containing a large number of vertices.

In terms of attack resiliency, as with our previous algorithm [22], a watermark produced by the new algorithm is resistant against mesh smoothing, because the watermark is embedded in the low-frequency component of the shape. This property combined with the repeated embedding [11] makes the watermark resistant against random noise added to the vertex coordinates.

In addition, with the improved algorithm, the watermark is now resistant against a connectivity alteration that tries to preserve shape. Examples of such connectivity alteration include remeshing and mesh simplification. Since the mesh spectral analysis depends totally on the connectivity of the mesh, our previous watermarking method [22] produces watermarks that are quite fragile against connectivity changes. For proper extraction, an identical connectivity must exist on both the reference mesh and the watermarked mesh so that mesh spectral coefficients can be compared. Our new algorithm thus added a step to recreate the mesh connectivity of the reference mesh on the watermarked mesh by a mesh resampling procedure. Combined with a careful mesh alignment, watermarks produced by our new algorithm are resistant against attacks that combine cropping and similarity transformation with the other attacks.

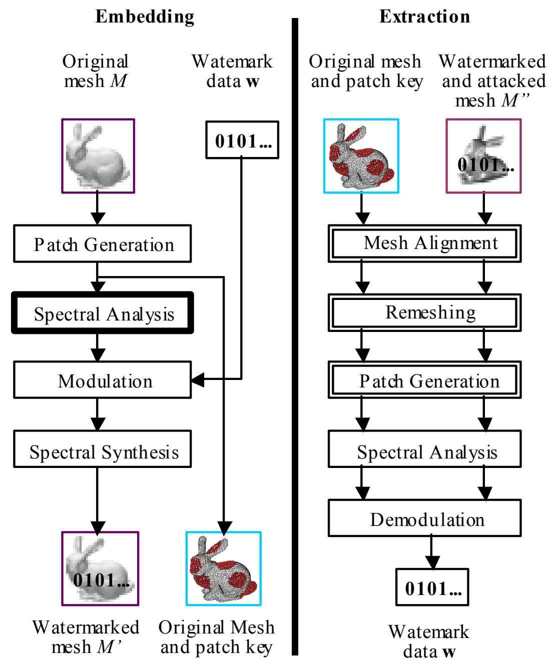


Figure 1. *Overview of our mesh-watermarking algorithm.*

2.1. Algorithm Overview

Our watermarking algori thm embeds information by following the steps below, which is illustrated in Figure 1. In Figure 1, boxes drawn with double lines indicate steps that are added to the original scheme, and a box drawn with thick lines indicates a step that is modified.

(1) Patch Generation: Generate *watermark patches* on the given mesh. A watermark patch is a subset of a mesh to be watermarked. All the following embedding steps, that are, the spectral analysis, modulation, and spectral synthesis are performed per-patch basis. The same watermarking message may be embedded repeatedly into multiple patches. Information on how the original mesh is partitioned, called partitioning key, is saved together with the original mesh for later extraction.

(2) Spectral Analysis: For each patch, a set of spectral coefficients is computed. The computation employs an efficient eigen-decomposition algorithm called Arnoldi method [9, 26].

(3) Modulation: Watermarking message bit string is duplicated many times. The duplicated bit string is converted to a zero-average, randomized {+1, -1} modulation symbol string. The modulation symbol string is used to modulate the amplitude of the spectral coefficients.

(4) Spectral Synthesis: The modified set of spectral coefficients is inverse-transformed into the 3D mesh coordinate value to produce a 3D mesh with watermark.

As an informed-detection watermark, extraction of the watermark requires the original or *reference mesh*. The extraction also requires the *patch key*, the information that uniquely determines the partitioning of the original mesh.

(1) **Mesh Alignment:** The watermarked and possibly attacked mesh is aligned with the reference mesh by minimizing the distance between the two mesh surfaces.

(2) **Remeshing:** Geometry of the watermarked (and possibly disturbed) mesh is resampled using the connectivity of the reference mesh.

(3) **Patch Generation:** The patches used for watermark embedding are recreated on the reference mesh, and then transferred onto the watermarked mesh.

(4) **Spectral Analysis:** For each patch in the reference mesh and its corresponding patch created on the watermarked mesh, spectral analyses are performed. This step produces two spectral coefficient vectors, one from the reference patch and the other from the watermarked patch.

(5) **Demodulation:** The two spectral coefficient vectors are compared. Assuming that the disturbances can be ignored, the modulation-symbol sequence, hence the original message bit string, is recovered.

We describe the details of the steps listed above in the following sections.

2.2. Watermark Embedding

Patch Generation

A *watermarking patch* is a subset of a mesh to be modified for watermarking. We semi-automatically generate patches on the original mesh, by specifying a pair of “seed” vertices that are the centers of the first two patches. A patch is generated around each seed vertex by incrementally adding all the adjacent vertices within a given surface topological distance from the seed vertex. The surface topological distance is computed by using the standard *Dijkstra’s* method. We treat the geodesic (analogue) that connects the two initial seed vertices as an edge of an imaginary regular triangle. Given the edge of a regular triangle, a planar surface can be uniquely tiled by the regular triangle. We approximate such a tiling on a curved mesh surface. While the tiling need not be exact, the patches must not overlap. We save the identifiers of the vertices chosen as the center of the patches together with the reference (*i.e.*, the original) mesh so that the exactly same patches can be re-created quickly during the watermark extraction.

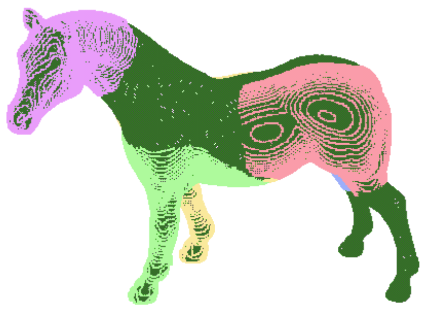


Figure 2. *The horse model (48485 vertices, 96966 faces) is partially covered with 5 watermark patches.*

Figure 2 shows five patches created on the mesh model of a horse having 48485 vertices (96966 faces) using the semi-automatic method. The lower bound for the patch size is set at 7000, so the patches created have sizes 7073, 7063, 7011, 7063, and 7040. The variation in patch size is typically less than a few percent of the size of the patch.

Spectral Analysis

A set of spectral coefficients of a polygonal mesh is computed from connectivity and coordinates of vertices of the mesh. Eigenanalysis of a *mesh Laplacian* matrix, which is defined only from the connectivity of the mesh, is required to produce the coefficients. We employed a mesh Laplacian defined by Bollabás, which is referred to as the *combinatorial Laplacian* or *Kirchhoff* matrix [2]. It is a different mesh Laplacian from what Karni and Gotsman used [15, 1]. 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;

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

A polygonal mesh M having n vertices produces a Kirchhoff matrix $\mathbf{K}$ of size $n \times n$, whose eigenvalue decomposition produces n eigenvalues λ_i and n n-dimensional eigenvectors $\mathbf{w}_i$ $(1 \le i \le n)$. The resulting set of eigenvalue-eigenvector pairs is sorted into an ascending order by the eigenvalue, approximating an ascending order of frequency.

After the sorting, projecting each component of the vertex coordinates $\mathbf{v}_i = (x_i, y_i, z_i)$ $(1 \le i \le n)$ 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.

Approximately, eigenvector-spectral coefficient pairs having smaller eigenvalues correspond to "low-frequency", representing global or large shape features. Conversely, the pairs having larger eigenvalues correspond to "high-frequency", representing local or detail shape features.

Eigenvalue decomposition takes up most of the computation time necessary for the mesh spectral analysis and thus watermark embedding or extraction. Using the Householder transformation followed by the Jacobi iteration, a PC with Athlon 1900+ CPU took 6,404 sec (1 hr 42 min 46 sec) to analyze a mesh having 2218 vertices. This is much too slow for most of the purposes and the eigenvalues in the high-frequency bands are numerically questionable. Furthermore, a few thousand vertices are often not enough to capture a salient shape feature in a densely sampled high-resolution mesh that has become common recently. A watermarking patch should cover the salient shape feature so that it can be protected from right abuses.

To improve efficiency, we adopted the *Arnoldi* method [9] for eigenvalue decomposition. We ported a C implementation of the algorithm in *Meschach* library [26] to our C++-based watermarking code. The Arnoldi method computes a specified number of eigenvalue-eigenvector pairs. If we ask the Arnoldi method to compute $2m$ ($2m \le n$) eigenvalues-eigenvector pairs of the matrix of size *n*, the method computes m smallest and m largest eigenvalues and their corresponding eigenvectors. Time saving due to the Arnoldi method is quite significant, especially if $2m \ll n$ and the matrix is sparse.

The mesh-watermarking algorithm of this paper computes and uses only a subset of eigenvalue-eigenvector pairs in the low-frequency band for watermarking. This is because a set of low-frequency spectral coefficients constitutes a good approximation for a coarse shape [15] and that the coarse shape is the target of our watermarking that should resist smoothing, additive random noise, and other attacks. Thus, the Arnoldi method, which computes a subset of eigenvalue-eigenvector pairs quickly, is quite suitable for our purpose. Of the $2m$ spectral coefficients computed, our watermarking algorithm uses only the m lower-frequency half of the coefficients for watermarking.

Modulation

Our algorithm embeds watermark by modulating amplitude of the mesh spectral coefficients. The coefficients are computed from the Kirchhoff matrix of a patch *P* containing *n* vertices. Each spectral axis *s*, *t*, and *u*, has a mesh spectral coefficient vector of size *n*. To modify the coefficients, our watermarking algorithm employs a spread-spectrum approach similar to that of Hartung et al [11] to modulate the sequence of numbers obtained by using the mesh spectral analysis.

The data to be embedded into a mesh 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 $m \cdot c \le n$;

$$b_i = a_j, \quad j \cdot c \le i < (j+1) \cdot c \tag{4}$$

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;

$$b'_i = \begin{cases} -1, & \text{if } b_i = 0; \\ 1, & \text{if } b_i = 1. \end{cases} \tag{5}$$

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 watermark-key k_w, and α $(\alpha > 0)$ be the modulation amplitude. The modulation amplitude α is computed by $\alpha = \phi \cdot \beta$ where ϕ is the maximum length of the axis-aligned bounding box of the model, and β is the modulation ratio. Watermarked *i*-th spectral coefficient $\hat{r}_{s,i}$ is computed by the following formula;

$$\hat{r}_{s,i} = r_{s,i} + b_i \cdot p_i \cdot \alpha \tag{6}$$

The extraction algorithm requires the same watermark-key, which is a seed for the PRNS used for the embedding, for extraction. Depending on the application of the watermark, the key may be made public. Or, the key may be delivered securely, for example, by using a public-key cryptography.

Performing the same to t and u components of the spectrum produces a set of watermarked set of spectral coefficients $\hat{\mathbf{r}}_i = (\hat{r}_{s,i}, \hat{r}_{t,i}, \hat{r}_{u,i})$.

Spectral Synthesis

Multiplying the eigenvectors $\mathbf{e}_i$ of equation (3) with the watermarked spectral coefficients $\hat{\mathbf{r}}_i = (\hat{r}_{s,i}, \hat{r}_{t,i}, \hat{r}_{u,i})$ and summing over i produce vertex coordinates of the watermarked patch P'.

$$\begin{aligned}(\hat{x}_1,\hat{x}_2,...,\hat{x}_n)^T &= \hat{r}_{s,1}\mathbf{e}_1+\hat{r}_{s,2}\mathbf{e}_2+\cdots+\hat{r}_{s,n}\mathbf{e}_n,\\ (\hat{y}_1,\hat{y}_2,...,\hat{y}_n)^T &= \hat{r}_{t,1}\mathbf{e}_1+\hat{r}_{t,2}\mathbf{e}_2+\cdots+\hat{r}_{t,n}\mathbf{e}_n,\\ (\hat{z}_1,\hat{z}_2,...,\hat{z}_n)^T &= \hat{r}_{u,1}\mathbf{e}_1+\hat{r}_{u,2}\mathbf{e}_2+\cdots+\hat{r}_{u,n}\mathbf{e}_n.\end{aligned} \tag{7}$$

2.3. Watermark Extraction

As an informed-detection watermark, the extraction requires the *reference-mesh*, i.e., an original mesh without watermark, as well as the watermarked, and possibly degraded, mesh.

Mesh Alignment

The extraction starts with alignment of the patch P on the reference mesh M and the patch $\hat{P}$ of the watermarked (and possibly attacked) mesh $\hat{M}$. To improve accuracy, the alignment algorithm could employ as many patches as available depending on the amount of cropping. Especially for a shape that results in a rotationally symmetric patch shape (e.g., a human head), it is advisable to employ more than one patch for accurate registration.

The mesh alignment is a two-step process; a coarse alignment followed by a detailed alignment. The coarse alignment is either manual or automatic, and is necessary to give the detailed alignment step a good start. If the mesh is not cropped, an automatic alignment method is used for the coarse alignment. If the mesh is cropped, we manually align the mesh approximately and turn it over to the detail alignment algorithm.

The automatic, coarse alignment algorithm uses axes of moment of inertia of point mass distributed on the mesh surface. The number of point mass per polygon is proportional to the area of the polygon, approximating the mesh surface with uniformly distributed mass. If the mesh simplification or remeshing preserves the shape reasonably well, the moment computed by the mass-on-surface approach should remain relatively unchanged. We adopted Osada's method [23] to distribute points on a triangle. The location of a point mass $\mathbf{p}$ is computed on a triangle whose vertex coordinates are $\mathbf{t}_1$, $\mathbf{t}_2$, and $\mathbf{t}_3$ using a pair of random numbers r_1 and r_2. Actually, instead of a random number, we used a *low-discrepancy sequence* by *Sobol* [25] for faster conversion.

$$\mathbf{p}=\left(1-\sqrt{r_1}\right)\mathbf{t}_1+\sqrt{r_1}\left(1-r_2\right)\mathbf{t}_2+\sqrt{r_1}\left(r_2\mathbf{t}_3\right) \tag{8}$$

Our mass-on-surface approach is similar to the mass-on-vertex approach by Gottschalk, *et al* [10]. However, if the mesh is attacked by mesh simplification or remeshing that changes the number and location of vertices, our methods produced more accurate alignment. Also, the *Sobol* sequence performed significantly better than the standard random number generator (drand48()).

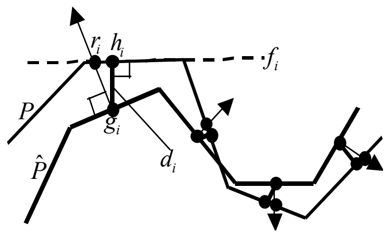


Figure 3. *Computing the mesh patch alignment error for the detailed mesh patch alignment.*

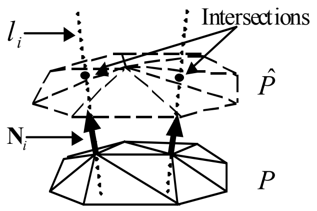


Figure 4. *Geometry of the watermark patch $\hat{P}$ is resampled using the connectivity of the reference patch P.*

For the detailed alignment step, we employ an iterative error-minimization approach similar to Chen [5]. Here, the error is the sum of distance from the reference patch to the watermarked model. The algorithm first computes the intersection $\mathbf{r}_i$ of the normal vector of a triangle through the barycenter $\mathbf{g}_i$ of the triangle on the patch P (Figure 3). The intersection $\mathbf{h}_i$ of the line through $\mathbf{g}_i$ that is perpendicular to the triangle containing the point $\mathbf{r}_i$ is computed. The total error $E(P,\hat{P})$ of the Euclid distance between the points $\mathbf{g}_i$ and $\mathbf{h}_i$ is minimized.

$$E\left(P,\hat{P}\right)=\sum_{i\in P}\sqrt{\left(g_i-h_i\right)^2} \tag{9}$$

To accelerate the ray-tracing necessary for the distance computation, candidate triangles for each intersection are culled by using a uniform space subdivision [12]. For the error minimization task, we used the Powell's method [25], a gradient decent method.

Remeshing

After the patches are aligned, the shape of the watermarked patch $\hat{P}$ is resampled by using the vertex connectivity of the reference patch P. The resampling is done by tracing a ray through each vertex of the reference mesh toward the direction of the normal vector of the vertex (Figure 4). The normal vector is computed as an average of all the normal vectors of the triangles adjacent to the vertex. This ray-tracing occasionally fails to find an intersection. For example, a ray may not have a surface to intersect. If the intersection point is not found in a given radius, the coordinate of the intersection defaults to the coordinate of the reference mesh vertex. The tay-tracing is accelerated by using the same intersection culling technique based on the uniform space subdivision culling implemented for the detailed mesh alignment above. This culling accelerated the remeshing more than 100 times.

Note that this remeshing step as well as the attacks that caused the remeshing adds noise to the vertex coordinates. Fortunately, our watermarking algorithm is resilient against such additive noise and low-pass filtering of mesh shape.

Spectral Analysis

After the remeshing, connectivity of the reference patch P becomes equal to that of the watermarked patch $\hat{P}$ so that comparison of shapes of the patches by using mesh spectral analysis becomes possible. The spectral analysis computation algorithm is exactly the same as that for embedding and is not repeated here.

Mesh spectral analyses of the patches produce the reference spectral coefficients $r_{s,i}$ for the patch P and the watermarked (and possibly attacked) spectral coefficients $\hat{r}_{s,i}$ for the patch $\hat{P}$.

Demodulation

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 watermark key k_w, and summing the result over c times produces the correlation sum for the spectral axis s. Adding the correlation sums from all three of the spectral axes produces the overall correlation sum q_j;

$$\begin{aligned} q_j &= \frac{1}{3} \sum_{l \in \{s,t,u\}} \sum_{i=j \cdot c}^{(j+1)\cdot c-1} (\hat{r}_{l,i} - r_{l,i}) \cdot p_i \\ &= \frac{1}{3} \sum_{l \in \{s,t,u\}} \sum_{i=j \cdot c}^{(j+1)\cdot c-1} b'_i \cdot \alpha \cdot p_i^2 \end{aligned} \tag{10}$$

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,

$$q_j = c \cdot \alpha \cdot b'_i \tag{11}$$

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

$$a_j = sign(q_j). \tag{12}$$

3. Experiments and Results

We have implemented the algorithm described above in C++. We run all the following experiments on a Linux PC with Athlon XP 1900+ (clock 1.6 GHz) processor and 1.5 GByte of memory.

3.1 Computational Efficiency

Table 1 compares the performance of the two eigenanalysis approaches, the Housholder transformation followed by the Jacobi'e method (Housholder + Jacobi) and the Arnoldi method. We used the bunny_09 mesh (2,218 vertices) and all the 2,218 eigenvalue-eigenvector pairs are computed. In this specific case, the Arnoldi method is twice as fast.

Table 2 compares, for three different mesh sizes and three different requested numbers of eigenvalue-eigenvector pairs, the time required by the Arnoldi method for eigenanalysis. It can be seen that the Arnoldi method is very efficient if only a small subset of the eigenvalue-eigenvector pairs are requested. For example, in the case of the bunny_02 mesh (18,957 vertices), the Arnoldi method computed 1,000 (out of the total of 18,957) eigenvalue-eigenvector pairs in about 2,000 sec. Note here that these eigenanalyses are performed on the entire model as a single domain, not on a subset (*e.g.*, a patch) of the model.

More important than the execution time is the fact that this performance improvement enables us to analyze and watermark meshes having large number of vertices. In Figure 2a, the patch having 7,000 vertices covers an entire head of the horse model, capturing a salient shape feature. If the patch size is 1,000 or less, for example, only a subset of the important shape features of the head is covered. Figure 5 shows another example, which is the watermarking the horse mesh having 8,485 vertices as a single domain. In this case, 1,000 out of the 8,485 total of 1,000 eigenvalue-eigenvector pairs are computed. We then employed the lower 500 for embedding a 32 bit message using the chip rate c=15.

3.2 Resiliency Against Attacks

Figure 6 shows an example of watermarking a model of a horse (Figure 6a) having 48,485 vertices (96,966 faces). Watermark patches are the same as in Figure 2,

Table 1. *Comparison of the two eigenanalysis methods.*

Model	Number of vertices	Eigenanalysis method	Time
bunny_09	2,218	Housholder + Jacobi	6,404 s
		Arnoldi	3,222 s

Table 2. *Timing for the eigenvalue decomposition using the Arnoldi method. Only a subset of eigenvalues is computed.*

Model	Number of vertices	Number of eigenvalues computed		
		500	1,000	1,500
bunny_09	2,218	65 s	295 s	836 s
bunny_09	7,565	236 s	800 s	1,465 s
bunny_09	18,957	818 s	2,125 s	3,187 s

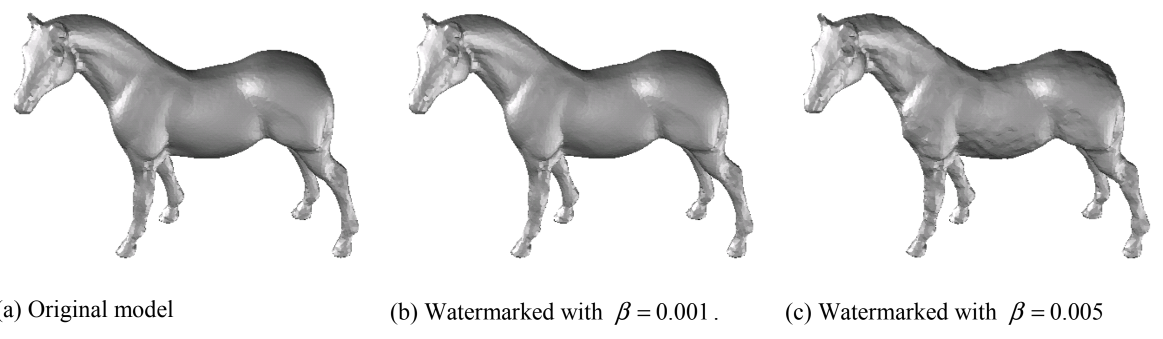

(a) Original model
(b) Watermarked with $\beta = 0.001$.
(c) Watermarked with $\beta = 0.005$

Figure 5. *Watermarking a horse model (8,485 vertices, 19,468 faces) as a single domain. Deformation becomes visible with the modulation factor* $\beta = 0.005$.

in which each patch contains slightly more than 7,000 vertices. Each patch is watermarked using the amplitude factor $\beta = 0.001$ and the chip rate $c = 10$ (Figure 6b). At this amplitude factor, the watermark is not noticeable by human observers.

After remeshing at 8,000 (Figure 6c) and 2,000 (Figure 6d) vertices using the *MeshToSS* [14], the watermark remained. If similarity transformation is applied, about 8,000 vertices are required for the alignment accurate enough for error-free extraction (Figure 6e). For the attack that combined remeshing, cropping, and similarity transformation, the algorithm required 30,000 or so vertices (before cropping) to extract the watermark (Figure 6f). This is due mostly to the alignment and thus remeshing errors, caused in part by a bad choice in the patch placements. Note in Figure 2 that the watermark patches that would remain after cropping cover the legs and the head, which are elongated and thus error prone if slight alignment error occurred.

As another example, we watermarked the bunny mesh of size 13,990 vertices (Figure 7a) using 34 small watermark patches (Figure 7b). The watermark remained after simplification down to 4,990 vertices (Figure 7c), or after the combined attack of mesh simplification down to 5,990 vertices followed by a similarity transformation and a cropping that left 4,730 vertices (Figure 7d).

4. Summary and Conclusion

We have proposed a robust, informed-detection watermarking algorithm for 3D polygonal meshes that embeds watermark by using mesh spectral analysis. It is based on the algorithm we have published previously [22], with major improvements in computational efficiency and attack resiliency.

We improved the performance of the mesh spectral analysis code by more than 10 times, by adopting the Arnoldi method for eigenvalue decomposition. Consequently, our improved algorithm is now able to analyze important shape features for watermarking large meshes. For example, our algorithm is now able to analyze, as a single domain, a mesh containing 10k or so vertices.

We improved robustness of the watermark against attacks. By modifying the low-frequency component of the shape, watermarks produced by our previous method [22] were resilient, to a certain extent, against mesh smoothing and additive random noise. Using the new algorithm reported in this paper, the watermarks are also resilient, to a certain extent, against such vertex connectivity altering operations as mesh simplification and remeshing. We also improved the robustness of the algorithm against combined attacks, *e.g.*, attacks that combine resection, similarity transformation, mesh simplification, and others.

We intend to continue the work on the technique reported in this paper, for example to give the method resiliency against a wider class of attacks and to further reduce computation time. Our future work also includes blind-detection watermarking algorithms and watermarking algorithms for geometric CAD models. A blind-detection watermark is difficult to realize for 3D mesh shapes because the mesh lacks natural parametrization. While Kanai et al. [13] realized a blind-detection watermark, they assumed a fixed parametrization having 4-to-1 subdivision connectivity. Watermarking of geometric CAD models is difficult since they rarely tolerate geometric distortion. We previously reported the results of our initial investigation into this subject [21], and intend to continue the effort.

Curreny it is difficult to compare 3D mesh (or, more broadly, 3D shape) watermarking techniques, hampering the research work. Some of the reasons for the difficulty are, (1) lack of the standard shape representation (NURBS patches, voxels, or meshes?), (2) lack of the standard 3D model data set for benchmarking, (3) lack of the standard sets of attacks, both intentional and unintentional (e.g., compression), and (4) lack of the effective shape distortion measure, both geometric and human-perception based. For example, while geometric distortion can be measured, for example, by the Metro tool [6], perceived distortion is difficult to quantify especially with the rendering and display parameters unconstrained. Researchers in the field of 3D shape (mesh) watermarking need to work together to improve this situation.

Acknowledgements

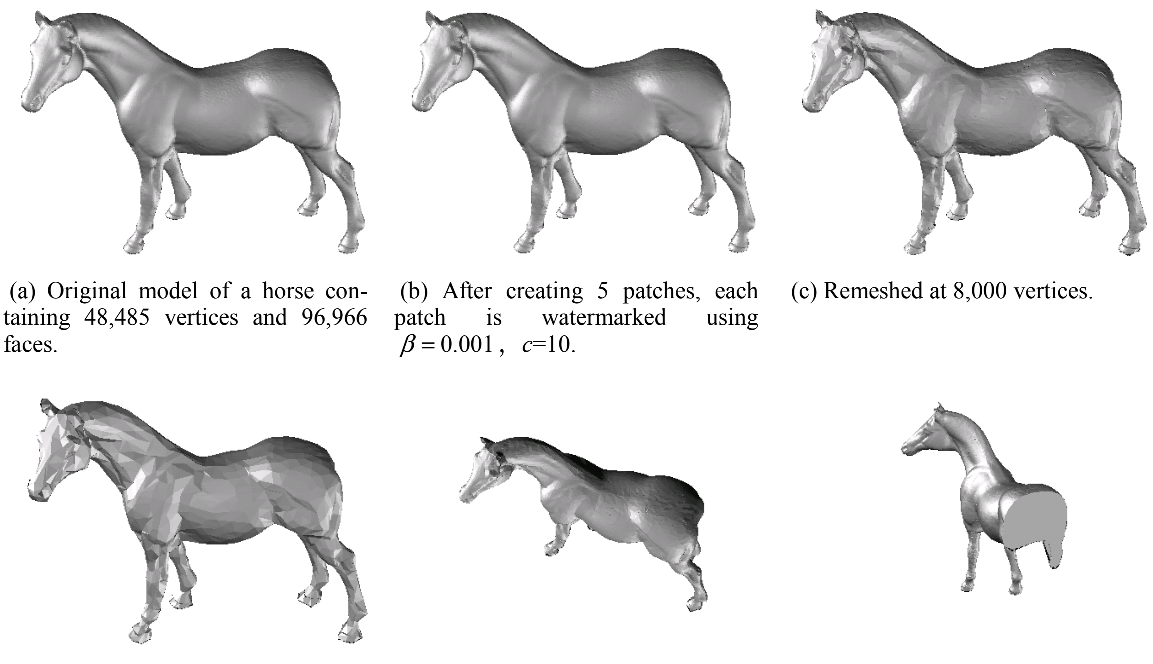
The first and the second authors are supported, respectively, in part by grants No. 12680432 and No. 12780185 from the Ministry of Education, Culture, Sports, Sciences, and Technology of Japan. Mesh data of the bunny model is obtained from the Stanford University.

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, Jan./Feb., 1999.
4. O. Benedens, C. Busch, Towards Blind Detection of Robust Watermarks in Polygonal Models, *Computer Graphics Forum*, Volume 19, No. 3, 2000 (Proc. EUROGRAPHICS 2000)
5. Y. Chen, G. Medioni, Object modeling by registration of multiple range images. *Image and Vision Computing*, Vol. 10, No. 3, pp. 145-155, 1992.
6. P. Cignoni, C. Rocchini and R. Scopigno, Metro: measuring error on simplified surfaces, *Computer Graphics Forum*, vol. 17, No. 2, pp. 167-174. June 1998.
7. I. J. Cox, M. L. Miller, J. A. Bloom, *Digital Watermarking*, Morgan Kaufman Publishers, 2002.
8. I. Guskov, W. Sweldens, P. Shröder, Multiresolution signal processing for meshes, Proc. *SIGGRAPH '99*, pp. 49-56, 1999.
9. Golub, G. H., Van Loan, C. F., *Matrix Computations*, Third Edition, Johns Hopkins University Press, 1996.
10. Gottschalk, S., Lin, M.C., Manocha, D., OBBTree: A Hierarchical Structure for Rapid Interference Detection, Proc. *SIGGRAPH '96*, pp. 171-180, 1996.
11. F. Hartung, P. Eisert, and B. Girod, Digital Watermarking of MPEG-4 Facial Animation Parameters, *Computer and Graphics*, 22:4, pp. 425-435, 1998.
12. Paul S. Heckbert. Ed., *Graphics Gems IV*, AP Professional, 1994
13. 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)
14. T. Kanai, *MeshToSS* Version 1.0.1, http://grahics.sfc.keio.ac.jp/MeshToSS/indexE.html.
15. Zachi Karni, Craig Gotsman, Spectral Compression of Mesh Geometry, Proc. *SIGGRAPH 2000*, pp. 279-286, 2000.
16. S. Katzenbeisser, F. A. P. Petitcolas, Digital Watermarking, Artech House, London, 2000.
17. Tae-hoon Kim, Jehee Lee, Sung yong Shin, Robust Motion Watermarking based on Multiresolution Analysis, *Computer Graphics Forum*, Vol. 19 No. 3, pp. 189-198, 2000 (Proc. EUROGRAPHICS '2000).
18. R. Ohbuchi, H. Masuda, and M. Aono, Watermarking Three-Dimensional Polygonal Models, Proc. *ACM Multimedia '97*, pp. 261-272, 1997.
19. R. Ohbuchi, H. Masuda, and M. Aono, Watermarking Three-Dimensional Polygonal Models Through Geometric and Topological Modifications, pp. 551-560, *IEEE JSAC*, 1998.
20. 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, 1998.
21. 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, 1999.
22. Ryutarou Ohbuchi, Shigeo Takahashi, Takahiko Miyazawa, and Akio Mukaiyama, Watermarking 3D Polygonal Meshes in the Mesh Spectral Domain, in Proc. *Graphics Interface 2001*, pp. 9-17, 2001.
23. Robert Osada, Thomas Funkhouser, Matching 3D Models with Shape Distributions, Proc. *Shape Modeling and Applications '01*, Genova, Italy, pp. 154-166, 2001.
24. Emil Praun, Hugues Hoppe, Adam Finkelstein, Robust Mesh Watermarking, Proc. *SIGGRAPH '99*, pp. 49-56, 1999.
25. W. H. Press et al., *Numerical Recipes in C-The Art of Scientific Programming*, 2nd Ed., Cambridge University Press, Cambridge, UK, 1992.
26. David E. Stewart, Zbigniew Leyk, *Meschach Library* Ver. 1.2b, http://www.netlib.org/c/meschach
27. M. G. Wagner, Robust Watermarking of Polygonal Meshes, Proc. *Geometric Modeling & Processing 2000*, pp. 201-208, Hong Kong, April 10-12, 2000.

28. B-L. Yeo and M. M. Yeung, Watermarking 3D Objects for Verification, *IEEE CG&A*, pp. 36-45, Jan./Feb. 1999.

29. Kangkang Yin, Zhigeng Pan, Jiaoying Shi, David Zhang, Robust mesh watermarking based on multiresolution processing, *Computers & Graphics*, Vol. 25 (2001), pp. 409-420.



(a) Original model of a horse containing 48,485 vertices and 96,966 faces.

(b) After creating 5 patches, each patch is watermarked using $\beta = 0.001$, c=10.

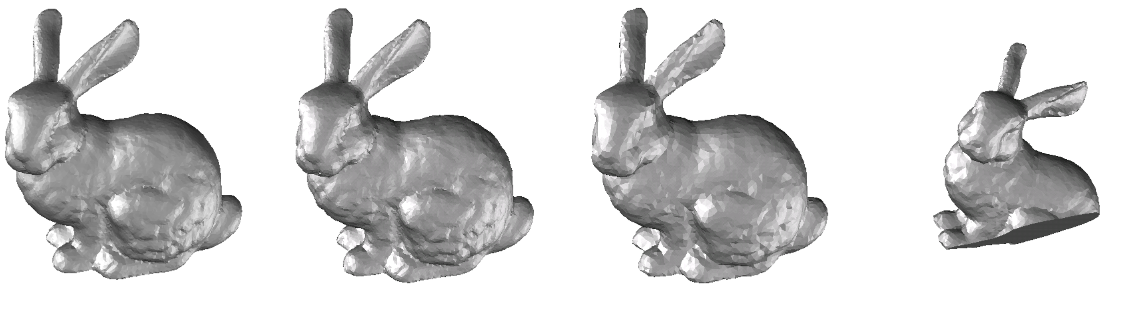
(c) Remeshed at 8,000 vertices.

(d) Remeshed at 2,000 vertices.

(e) Remeshed at 8,000 vertices and similarity transformed.

(f) Remeshed at 30,000 vertices, cropped, and similarity transformed.

Figure 6. *A mesh of a horse (48485 vertices) is watermarked with a 32 bit-long message. The model is watermarked using 5 patches of slightly more than 7000 vertices each.*



(a) The original bunny model containing 13,990 vertices and 27976 and faces.

(b) Watermarked using 34 patches with parameters $\beta = 0.002$, $c = 2$.

(c) Simplified down to 4,990 vertices and 9,976 faces.

(d) Simplified down to 5,990 vertices, cropped to 4737 vertices, and then similarity transformed.

Figure 7. *Examples of watermarking a bunny (13,390 vertices) model with a 32 bit-long message using smaller watermark patches.*