MATHEMATICAL ENGINEERING
TECHNICAL REPORTS

Generalized SSA and Its Applications to Watermarking 3D Polygonal Meshes

Kohei MUROTANI and Kokichi SUGIHARA

METR 2004–17 March 2004

DEPARTMENT OF MATHEMATICAL INFORMATICS
GRADUATE SCHOOL OF INFORMATION SCIENCE AND TECHNOLOGY
THE UNIVERSITY OF TOKYO
BUNKYO-KU, TOKYO 113-8656, JAPAN

WWW page: http://www.i.u-tokyo.ac.jp/mi/mi-e.htm

Generalized SSA and Its Applications to Watermarking 3D Polygonal Meshes

Kohei MUROTANI and Kokichi SUGIHARA

Department of Mathematical Informatics
Graduate School of Information Science and Technology
The University of Tokyo
{muro,sugihara}@mist.i.u-tokyo.ac.jp

March 25th, 2004

Abstract

This paper presents a generalization of the singular spectrum analysis (SSA) and applies it to the construction of a new robust watermarking method that adds a watermark to a 3D polygonal mesh in the spectral domain. The SSA is originally designed as a tool for analyzing one-dimensional sequence such as time series, and hence it is not suitable for multi-dimensional data. In order to overcome this difficulty, we generalize the basic SSA in such a way that it can be used for the 3D polygonal meshes. Watermarks embedded by this method are resistant to similarity transformations and random noises.

1 Introduction

Digital watermarking is a technique for adding secret information called a watermark to various target objects data. The watermark must not interfere with the intended purpose of the target object data (e.g., if the target object data is an image, the watermark should not decrease the geniality of the image), and the watermark should not be separable from the target object data. Embedded watermarks can be used to secure copyright, to add comments, to detect tampering and to identify authorized purchasers of the object.

In general, watermarks are classified into private watermarks and public ones. Private watermarks are retrieved by both of the original data and the watermarked data for reproduction, while public watermarks are retrieved by only the watermarked data. A lot of papers on watermarking have been published [18]. However most of the previous researches have been concentrating on watermarking "classical" object data types, such as texts, 2D

still images, 2D movies, and audio data. Recently, on the other hand, 3D objects data, such as 3D polygonal meshes and various 3D geometric CAD data, become more and more popular and important, and hence techniques to watermark 3D models also become more important [2], [7]-[14], [16]-[17], [19].

In the field of image watermarking, a majority of the watermarking algorithms published depends on some form of transformations, e.g., wavelet or Fourier transformations. This is because transformed domain techniques offer various advantages. For example, by modifying the spatial frequency band which human are not very sensitive to, we can make a watermark embedded in an image less visible. Moreover, the transformed domain is a suitable place to hide the secret data (watermarks). Therefore, in those techniques of watermarking, some kind of spectrum decomposition is required.

Ohbuchi et al. [13] used the eigenvalue decomposition of a Laplacian matrix derived from connectivity of the mesh. Since the eigenvectors of the Laplacian matrix can be regarded as orthogonal coordinate axes in the mesh-spectral domain, the components of the vertex coordinates of the mesh are projected in these eigenvectors. Furthermore, the watermark can be embedded without perturbing the eigenvector-spectral coefficients largely.

Kanai et al. [7] first decomposed a 3D polygonal mesh using lazy wavelets. Then, they modified wavelet coefficients to embed a watermark. This algorithm works in the mesh's wavelet-transformed domain. As a result, their method requires the mesh to have 1-to-4 subdivision connectivity.

The methods proposed by Praun and Hoppe [14] and by Yin et al. [19] are based on a multiresolution decomposition of a 3D polygonal mesh using "progressive mesh" [6]. Progressive mesh is a kind of the spectral decomposition for the mash shape. The Praun and Hoppe [14] modified the shape of the mesh by using a coarse base mesh to embed information in the low frequency component of the shape. They generalized the spread spectrum method by Cox et al. [2] for the robust watermarking method for images, video and sounds to 3D polygonal meshes. They constructed a multiresolution set of scalar basis functions over the mesh and perturbed the coordinates of the mesh vertices by the basis functions as weights. Yin et al. [19] proposed a similar method proposed in the way of embedding watermark, and also used the signal processing tools for meshes developed by Guskov et al. [5] as the multiresolution decomposition of a 3D polygonal mesh. The method of Guskov et al. [5] can separate a mesh into detail and coarse feature sequences by repeatedly applying local smoothing combined with shape difference.

This paper presents an algorithm that embeds watermarks data into 3D polygonal meshes. The proposed method is based on a new kind of spectrum decomposition, and can be used for any mesh structures. The watermark embedded by the algorithm is robust against similarity transformation (i.e.,

rotation, translation, and uniform scaling). It is also resistant against random noises added to vertex coordinates.

In section 2, we will review the singular spectrum analysis (SSA), which is a basic tool for our algorithm. In section 3, we generalize the SSA in order to use it for the 3D polygonal meshes. In section 4, the algorithm for embedding and extracting watermarks will be described. In section 5, we will present experimental results, and in section 6 we conclude this paper with summary and future work.

2 Basic SSA

The singular spectrum analysis (SSA) [3][4] is a novel technique for analyzing time series incorporating the elements of classical time series, multivariate statistics, multivariate geometry, dynamical systems and signal processing. Recently, SSA is one of the popular statistical methods for signal detection in climatology and meteorology. From the fact that Fourier transformation for the auto-correlation function is power spectrum, we can see that the basic SSA is an algorithm of the spectral decomposition.

In this section, we describe the basic algorithm of SSA. In the next section, we generalize it.

2.1 Algorithm of the Basic SSA

Let N be a positive (usually large) integer. Consider a real-value time series $F = (f_0, f_1, \ldots, f_{N-1})$ of length N. Assume that F is a nonzero series, that is, there exist at least one i such that $f_i > 0$. The basic SSA consists of two complementary stages, the decomposition stage and the reconstruction stage.

2.1.1 Decomposition Stage

The decomposition stage consists of the next two steps.

1st Step (Embedding): In the first step, the original time series F is mapped to a sequence of lagged vectors in the following way. Let L be an integer (called a window length) such that $1 < L < N$. We define $K = N - L + 1$, and vectors X_k by

$$X_k = (f_{k-1}, \ldots, f_{k+L-2})^{\mathrm{T}}, \qquad 1 \le k \le K. \tag{1}$$

We shall call X_k's L-lagged vectors. The L-trajectory matrix (or simply trajectory matrix) of the original time series F is defined by

$$\boldsymbol{X} = [X_1 : \ldots : X_K], \tag{2}$$

whose columns are the L-lagged vectors. In other words, the trajectory matrix is

$$\boldsymbol{X} = (x_{l,k})_{l,k=1}^{L,K} = \begin{pmatrix} f_0 & f_1 & f_2 & \cdots & f_{K-1} \\ f_1 & f_2 & f_3 & \cdots & f_K \\ f_2 & f_3 & f_4 & \cdots & f_{K+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{L-1} & f_L & f_{L+1} & \cdots & f_{N-1} \end{pmatrix}. \tag{3}$$

Obviously $x_{l,k} = f_{l+k-2}$ and the trajectory matrix $\boldsymbol{X}$ has equal elements on the diagonal $l + k = \text{const}$. Thus, the trajectory matrix is a Hankel matrix. Certainly if N and L are fixed, then there is a one-to-one correspondence between the trajectory matrix $\boldsymbol{X}$ and the original time series F.

2nd Step (Singular Value Decomposition): In the second step, the singular value decomposition is applied to the trajectory matrix $\boldsymbol{X}$. Let $\boldsymbol{S} = \boldsymbol{X}\boldsymbol{X}^{\mathbf{T}}$. Denote by $\lambda_1, \ldots, \lambda_L$ the eigenvalues of $\boldsymbol{S}$ taken in the decreasing order of magnitude ($\lambda_1 \geq \ldots \geq \lambda_L \geq 0$), and by $U_1, \ldots, U_L$ the orthonormal system of the eigenvectors of the matrix $\boldsymbol{S}$ corresponding to these eigenvalues. Let $d = \max\{i \mid \lambda_i > 0\}$. We defines $V_i = \boldsymbol{X}^{\mathrm{T}} U_i / \sqrt{\lambda_i}$ and $\boldsymbol{X}^{(i)\mathrm{T}} = \sqrt{\lambda_i} U_i V_i^{\mathrm{T}}$ ($i = 1, \ldots, d$). Then the SVD of the trajectory matrix $\boldsymbol{X}$ can be written as

$$\boldsymbol{X} = \boldsymbol{X}^{(1)} + \boldsymbol{X}^{(2)} + \ldots + \boldsymbol{X}^{(d)}. \tag{4}$$

The matrix $\boldsymbol{X}^{(i)}$ has rank 1. Therefore they are elementary matrices. The collection (λ_i, U_i, V_i) is called i-th eigentriple of the singular value decomposition (4).

2.1.2 Reconstruction Stage

3rd Step (Diagonal Averaging): In the last step of the basic SSA, each matrix in the decomposition (4) is transformed into a new time series of length N. This step is called the diagonal averaging. Let The matrix $\boldsymbol{Y}$ be an $L \times K$ matrix with elements $y_{l,k}, 1 \leq l \leq L, 1 \leq k \leq K$.

Diagonal averaging transfers the matrix $\boldsymbol{Y}$ to the time series $(g_0, \ldots, g_{N-1})$ by the formula

$$g_n = \begin{cases} \frac{1}{n+1} \sum_{m=1}^{n+1} y_{m,n-m+2} & (0 \leq n < L-1), \\ \frac{1}{L} \sum_{m=1}^{L} y_{m,n-m+2} & (L-1 \leq n < K), \\ \frac{1}{N-n} \sum_{m=n-K+2}^{N-K+1} y_{m,n-m+2} & (K \leq n < N). \end{cases} \tag{5}$$

The expression (5) corresponds to averaging of the matrix elements over the diagonal $l + k = n + 2$. For $n = 0$, we have $g_0 = y_{1,1}$, for $n = 1$ we have $g_1 = (y_{1,2} + y_{1,2})/2$, and so on. Note that if $\boldsymbol{Y}$ is the trajectory matrix of

some time series $(h_0, \ldots, h_{N-1})$ (in other word, if $\boldsymbol{Y}$ is the Hankel matrix), then $g_n = h_n$ for all n. Diagonal averaging (5) applied to the decomposition matrix $\boldsymbol{X}^{(i)}$ produces the decomposed time series $F^{(i)} = (f_0^{(i)}, \ldots, f_{N-1}^{(i)})$ and therefore the original series $F = (f_0, \ldots, f_{N-1})$ is obtained by the sum of d series

$$F = \sum_{i=1}^{d} F^{(i)}. \tag{6}$$

2.2 Optimality of SVD and the Hankel Matrix

Here, we describe two optimal features in the process of SSA [4]. The first optimality is related to singular value decomposition.

Let $\boldsymbol{X} = [X_1 : \ldots : X_K]$ be the matrix defined by the equation (2), and let $\boldsymbol{X}^{(i)}$ be the matrices defined by the equation (4). Then the following two statements hold.

Propsition 1 *The vector $Q = U_1$ is the solution of the problem*

$$\nu_1 := \sum_{k=1}^{K} (X_k, Q) = \max_{P} \sum_{k=1}^{K} (X_k, P) \tag{7}$$

where the maximum on the right hand side of (7) is taken over all $P \in \boldsymbol{R}^L$ with $||P|| = 1$, and also $\nu_1 = \lambda_1$ holds.

Corollary 2 *Let Q be the solution of the following optimization problem*

$$\nu_i := \sum_{k=1}^{K} (X_k, Q) = \max_{P} \sum_{k=1}^{K} (X_k, P) \tag{8}$$

where the maximum on the right hand side of (8) is taken over all $P \in \boldsymbol{R}^L$ such that $||P|| = 1$ and $(P, U_k) = 0$ for $1 \leq k < i$. If $i \leq d$, then the $Q = U_i$ and $\nu_i = \lambda_i$. If $i > d$, then $\nu_i = 0$.

Proposition 1 and Corollary 2 enables us to call the vector U_i the i-th principal vector of collection $X_1, \ldots, X_K$.

The second optimality is related to diagonal averaging. When a general matrix is transformed to the Hankel matrix, diagonal averaging have the optimality as stated in the following proposition.

Propsition 3 *Assume that $\boldsymbol{Z} = \psi(\boldsymbol{Y})$ is a Hankel matrix of the same dimension as $\boldsymbol{Y}$ such that the difference $\boldsymbol{Y} - \boldsymbol{Z}$ has the minimal Frobenius norm. Then the element $\tilde{y}_{l.k}$ of the matrix $\psi(\boldsymbol{Y})$ is given by*

$$\tilde{y}_{l,k} = \begin{cases} \frac{1}{n+1} \sum_{m=1}^{n+1} y_{m,n-m+2} & (0 \leq n < L-1), \\ \frac{1}{L} \sum_{m=1}^{L} y_{m,n-m+2} & (L-1 \leq n < K), \\ \frac{1}{N-n} \sum_{m=n-K+2}^{N-K+1} y_{m,n-m+2} & (K \leq n < N). \end{cases} \tag{9}$$

The linear operator ψ is called the Hankelization operator.

3 Generalized SSA

Since the basic SSA is designed for the analysis one dimensional sequences such as time series, it is not necessarily appropriate for the 3D polygonal mesh. In this section, we generalize the basic SSA in such a way that it can be applied to the analysis of multi-dimensional data such as polygonal meshes.

3.1 From Basic SSA to Generalized SSA

In the case of the basic SSA, the original series F is transformed to the trajectory matrix $\boldsymbol{X}$. Here, we generalize this process. Let us define the linear operator $\boldsymbol{A}$ which maps F to $\boldsymbol{X} = \boldsymbol{A}(F)$ as

$$\boldsymbol{A}(F) = \begin{pmatrix} FA_{0,0} & FA_{0,1} & \cdots & FA_{0,K-1} \\ FA_{1,0} & FA_{1,1} & \cdots & FA_{1,K-1} \\ \vdots & \vdots & \ddots & \vdots \\ FA_{L-1,0} & FA_{L-1,1} & \cdots & FA_{L-1,K-1} \end{pmatrix} \tag{10}$$

where

$$A_{l,k} = (a_{l,k,0},\ a_{l,k,1}, \cdots,\ a_{l,k,N-1})^{\mathrm{T}} \tag{11}$$

and the elements of $\boldsymbol{A}(F)$ are

$$FA_{l,k} = \sum_{n=0}^{N-1} a_{l,k,n} f_n. \tag{12}$$

Thus, we consider the generalized trajectory matrix (10) instead of (3).

The decomposition is done in the same way as the basic SSA, that is, we decompose $\boldsymbol{X}$ into $\boldsymbol{X}^{(1)}, \boldsymbol{X}^{(2)}, \cdots, \boldsymbol{X}^{(d)}$ shown in the expression (4) by the singular value decomposition.

Our next question is how to reconstruct the original series F from $\boldsymbol{X}^{(1)}, \cdots, \boldsymbol{X}^{(d)}$. In the basic SSA, this is done by diagonal averaging show in the expression (5). In the generalized case, on the other hand, it dose not work; we need some other idea. To answer this question, we define the series $F^{(i)} = (f_0^{(i)}, f_1^{(i)}, \cdots, f_{N-1}^{(i)})$ as the solution of the next optimization problem:

$$\begin{aligned} &\min \sum_{i=1}^{d} \parallel \boldsymbol{X}^{(i)} - \boldsymbol{A}(F^{(i)}) \parallel^2 \\ &\qquad = \min \sum_{i=1}^{d} \sum_{l,k} (x_{l,k}^{(i)} - F^{(i)} A_{l,k})^2 \end{aligned} \tag{13}$$

$$\text{s.t.} \quad F = \sum_{i=1}^{d} F^{(i)}, \tag{14}$$

where the norm of the matrix is the Frobenius norm. If $A_{l,k}(0 \leq l \leq L-1, 0 \leq k \leq K-1)$ span N dimensional spaces, then the matrix $\sum_{l,k} A_{l,k} A_{l,k}^{\mathrm{T}}$ is regular, and consequently the solution of the expression (13) is given by

$$F^{(i)} = \left(\sum_{l,k} A_{l,k} A_{l,k}^{\mathrm{T}} \right)^{-1} \sum_{l,k} x_{l,k}^{(i)} A_{l,k}^{\mathrm{T}}. \tag{15}$$

Since this $F^{(i)}$ satisfies

$$\begin{aligned} \sum_{i=1}^{d} F^{(i)} \sum_{l,k} A_{l,k} A_{l,k}^{\mathrm{T}} &= \sum_{i=1}^{d} \sum_{l,k} x_{l,k}^{(i)} A_{l,k}^{\mathrm{T}} \\ &= \sum_{l,k} \left(\sum_{i=1}^{d} x_{l,k}^{(i)} \right) A_{l,k}^{\mathrm{T}} \\ &= \sum_{l,k} \Big(F A_{l,k} \Big) A_{l,k}^{\mathrm{T}} \\ &= F \sum_{l,k} A_{l,k} A_{l,k}^{\mathrm{T}}, \end{aligned}$$

the constraint (14) is satisfied automatically. The solution $F^{(i)}$ of the optimization problem (13) and (14) is obtained by the expression (15).

Finally, from the expression (15), the original series F is reconstructed as $F = \sum_{i=1}^{d} F^{(i)}$. These are the basic ideas of the generalized SSA.

3.2 Linear Operator $\boldsymbol{A}$

In this subsection, we give a particular example of the linear operator $\boldsymbol{A}$ in the expression (10) that reflects the connectivity structure of the mesh. The linear operator constructed in this section is used in the experiments of watermarking in section 5.

Let P be a 3D polygonal mesh, and let v_i, $i = 0, 1, \cdots, N-1$ be the vertices of the mesh. Suppose that some scalar value f_i is assigned to each vertex v_i, and let F be the series $F = (f_0, f_1, \cdots, f_{N-1})$. We define the distances $D_{v_i}(v_j)$ from v_i to v_j as the shortest distance in the graph where the weight 1 is given to the all edges, so-called the Dijkstra distance. Fig. 1 shows an example of a part of the graph structure associated with the polygonal mesh. Let v_l be the vertex represented by the black dot in Fig. 1. Then, the vertices v_j's with $D_{v_i}(v_j) = 1$ are as shown by empty circles, and the vertices with $D_{v_i}(v_j) = 2$ are as shown empty squares.

Let the number of the rows of the trajectory matrix $\boldsymbol{A}(F)$ be $L := N$. Let the elements on the first column and the l-th row $(l = 0, 1, \cdots, N-1)$ of the trajectory matrix $\boldsymbol{A}(F)$ be the value f_l given to the vertex v_l and let the elements on the k-th $(1 \leq k \leq K-1)$ column and the l-th row be

the average value of the values on the vertices where the Dijkstra distances from v_l are k. Therefore the l-th row of the matrix $\boldsymbol{A}(F)$ corresponds to the vertex v_l of the mesh. For example; for the vertex v_l in Fig. 1. $(l,1)$ element of $\boldsymbol{A}(F)$ is f_l, $(l,2)$ element of $\boldsymbol{A}(F)$ is the average of the value f_j over the vertices represented by the empty circles, $(l,3)$ element of $\boldsymbol{A}(F)$ is the average of the value f_j over the vertices represented by the empty squares. Thus, let $FA_{l,k}$ be

$$FA_{l,k} = \frac{\sum_{D_{v_l}(v_j)=k} f_j}{\#\{D_{v_l}(v_j) = k\}} \tag{16}$$

where $\#\{D_{v_l}(v_j) = k\}$ is the number of the vertices where the Dijkstra distances from v_l are k. The linear operator $\boldsymbol{A}$ in the expression (16) is represented as

$$a_{l,k,n} = \begin{cases} \frac{1}{\#\{D_{v_l}(v_n)=k\}} & (D_{v_l}(v_n) = k), \\ 0 & (D_{v_l}(v_n) \neq k). \end{cases} \tag{17}$$

The rows of $\boldsymbol{A}$ correspond to the vertices of the mesh, and the columns of $\boldsymbol{A}$ correspond to the set of vertices with the same Dijkstra distances. Here the linear operator reflects the connectivity structure of the mesh. We use this linear operator $\boldsymbol{A}$ in our watermarking method.

4 Algorithms for Watermarking in the Spectral Domain

In this section, we propose two algorithms for embedding watermarks into 3D polygonal meshes, one based on the basic SSA and the other based on the generalized SSA. In both of the algorithms, the spectra of the 3D polygonal mesh are computed by the singular decomposition of the trajectory matrix, and the watermarks are the singular values. This method is for private watermarking, meaning that the watermark extraction requires both the watermarked mesh and the original non-watermarked mesh. The watermark can be extracted by comparing singular values of the watermarked data and the original data in the spectral domain. In the following, we describe this algorithm in more details.

4.1 Spectral Decomposition

In this subsection, we perform the spectral decomposition of the 3D polygonal meshes using the basic SSA and the generalized SSA.

In the case of the basic SSA, since the basic SSA is the method of the spectral decomposition of the one dimensional sequence, we assign a linear

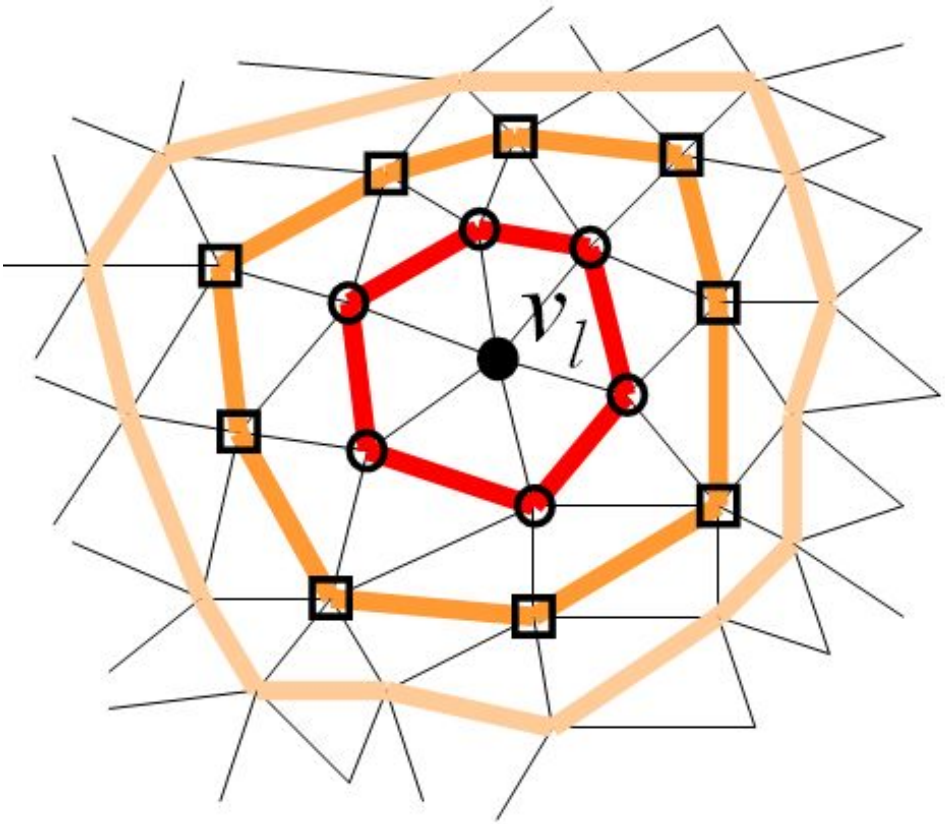


Figure 1: A vertex of the mesh and the set of vertices with the same Dijkstra distances.

order to the vertices. Examples of such orders include the random order and the shortest Hamiltonian path order.

In the case of the generalized SSA, we need not assign a linear order to the vertices; instead we just use the linear operator $\boldsymbol{A}$ constructed in section 3.2.

While we have been considering a scalar-value series $F = (f_0, f_1, \ldots, f_{N-1})$, we hereafter consider tri-value series $\boldsymbol{F} = (F_0, \ldots, F_{N-1})$ with the coordinates $F_n = (f_{n,x}, f_{n,y}, f_{n,z})^{\mathrm{T}}$ of the vertex v_n. Consequently, the trajectory matrix (3) in the case of the basic SSA is expanded into a $3L \times K$ matrix

$$\boldsymbol{X} = \begin{pmatrix} F_0 & F_1 & F_2 & \cdots & F_{K-1} \\ F_1 & F_2 & F_3 & \cdots & F_K \\ F_2 & F_3 & F_4 & \cdots & F_{K+1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ F_{L-1} & F_L & F_{L+1} & \cdots & F_{N-1} \end{pmatrix} \tag{18}$$

and the trajectory matrix (18) in the case of the generalized SSA is expanded

into a $3L \times K$ matrix

$$\boldsymbol{X} = \boldsymbol{A}(\boldsymbol{F}) = \begin{pmatrix} \boldsymbol{F}A_{0,0} & \cdots & \boldsymbol{F}A_{0,K-1} \\ \boldsymbol{F}A_{1,0} & \cdots & \boldsymbol{F}A_{1,K-1} \\ \vdots & \ddots & \vdots \\ \boldsymbol{F}A_{L-1,0} & \cdots & \boldsymbol{F}A_{L-1,K-1} \end{pmatrix}, \tag{19}$$

$$\boldsymbol{F}A_{l,k} = \left(\sum_{n=0}^{N-1} a_{l,k,n} f_{n,x}, \sum_{n=0}^{N-1} a_{l,k,n} f_{n,y}, \sum_{n=0}^{N-1} a_{l,k,n} f_{n,z} \right)^{\mathrm{T}}. \tag{20}$$

We perform the singular value decomposition (SVD) for both of these trajectory matrices. The SVD produces a sequence of singular values and the corresponding sequence of elementary matrices.

Approximately, large singular values correspond to lower spatial frequencies, and small singular values correspond to higher spatial frequencies. Elementary matrices associated with higher singular values represent global shape features, while elementary matrices associated with lower singular values represent local or detail shape features.

4.2 Embedding Watermark

Suppose that we want to embed an m-dimensional bit vector $\boldsymbol{a} = (a_1, a_2, \cdots, a_m)$ where each bit takes value 0 or 1. Each bit a_i is duplicated by chip rate c to produce a watermark symbol vector $\boldsymbol{b} = (b_1, b_2, \cdots, b_{mc})$, $b_i \in \{0, 1\}$ of length $m \times c$

$$b_i = a_j, \qquad (j-1)c + 1 \le i \le jc, \qquad \text{for } j = 1, 2, \cdots, m. \tag{21}$$

Embedding the same bit c times increases the resistance of the watermark against random noises, because averaging the detected signal c times reduces the effect of the random noises. Let $b' = (b'_1, b'_2, \cdots, b'_{mc})$, $b'_i \in \{-1, 1\}$ be the vector defined by

$$b'_k = \begin{cases} -1 & \text{for } b_i = 0, \\ 1 & \text{for } b_i = 1. \end{cases} \tag{22}$$

For $i = 1, 2, \cdots, mc$, let us choose $p_i \in \{-1, 1\}$ in an appropriate way described later. Moreover, let α be a positive constant, called the watermark embedding amplitude. We keep the largest s singular values $\lambda_1, \lambda_2, \cdots, \lambda_s$ unchanged. The i-th bit b_i is inserted into the $(i+s)$-th singular value λ_{i+s}, i.e., the $(i+s)$-th singular value is converted by the following formula

$$r_i = \sqrt{\lambda_{i+s}} + b'_i p_i \alpha, \tag{23}$$

where p_i's are secret number chosen from $\{1, -1\}$. Using r_i for $i = 1, 2, \ldots, d-s$, we construct the trajectory matrix by

$$\begin{aligned} \boldsymbol{X'} &= \sum_{i=1}^{s} \sqrt{\lambda_i} U_i V_i^{\mathrm{T}} + \sum_{i=s+1}^{d} r_{i-s} U_i V_i^{\mathrm{T}} \\ &= \sum_{i=1}^{d} \sqrt{\lambda_i} U_i V_i^{\mathrm{T}} + \sum_{i=s+1}^{d} b'_{i-s} p_{i-s} \alpha U_i V_i^{\mathrm{T}}. \end{aligned} \tag{24}$$

From this matrix, the vertex coordinates $\boldsymbol{F'} = (F'_0, \ldots, F'_{N-1})$ with $F'_n = (f'_{n,x}, f'_{n,y}, f'_{n,z})^{\mathrm{T}}$ are computed by using the formula (5) in the case of the basic SSA or (15) in the case of the generalized SSA. As a result, the vertices of the 3D original polyhedral mesh are converted into the watermarked vertices, which are slightly altered from the original positions.

Finally, we consider how to chose p_i's. If $p_i \in \{-1, 1\}$ are chosen randomly, both the centers of gravity $\phi(\boldsymbol{F})$ and $\phi(\boldsymbol{F'})$ are quite different (where $\phi(\boldsymbol{F})$ denotes the center of gravity of $\boldsymbol{F}$). The center of gravity is a very important invariant for recovering the watermark, and hence it is desirable if we could select p_i so that $\phi(\boldsymbol{F}) = \phi(\boldsymbol{F'})$. Since we cannot satisfy $\phi(\boldsymbol{F}) = \phi(\boldsymbol{F'})$ accurately, we try to minimize the absolute value of the second term in the right-hand side of equation (24), i.e., we choose the approximately optimal p_i by solving the optimization problem

$$\min \parallel \sum_{i=s+1}^{d} b'_{i-s} p_{i-s} \phi(\boldsymbol{F}_i) \parallel_2 \qquad \text{s.t.} \quad p_i \in \{-1, 1\}. \tag{25}$$

4.3 Extracting Watermark

In our method, the extraction of the watermark requires both of the original 3D polygonal mesh and the watermarked 3D polygonal mesh. The watermarked 3D polygonal mesh may be transformed by similarity transformations. Hence, the extraction starts with fitting the original 3D polygonal mesh to the watermarked 3D polygonal mesh by translations, rotations and scaling. First, the data are translated so that the center of gravity of the watermarked 3D polygonal mesh coincides with that of the original mesh. Next, coarse approximations of their 3D polygonal meshes are reconstructed from the first s singular values and the corresponding elementary matrices. The number s is reasonably determined by experiences. Then, each set of eigenvectors is computed from a 3×3 covariance matrix derived from each reconstructed shape. Then the data are rotated and scaled so that the directions and the sizes of two sets of eigenvectors coincide with each other.

An example is shown as Fig. 2, where (a) shows the original 3D polygonal mesh, (b) shows the coarse approximation of the mesh reconstructed from the first s singular values and the corresponding elementary matrices ($s =$

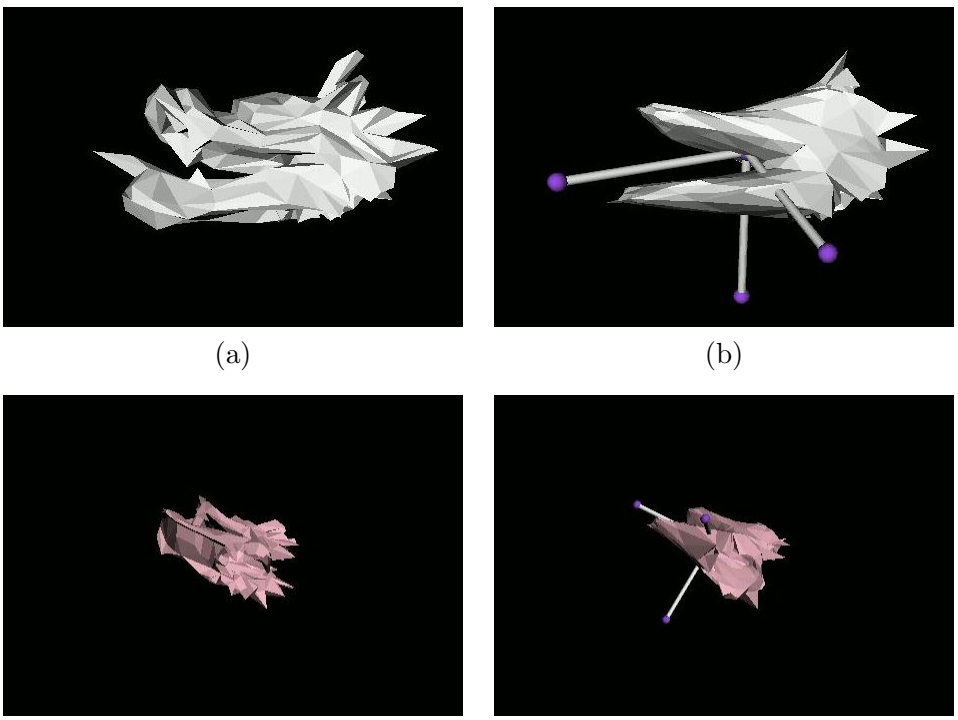


Figure 2: The perception of similarity transformation.

15 in the case of the basic SSA and $s = 3$ in the case of the generalized SSA). On the other hand, (c) shows a watermarked mesh transformed by similarity transformation, and (d) shows the coarse approximation of this mesh. The arrows in (b) and (d) show the three principal axes detected from the covariance matrices of the reconstructed meshes.

Next, the singular value decomposition is performed for the original trajectory matrix $\boldsymbol{X}$ to produce the singular values $\sqrt{\lambda_i}$ and the associated elementary matrices $\boldsymbol{X}^{(i)}$. For the watermarked trajectory matrix $\boldsymbol{X'}$, we do not apply the singular value decomposition; instead we use the orthogonal matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ for $\boldsymbol{F}$ and compute $\sqrt{\lambda'_i}$ by the equations (7) and (8). Multiplying the difference $(\sqrt{\lambda'_{i-s}} - \sqrt{\lambda_{i-s}})$ with p_i and summing the result over c times, we obtain

$$q_j = \sum_{i=c(j-1)+1}^{cj} (\sqrt{\lambda'_{i+s}} - \sqrt{\lambda_{i+s}})p_i \approx \sum_{i=c(j-1)+1}^{cj} b'_i p_i^2 \alpha. \tag{26}$$

Since $p_i = -1$ or 1, we set

$$q_j \approx b'_i \alpha c \tag{27}$$

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

sequence

$$a_j = \{\mathrm{sign}(q_j) + 1\}/2. \tag{28}$$

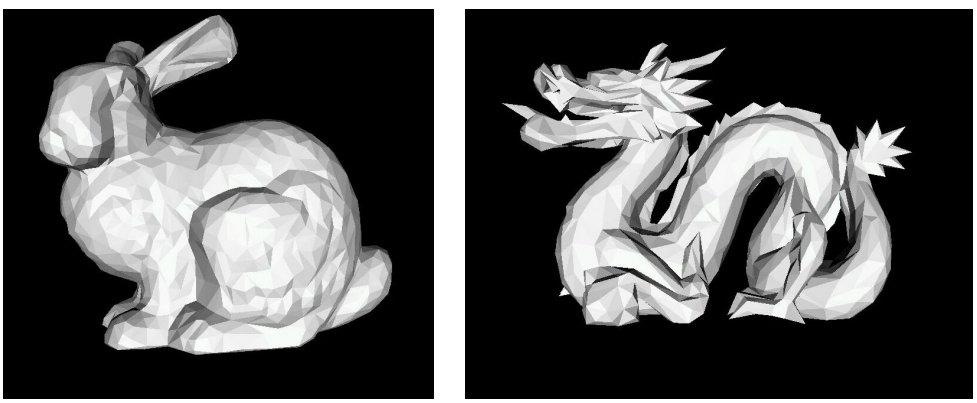
As a result, we can extract the embedded watermark.

5 Experiments and Results

We made computational experiments in order to evaluate the performance of the proposed algorithms. In this section, we describe the methods and the results together with discussion.

5.1 Method

In our experiments, we used two popular mesh models, the bunny model (1494 vertices, 2915 faces) shown in Fig. 3 (a) and the dragon model (1257 vertices, 2730 faces) shown in Fig. 3 (b).

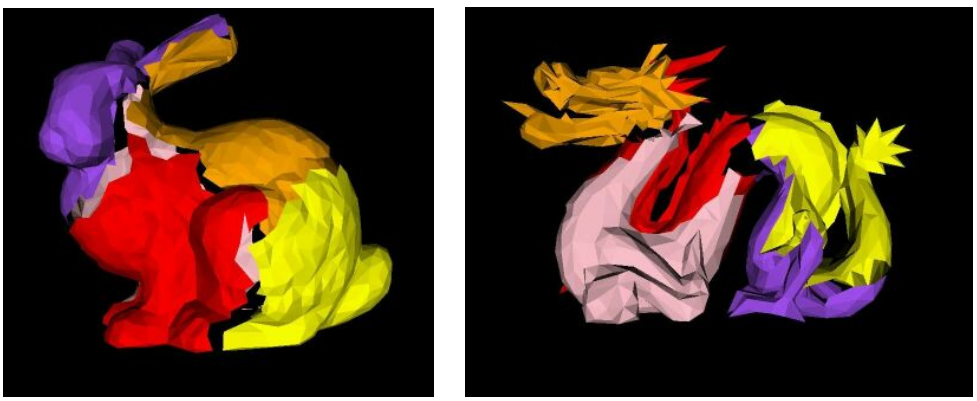


(a) (b)

Figure 3: The original meshes of the bunny and dragon models.

We compare five methods. Some of them require high computational cost, and hence we partitioned the vertex series into smaller vertex sub-series of a treatable size, so that the calculations times are decreased and the accuracies are increased. In this experiments, the vertex series is partitioned into 5 groups of the same size, as shown in Fig. 4. The embedding of watermarks was performed for the individual vertex sub-series separately.

We compared the performances of the following methods. The first three methods are based on the basic SSA. In the basic SSA method, we have freedom in the choice of the order of the vertices in $\boldsymbol{F}$. In Fig. 5, we show three kinds of the vertex series. The vertex series of Fig. 5 (a) and (d) are



(a) (b)

Figure 4: Partition of the vertices. (a) and (b) are the bunny and dragon meshes obtained by partitioning the vertices into 5 groups of the same size according to the Voronoi regions.

obtained by solving TSP (Traveling Salesman Problems) where the metric is the Euclidean norm for the vertices in the respective regions; we call the watermarking method with this order "Euclidean norm". The vertex series of Fig. 5 (b) and (e) are obtained by choosing vertices in random order; we call the method using this order "Random order". The vertex series of Fig. 5 (c) and (f) are obtained by solving TSP where the metric is the inner product for the vertices in the respective regions; we call the method using this order "Inner product".

The forth method is the Laplacian matrix method proposed by Ohbuchi et al. [13]. We call this method "Ohbuchi's method".

The fifth method is based on our generalized SSA. We call this method "Generalized SSA". In this method, we use the linear operator $\boldsymbol{A}$ constructed in subsection 3.2. The rank of the trajectory matrix is $d = \min_i \max_j D_{v_i}(v_j)$. The bunny model has $d = 21$ and the dragon model has $d = 18$. (If we want to embed more information, we need to construct the other linear operator $\boldsymbol{A}$ with a larger rank.)

In "Generalized SSA" method, we embedded 18 bits data for the bunny model and 15 bits data for the dragon model, and each bit was embedded only once (i.e., chip rate is 1). In the other method, we embedded a 50 bits data respectively 20 times (i.e., chip rate is 20) for the bunny model and 18 times (i.e., chip rate is 18) for the dragon model. If a mesh is fixed, a higher chip rate means a lower data capacity and more robustness.

The watermark embedding amplitudes α is defined as $\alpha = \beta \times l$ where l is the largest length of edge of the axis-aligned bounding box of the target mesh. In Fig. 7, the appearances for $\beta = 0.1, 1$ are presented. If the α is

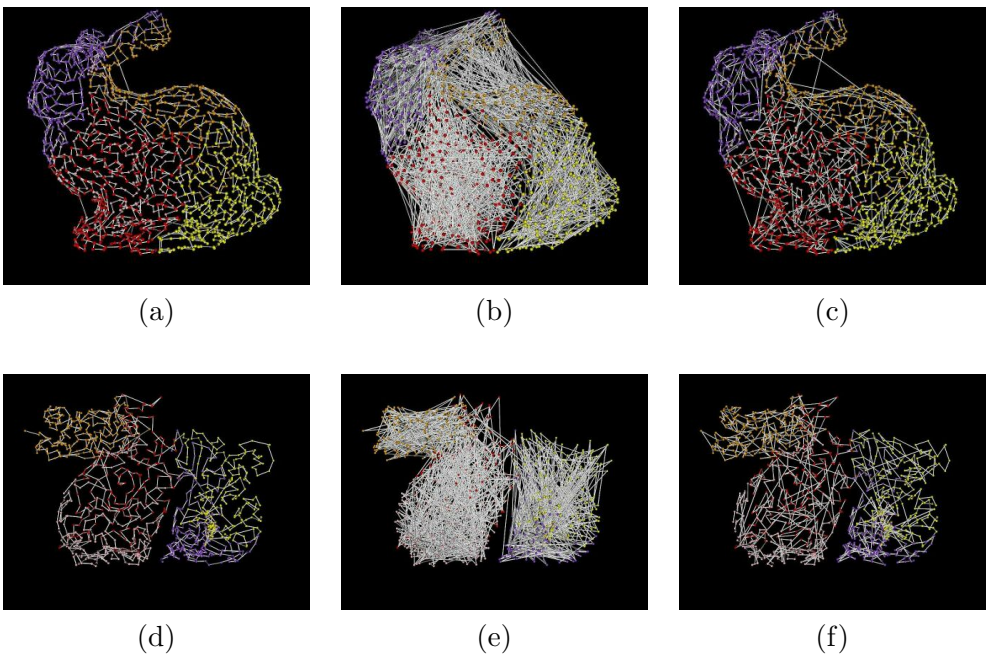


Figure 5: Three kind of the vertex series. (a) and (d) are "Euclidean norm". (b) and (e) are "Random order". (c) and (f) are "Inner product".

larger, the watermark withstands against more disturbances, (for example, adding random noises and mesh smoothing) but the shape itself is distorted.

5.2 Appearances of the Watermarked Meshes

Fig. 7 shows appearances of the watermarked meshes, where (a) and (h) show the original meshes, while (b), (c), (d), (i), (j), (k) and (e), (f), (g), (l), (m), (n) show the watermarked meshes for $\beta = 0.1$ and 1, respectively. The appearances of (b), (c), (d) or (i), (j), (k) can hardly be distinguished from the appearances of the original mesh (a) or (h). Thus they are watermarked successfully. On the other hand, the appearances of the original meshes are not preserved in (e), (f), (g) or (l), (m), (n). Thus the watermarks are too large in those cases.

In the appearances of the watermarked meshes, we cannot see much difference among the methods using the basic SSA and the method using the generalized SSA.

5.3 Robustness

We experimentally evaluate the robustness of our watermarks against the similarity transformations, uniform random noises and mesh smoothing.

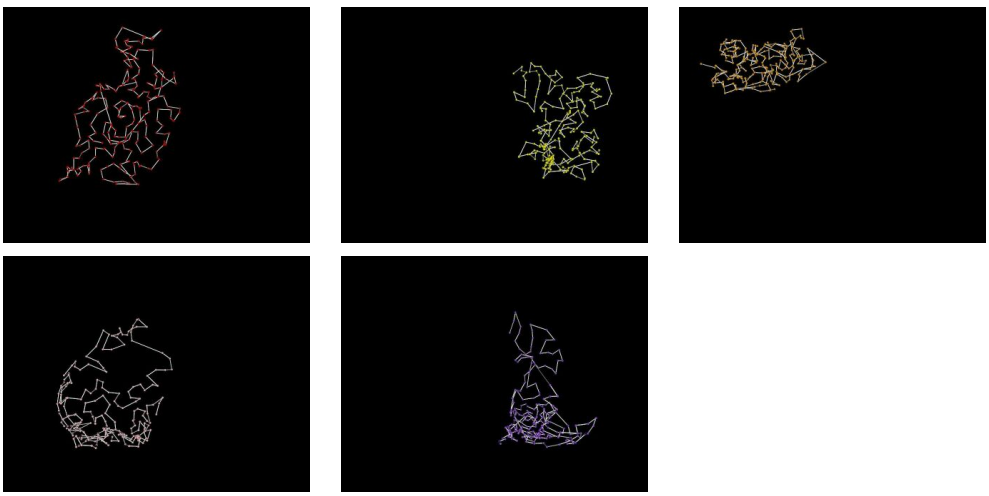


Figure 6: The respective vertex sub-series of Fig.5 (f).

5.3.1 Similarity Transformation

In the case of the dragon model mesh, when the p_i was randomly selected, the center of gravity moved by about $0.01 \times \alpha$ in the 2-norm. On the other hand, when the optimal p_i is selected by the expression (25), the center of gravity moved by about $10^{-5} \times \alpha$ in the Euclidean norm. We performed some kind of similarity transformations for the watermarked meshes, counted the number of the values answered correctly from the mesh and repeated the experiment 10 times. As the result, as shown in Table 1, we obtained the average false values of about $30\% \sim 40\%$ when the p_i was randomly selected whereas we reconstructed most of the bits correctly when the optimal p_i was selected by the expression (25). Thus, it is effective to use the optimal p_i's.

Table 1: Ratios of the correctly recovered watermarks under similarity transformations.

	Euclidean norm	
	bunny	dragon
Randomly selected p_i	30.1/50bit (60.2%)	32.3/50bit (64.6%)
Optimal selected p_i	50.0/50bit (100.0%)	50.0/50bit (100.0%)
	Generalized SSA	
	bunny	dragon
Randomly selected p_i	12.5/18bit (69.4%)	10.8/15bit (72.0%)
Optimal selected p_i	18.0/18bit (100.0%)	15.0/15bit (100.0%)

5.3.2 Uniform Random Noises

Fig. 8 shows the appearances of the watermarked mesh whose vertex coordinates were disturbed with uniform random noises with amplitude $\alpha \times \gamma$ ($\beta = 0.1$). Fig. 8 (a) and (e) are the original 3D polygonal meshes of the models, (b) and (f) are the models with uniform random noises with $\gamma = 0.01$, (c) and (g) are the models with uniform random noises with $\gamma = 0.1$, and (d) and (h) are the models with uniform random noises with $\gamma = 1$. From Fig. 8, we can see that the noises of $\gamma = 0.1$ deformed the appearances of the original meshes to a certain extent, and the noises of $\gamma = 1$ are too large to preserve the appearances of the original meshes.

We counted the number of the values answered correctly out of the inputted bit and repeated the experiment 100 times. The result is shown in Table 2. From this experiment, we can see that the watermark can withstand against uniform noises for $\gamma \leq 0.01$, as shown in Table 2. Moreover, we cannot see much difference among these methods.

In the basic SSA method [8] and in Ohbuchi's method [13], the same bit b_i was embedded many times (20 and 18 times, for example) because each bit is very fragile. On the other hand, in the proposed method, each bit is embedded only once, but still the watermark can be reconstructed almost in the same accuracy as the other methods, as shown in Table 2. In this sense, the proposed watermark method is very robust against random noises.

Table 2: Ratios of the correctly recovered watermarks under random noises.

	bunny		
	$\gamma = 1$	$\gamma = 0.1$	$\gamma = 0.01$
Euclidean norm	21.14/50bit (42.28%)	45.47/50bit (90.94%)	50.00/50bit (100.00%)
Random order	22.53/50bit (45.06%)	48.97/50bit (97.94%)	50.00/50bit (100.00%)
Inner product	18.07/50bit (36.14%)	49.38/50bit (98.76%)	50.00/50bit (100.00%)
Ohbuchi's method	32.31/50bit (64.62%)	49.13/50bit (98.26%)	50.00/50bit (100.00%)
Generalized SSA	11.08/18bit (61.55%)	17.31/18bit (96.16%)	18.00/18bit (100.00%)
	dragon		
	$\gamma = 1$	$\gamma = 0.1$	$\gamma = 0.01$
Euclidean norm	28.14/50bit (56.28%)	47.77/50bit (95.54%)	50.00/50bit (100.00%)
Random order	26.73/50bit (53.46%)	48.02/50bit (96.04%)	50.00/50bit (100.00%)
Inner product	26.43/50bit (52.86%)	48.76/50bit (97.52%)	50.00/50bit (100.00%)
Ohbuchi's method	29.59/50bit (59.18%)	47.13/50bit (94.26%)	50.00/50bit (100.00%)
Generalized SSA	9.13/15bit (60.86%)	14.22/15bit (94.80%)	15.00/15bit (100.00%)

5.3.3 Mesh Smoothing

This experiment was done for "Euclidean norm" method and "Random order" method. Fig. 9 shows the appearances of the mesh resulted from once, twice and thrice applications of Taubin's smoothing filter [15] to the watermarked mesh. When Taubin's smoothing filter is used for a mesh, the mesh is rounded in the sense of the low-pass filter. Since the details of the meshes are smoothed out, even though 50 bits data were embedded in all spectral domains, we could not extract the watermark data correctly, as shown in Table 3. But, when 10 bits data were embedded in the only low spectral domains, we could extract the watermark data except in some circumstances. The exceptional circumstances are that the shape of mesh is complex and the vertex series oscillate wildly. Since the dragon model is more complex than the bunny model and the vertex series chosen in random order oscillate wilder than the vertex series obtained by solving TSP, 10 bits data embedded in the only low spectral domain of the dragon model using the vertex series by choosing the vertices in random order could not be extracted correctly, as shown in Table 3.

Table 3: Numbers of the correctly recovered watermarks under mesh rounding.

	bunny			
	Euclidean norm		Random order	
	50bits	10bits	50bits	10bits
First	50	10	50	10
Second	50	10	46	10
Third	44	10	43	10
	dragon			
	Euclidean norm		Random order	
	50bits	10bits	50bits	10bits
First	49	10	50	10
Second	45	10	50	8
Third	42	10	40	6

6 Summary and Future Work

We generalized the basic SSA in a way suitable for the 3D polygonal meshes, and on the basis of this generalized SSA, we proposed a new method of watermarking for the 3D polygonal meshes. The watermark embedded by our

method is robust against similarly transformations and moderate uniform noises added to vertex coordinates.

In the generalized SSA, since the rank of the trajectory matrix is $d = \min_i \max_j D_{v_i}(v_j)$, the number of the bases obtained by the spectral decomposition is not large. Recall that $d = 21$ for the bunny model and $d = 18$ for the dragon model, while the rank of the trajectory matrix for the basic SSA was more than 1000. From this reason, we can embed only a limited number of information as watermarking. However, each embedded bit is robust, we do not need to embed the same watermarking information redundantly.

The 3D polygonal mesh represents the boundary of the three-dimensional region, and this boundary is two-dimensional manifold. However, it is not easy to globally parameterize any two-dimensional manifold. Note that the generalized SSA proposed in this paper dose not require any parameterization of the boundary. On the other hand, the traditional of the multidimensional spectral decompositions such as the multidimensional Fourier transformation and the multidimensional wavelet transformation require the parameterization of the boundary, and hence cannot be used for general 3D polygonal meshes. Therefore, the generalized SSA can be a powerful new tool for the analysis of polygonal meshes.

Since the spectral decomposition of a mesh using the Laplacian matrix in [13] requires the eigenvalue decomposition of a matrix where rank is the number of the vertices, the calculation cost is large and the method cannot be applied to huge meshes. On the other hand, since our generalized SSA requires the eigenvalue decomposition of a small the matrix whose rank is determined by the linear operator (e.g., 21 for the bunny model and 18 for the dragon model), the calculation cost is small, and consequently our generalized SSA method can be applied to huge meshes.

There is large freedoms in the choice of the linear operator $\boldsymbol{A}$. Therefore, one of our future work is to search for other choices of $\boldsymbol{A}$ that are sitable to watermarking. Moreover, the generalized SSA may have many applications other than the watermarking. It is also our future work to develop new application area of the generalized SSA.

Acknowledgement

This work is partly supported by the 21st Century COE Program on Information Science and Technology Strategic Core, and Grant-in-Aid for Scientific Research (S) of the Japanese Ministry of Education, Culture, Sports, Science and Technology.

References

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

[2] Cox, I., J., Killian, J., Leighton, T. and Shamoon, T., Secure spread spectrum watermarking for multimedia. *IEEE Transactions on Image Processing 12, 6* pp. 1673-1687, 1997.

[3] Galka, A., *Topics in Nonlinear Time Series Analysis*, World Scientific, pp. 49-71, 2001.

[4] Golyandina, N., Nekrutkin, V. and Zhigljavsky, A., *Analysis of Time Series Structure–SSA and Related Techniques*, Chapman & Hall/CRC, 2001.

[5] Guskov, I., Sweldens, W. and Shroder, P., Multiresolution signal processing for meshes, *Proceedings SIGGRAPH99*, pp. 49-56, 1999.

[6] Hoppe, H., Progressive meshes. *ACM SIGGRAPH 96 Conference Proceedings*, pp. 99-108, August 1996.

[7] Kanai, S., Date, H. and Kishinami,T., Digital Watermarking for 3D Polygons Using Multiresolution Wavelet Decomposition, *Proceedings of the Sixth IFIP WG 5.2 International Workshop on Geometric Modeling: Fundamentals and Applications (GEO-6)*, pp. 296-307, Tokyo, Japan, December 1998.

[8] Murotani, K. and Sugihara, K., Watermarking 3D Polygonal Meshes Using the Singular Spectrum Analysis, *Proceedings of the 10th IMA International Conference on The Mathematics of Surfaces*, pp. 85-98, Leeds, UK, September, 2003.

[9] Ohbuchi, R., Masuda, H. and Aono, M., Watermarking Three-Dimensional Polygonal Models, *Proceedings of the ACM International Conference on Multimedia '97*, pp. 261-272, Seattle, USA., November, 1997.

[10] Ohbuchi, R., Masuda, H. and Aono, M., Watermarking Three-Dimensional Polygonal Models Through Geometric and Topological Modifications, *IEEE Journal on Selected Areas in Communication*, Vol. 16, No. 4, pp. 551-560, May, 1998.

[11] Ohbuchi, R., Masuda, H. and Aono, M., Targeting Geometrical and Non-Geometrical Components for Data Embedding in Three-Dimensional Polygonal Models, *Computer Communications*, Vol. 21, pp. 1344-1354, October, 1998.

[12] Ohbuchi, R., Masuda, H. and Aono, M., A Shape-Preserving Data Embedding Algorithm for NURBS Curves and Surfaces, *Proceedings of the Computer Graphics International'99*, pp. 180-177, Canmore, Canada, June 7-11, 1999.

[13] Ohbuchi, O., Takahashi, S., Miyazawa, T. and Mukaiyama, A., Watermarking 3D Polygonal Meshes in the Mesh Spectral Domain, *Proceedings of the Graphics Interface 2001*, pp. 9-17, Ontario, Canada, June 2001.

[14] Praun, E., Hoppe, H. and Finkelstein, A., Robust Mesh Watermarking, *ACM SIGGRAPH 1999*, pp. 69-76, 1999.

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

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

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

[18] Matsui, K., *Basic of watermarks* (in Japanese), Morikita Shuppan Publishers, Tokyo, 1998.

[19] Yin, K., Pan, Z., Shi, J. and Zhang, D., Robust mesh watermarking based on multiresolution processing, *Computers & Graphics, Vol. 25*, pp. 409-420, 2001.

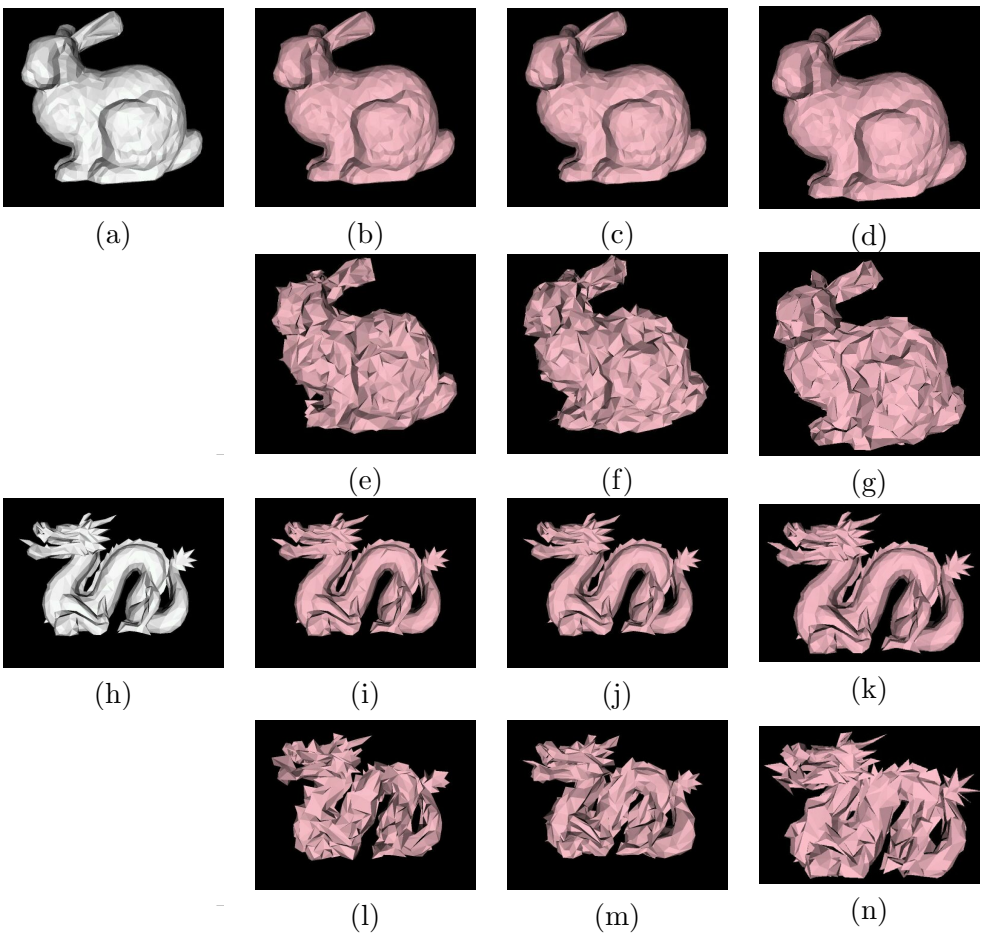
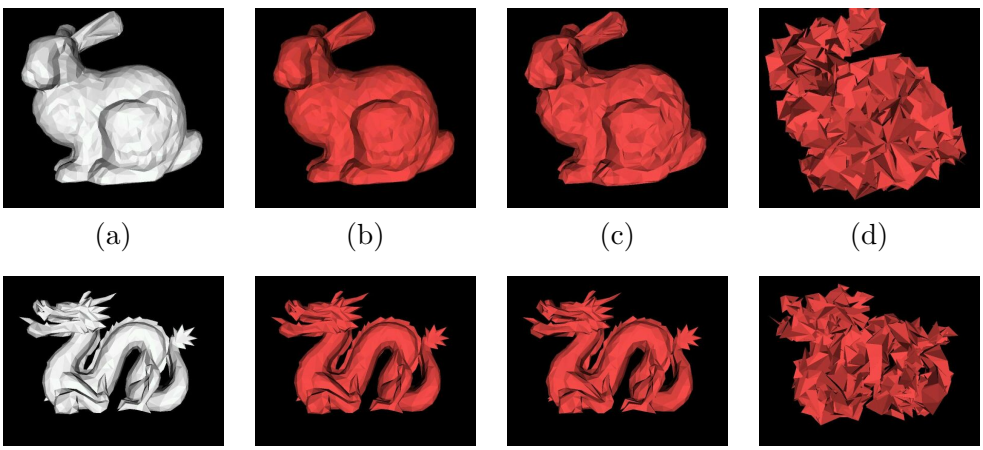


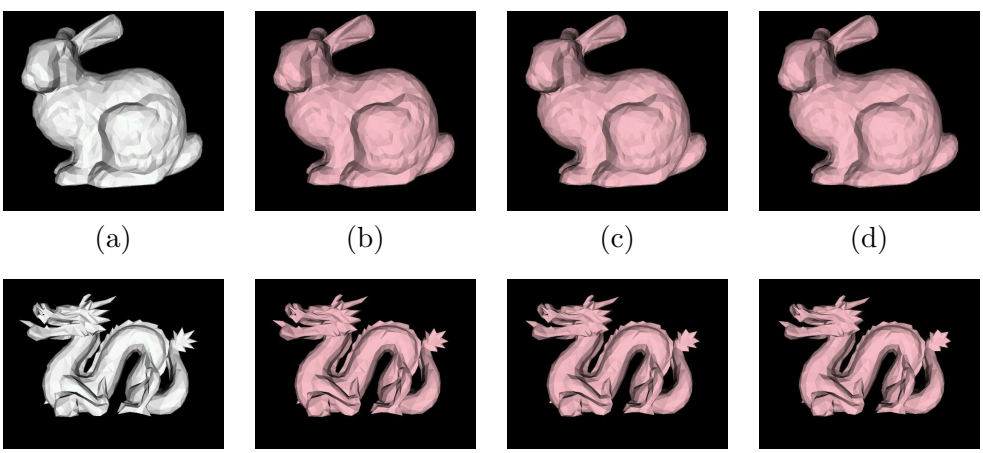
Figure 7: The original meshes and the watermarked meshes. (a) and (h) are the original 3D polygonal meshes. (b) and (i) are the watermarked meshes with $\beta = 0.1$ using "Euclidean norm". (e) and (l) are the watermarked meshes with $\beta = 1$ using "Euclidean norm". (c) and (j) are the watermarked meshes with $\beta = 0.1$ using "Random order". (f) and (m) are the watermarked meshes with $\beta = 1$ using "Random order". (d) and (k) are the watermarked meshes with $\beta = 0.1$ using "Generalized SSA." (g) and (n) are the watermarked meshes with $\beta = 1$ using "Generalized SSA".



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 8: The bunny and dragon models to which uniform random noises with amplitude $\alpha \times \gamma$ ($\beta = 0.1$) are added. (a) and (e) are the original 3D polygonal meshes of the models, (b) and (f) are the models with uniform random noises with $\gamma = 0.01$, (c) and (g) are the models with uniform random noises with $\gamma = 0.1$, and (d) and (h) are the models with uniform random noises with $\gamma = 1$.



(a) (b) (c) (d)

(e) (f) (g) (h)

Figure 9: The meshes resulted from once, twice and thrice applications of Taubin's smoothing filter [15] to the watermarked mesh. (a) and (e) are the original meshes. (b) and (f) are the appearances of the mesh resulted from once applications of Taubin's smoothing filter. (c) and (g) are the appearances of the mesh resulted from twice applications of Taubin's smoothing filter. (d) and (h) are the appearances of the mesh resulted from thrice applications of Taubin's smoothing filter.