

A Digital Watermarking Scheme for Polygonal Meshes using Modified Spectral Decomposition

Jung-Kyo Sohn*, Hyeong-In Choi*, Tae-wan Kim**, Song-Hwa Kwon***,
Sang-Hun Park*, Heon-Ju Shin****

*Dept. of Mathematics, Seoul National University, Korea

**Dept. of Naval Architecture and Ocean Engineering, Seoul National University, Korea

***Statistical Research Center for Complex Systems, Seoul National University, Korea

****Dept. of Computational Science and Technology, Seoul National University, Korea

E-mail: taewan@snu.ac.kr

ABSTRACT

Already used to establish the copyright of graphics, audio and text, digital watermarking is now becoming important for the protection of geometric data. Watermarking polygonal models in the spectral domain gives protection against a range of attacks. We generalize an existing spectral watermarking scheme and propose a new technique based on this generalization. While inserting the watermark, we avoid the cost of finding the eigenvalues and eigenvectors of a Laplacian matrix; instead we use linear operators derived by scaling functions. Experimental results show how the cost of inserting and detecting watermarks can be traded against robustness under attack.

Keywords

Digital watermarking, 3D mesh processing, digital signal processing, spectral decomposition

1. INTRODUCTION

Because of the rapid development of computers and the wide adaption of high-speed communication networks, audio, image, and video content in digital form are now ubiquitous. Digital data can be saved and edited much more easily than analogue data. Protecting proprietary rights becomes much more problematic when digital content can easily be disseminated through communication channel such as the internet [13] [6].

Digital watermarking is a technical way to protect the copyright on digital contents. Copyright information (the mark) by making changed to the image, video or audio contents so minute that they are not apparent to the consumer. Nevertheless the inserted information can be extracted when required.

For many years, the watermarking technology for images, video and audio is already well developed. There has been much less research on the watermarking of 2D and 3D geometrical models, composed of objects such as curves or surfaces. In the last few

years, the interest in this topic has quickened. One driver for this research is the requirement protect three-dimensional games. Another concern is the increase in cooperative CAD modelling over the internet. Digital watermarking technology for the three-dimensional polygon models and other CAD data has been announced by several researchers. Ohbuchi et al [10] [8] [11] [9], Benedens [3], Wagner [17] and Praun et al [14] have presented methods of inserting watermarks into three-dimensional polygon models. Ko et al [7] recently described methods for matching and similarity evaluation of two NURBS surfaces, and outlined their application to the copyright protection of digital data representing NURBS surfaces, which are the standard in CAD models.

Ohbuchi et al [12] have also proposed a watermarking scheme for 3D polygonal meshes in the spectral domain instead of the spatial domain. The spectral domain of a mesh can be characterized by the eigenvalues and their corresponding eigenvectors of the *Laplacian matrix* [4], which captures the topological structure of the mesh. This matrix, Δ , contains spectral information about the mesh M in its eigenvalues and eigenvectors. But the eigenvalue problem, whether the matrix is symmetric or not, may incur stability problems [2] [5]. In particular, the stability of eigenvectors relies heavily on the nearness of the eigenvalues. If two eigenvalues are very close to each other, a small perturbation of the matrix can lead to a large change in its eigenvectors. Since a watermarking scheme should work on any independent mesh, and it is reasonable expect every matrix Δ to have a nice eigenvalue distribution.

Taubin [16] proposed a fairing algorithm using simple linear operators, which are polynomials of the matrix Δ of the mesh M . When applied repeatedly to a mesh signal, the operators progressively attenuate the power of the components of high frequencies of the mesh signals that belong to the spatial domain of M . The essence of Taubin's idea is that we can approximate the operations on the spectra of a mesh by using appropriate linear operators in the spatial domain.

Following the line of Taubin's work, we will first provide a generalization of existing watermark schemes such as the one described in Ohbuchi et al [12] so that we can avoid known problems such as the stability of eigenvalues and eigenvectors. We will then show that Ohbuchi et al's scheme is a special case of our general framework.

Our scheme is based on linear operators, which are polynomials of the matrix Δ of M . We then form products of these linear operators and signal vectors by successive matrix-vector multiplication. Elementary calculations such as matrix-vector and scalar-vector multiplication, if carefully done, are fast and stable.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$5.00.

The rest of this paper is organized as follows. In Section 2, we describe how to extend signal processing to signals defined on polygonal meshes of arbitrary topology, and we give a generalization of the existing polygonal mesh watermarking schemes in the mesh spectral domain. In Section 3, we propose a new watermarking scheme using scaling functions, based on the generalization in Section 2. In Section 4, we present experimental results. Finally, in Section 5 we make some closing remarks.

2. BACKGROUND

2.1 Signal processing model for polygonal meshes

Given a mesh model M , we can think of it as a graph $G_M = (V, E)$, where $V = \{v_1, v_2, \dots, v_N\}$ is the vertex set and $E = \{e_{ij} = (v_i, v_j)\}$ is the edge set of the mesh M . The edge set E together with V contains the topological contents (i.e. connectivity information) of the mesh M , and from E we can derive the *Laplacian operator*, which will be explained in detail later. Let the surface signal set $S = \{s : V \rightarrow \mathbb{R}\}$ be a set of real-valued functions defined on the vertex-set V of the mesh M . For a fixed vertex-set V , we can view each $s \in S$ as a vector $\mathbf{s} = (s(v_1), s(v_2), \dots, s(v_N))^T \in \mathbb{R}^N$. We will call \mathbf{s} a *signal vector* of the mesh M (in the spatial domain). If $(x, y, z) \in \mathbb{R}^3$ is a realization of a vertex $v \in V$ in \mathbb{R}^3 , let $x(v), y(v)$ and $z(v)$ be the functions in S which map each $v \in V$ to x, y , and z respectively. We will denote their vector analogs (signal vectors) by \mathbf{x}, \mathbf{y} , and \mathbf{z} respectively.

2.2 Generalized watermark scheme

2.2.1 Partition of the identity operator

Suppose that we want to decompose a given signal vector \mathbf{x} as follows:

$$\mathbf{x} = c_1 \mathbf{x}_1 + c_2 \mathbf{x}_2 + \dots + c_n \mathbf{x}_n, \quad (c_i \in \mathbb{R}, \|\mathbf{x}_i\| = 1), \quad (1)$$

subject to the following **conditions**:

1. there exists a set $\{L_1, L_2, \dots, L_n\}$ of linear operators such that each $L_i(\mathbf{x}) = c_i \mathbf{x}_i$, $i = 1, 2, \dots, n$,
2. $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ are linearly independent,
3. if $i \neq j$, then for a sufficiently small $\epsilon \geq 0$, $|\langle \mathbf{x}_i, \mathbf{x}_j \rangle| \leq \epsilon$, and
4. for $i < j$, \mathbf{x}_j is the part of \mathbf{x} with a higher frequency (in some sense, which will be clear later) than \mathbf{x}_i .

If the choice of the set $\{L_1, L_2, \dots, L_n\}$ can be made entirely independent of \mathbf{x} , we call $\{L_1, L_2, \dots, L_n\}$ a *partition of the identity operator*, since

$$L_1 + L_2 + \dots + L_n = I.$$

But note that this partition depends on the topological structure of the mesh model M .

It can be proved that the Condition 2 follows from Condition 3 for $0 \leq \epsilon \leq 1/n$. We have introduced Condition 4 to allow for possible smoothing attacks on the watermark. Condition 3 makes the following analysis easier; but experiments show that this condition may be relaxed to some extent.

The mapping $\mathbf{x} \mapsto (c_1, c_2, \dots, c_n)$ can be thought of as a generalized Fourier transform, and c_1, c_2, \dots, c_n as the Fourier coefficients of \mathbf{x} . We do not need to compute the coefficients explicitly in our scheme. Only the linear operators L_1, L_2, \dots, L_n are needed to achieve our goal.

As we shall see, one of the most obvious ways to decompose \mathbf{x} as set out in Equation(1) is to represent \mathbf{x} as a linear combination of the eigenvectors of a Laplacian matrix, an approach which will be discussed later. Thus, the watermark scheme described in Ohbuchi et al [12] can be shown to be a special case of our general framework.

2.2.2 Watermark insertion and extraction

Let $\mathbf{p} = (p_j)_{j=1,2,\dots,n} \in \{-1, 1\}^N$ be a pseudo-random sequence. Suppose that a given signal vector \mathbf{x} has been decomposed as in Equation(1). To embed a watermark bit $\alpha(1 \text{ or } -1)$, we define a watermark vector as follows:

$$\mathbf{w} = \sum_{j=1}^n (\alpha \cdot \beta \cdot p_j) \mathbf{x}_j,$$

where β is a scaling parameter. We then add this watermark vector \mathbf{w} to the original vector \mathbf{x} and publish the resultant signal vector

$$\mathbf{x}^w = \mathbf{x} + \mathbf{w}. \quad (2)$$

Suppose we are given a suspicious signal vector \mathbf{x}^a thought to be a version of \mathbf{x} that has been attacked with the intent of obliterating the watermark. After subtracting the watermarked signal vector \mathbf{x}^w from \mathbf{x}^a , we denote the remainder by \mathbf{a} , which can be thought of as the signal vector of the attack. We then define the *watermark detection value* $W(\mathbf{x}, \alpha)$ of α as follows:

$$W(\mathbf{x}, \alpha) = \left\langle \mathbf{x}^a - \mathbf{x}, \sum_{j=1}^n p_j \mathbf{x}_j \right\rangle.$$

The sign of $W(\mathbf{x}, \alpha)$ completely determines the embedded watermark bit α .

$$\begin{aligned} & \left\langle \mathbf{x}^a - \mathbf{x}, \sum_{j=1}^n p_j \mathbf{x}_j \right\rangle \\ &= \left\langle \sum_{j=1}^n (\alpha \cdot \beta \cdot p_j \mathbf{x}_j + \mathbf{a}), \sum_{j=1}^n p_j \mathbf{x}_j \right\rangle \\ &= \alpha \cdot \beta \left\langle \sum_{j=1}^n p_j \mathbf{x}_j, \sum_{j=1}^n p_j \mathbf{x}_j \right\rangle + \left\langle \mathbf{a}, \sum_{j=1}^n p_j \mathbf{x}_j \right\rangle \end{aligned}$$

The first term on the right hand side can be estimated as

$$\begin{aligned} & \left| \left\langle \sum_{j=1}^n p_j \mathbf{x}_j, \sum_{j=1}^n p_j \mathbf{x}_j \right\rangle_D \right| \\ &= \left| n + \sum_{i \neq j} p_i p_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \right| \\ &\geq n - \epsilon(n^2 - n). \end{aligned}$$

And, the second term can be estimated as

$$\begin{aligned} & \left| \left\langle \mathbf{a}, \sum_{j=1}^n p_j \mathbf{x}_j \right\rangle \right| \\ &= \left| \left\langle \sum_{j=1}^n a_j \mathbf{x}_j, \sum_{j=1}^n p_j \mathbf{x}_j \right\rangle \right| \\ &\leq \left| \sum_{j=1}^n p_j a_j \right| + \epsilon(n-1) \|\mathbf{a}\|_1, \end{aligned}$$

where $\|\cdot\|_1$ is the usual 1-norm.

The properties of a pseudo-random sequence are such that, for most $\mathbf{a} = \sum_{i=1}^n a_i \mathbf{x}_i$, the sum $\sum_{j=1}^n p_j a_j$ may be expected to be much smaller than n . The smaller the value of ϵ , the higher the probability of detecting the right watermark bit embedded in \mathbf{x} , because it is more likely that the first term dominates the value of $W(\mathbf{x}, \alpha)$. Note that, if we set $\mathbf{x}_j = \phi_j$, $j = 1, 2, \dots, N$ to be the eigenvector of Δ , we get the scheme described in [12].

Let $\mathbf{q} = (q_1, q_2, \dots, q_n)$ be another pseudo-random sequence, such that

$$\sum_{j=1}^n p_j q_j = 0.$$

Then the inner product $\langle \sum p_j \mathbf{x}_j, \sum q_j \mathbf{x}_j \rangle$ can be estimated as

$$\begin{aligned} & \left| \left\langle \sum_{j=1}^n p_j \mathbf{x}_j, \sum_{j=1}^n q_j \mathbf{x}_j \right\rangle \right| \\ & \leq \left| \sum_{j=1}^n p_j q_j \right| + \epsilon(n^2 - n) = \epsilon(n^2 - n). \end{aligned}$$

Thus, if we have embedded k bits $(\alpha_1, \alpha_2, \dots, \alpha_k)$ into \mathbf{x} , the watermark detection value $W(\mathbf{x}, \alpha_i)$ can be estimated as follows:

$$\begin{aligned} |W(\mathbf{x}, \alpha_i)| &= \left| \left\langle \mathbf{x}^a - \mathbf{x}, \sum_{j=1}^n p_j^{(i)} \mathbf{x}_j \right\rangle \right| \\ &\geq n - \left| \sum_{j=1}^n p_j^{(i)} a_j \right| - \epsilon((n-1)\|\mathbf{a}\|_1 + k(n^2 - n)), \end{aligned} \quad (3)$$

where $\mathbf{p}^{(i)} = (p_1^{(i)}, p_2^{(i)}, \dots, p_n^{(i)})$ is the pseudo-random sequence used in embedding the bit α_i into \mathbf{x} . The inequality in Equation(3) is a very conservative estimate. Even if it becomes an equality, there is still a high probability of detecting the correct bit sequence $(\alpha_1, \alpha_2, \dots, \alpha_k)$, as long as we can obtain signal vectors $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n$ for which ϵ is less than $1/kn^2$.

3. WATERMARKING POLYGONAL MESHES

3.1 The Laplacian Matrix

An operator \mathcal{L} on a surface signal set S is called a *Laplacian operator* if it satisfies the following equation:

$$\mathcal{L}s(v_i) = \sum_{(v_i, v_j) \in E} w_{ij}(s(v_j) - s(v_i)), \quad s \in S, \quad (4)$$

where the weights w_{ij} are non-negative real numbers and $\sum_{(v_i, v_j) \in E} w_{ij} \neq 0$, let W_i be the eigenspace of Δ corresponding to the eigenvalue λ_i , and let Δ_i be the orthogonal projection on W_i . Then the following are true.

$$\Delta \mathbf{s} = (I - W) \mathbf{s}, \quad (5)$$

where $W = (w_{ij})_{1 \leq i, j \leq N} \in \mathbb{R}^{N^2}$, and Δ is an $N \times N$ matrix. The matrix form Δ of \mathcal{L} is called a *Laplacian Matrix*. Since \mathbf{s} is arbitrary, Δ is given by

$$\Delta = I - W.$$

We denote the valence of the vertex $v \in V$ by $d(v)$. The we set $w_{ij} = 1/d(v_i)$ for $(v_i, v_j) \in E$, and $w_{ij} = 0$ otherwise. Since W is not in general symmetric, we cannot say that the eigenvalues of the Laplacian matrix Δ are all real. However, by choosing an appropriate weight matrix W we can make the spectrum of the Laplacian, $\sigma(\Delta)$, lie in the real line \mathbb{R} . Let $\lambda_1 \leq \lambda_2 \leq \dots \leq \lambda_N$

be the N eigenvalues (not necessarily distinct) of Δ and let $\Phi = \{\phi_1, \phi_2, \dots, \phi_n\}$ be the corresponding eigenvectors.

We now review a few useful facts:

- $\Lambda \subset [0, 2]$.
- $\lambda_1 + \lambda_2 + \dots + \lambda_n = n$.
- The first eigenvalue and its corresponding eigenvector are $\lambda_1 = 0$, and $\phi_1 = \mathbf{1} = (1, 1, \dots, 1)^T$.

Although Φ is linearly independent, and hence a basis of the vector space \mathbb{R}^N , Φ is not an orthogonal basis with respect to the standard inner product. A slight modification of the standard inner product $\langle \cdot, \cdot \rangle$ cures this problem. We define this new inner product on \mathbb{R}^N as follows:

$$\langle \mathbf{x}, \mathbf{y} \rangle_D = \mathbf{x}^T D \mathbf{y}, \quad \mathbf{x}, \mathbf{y} \in \mathbb{R}^N, \quad (6)$$

where D is a diagonal matrix whose diagonal entries are $1/d(v_j)$, $j = 1, 2, \dots, N$. The norm $\|\cdot\|_D$ can now be derived from $\langle \cdot, \cdot \rangle_D$ as follows:

$$\|\mathbf{x}\|_D = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle_D}.$$

We are now able to derive the following lemma.

LEMMA 3.1. *A Laplacian matrix Δ is a self-adjoint operator with respect to the modified inner product $\langle \cdot, \cdot \rangle_D$. Indeed, for any $\mathbf{x}, \mathbf{y} \in \mathbb{R}^N$,*

$$\langle \mathbf{x}, \Delta \mathbf{y} \rangle_D = \langle \Delta \mathbf{x}, \mathbf{y} \rangle_D. \quad (7)$$

Thus, if we equip the vector space \mathbb{R}^N with the new inner product $\langle \cdot, \cdot \rangle_D$, we get the following theorem.

THEOREM 3.2. *The set $\Phi = \{\phi_j | j = 1, 2, \dots, N\}$ of eigenvectors of a Laplacian matrix Δ forms an orthogonal (after normalization, orthonormal) basis of \mathbb{R}^N .*

Using Theorem (3.2), any $\mathbf{x} \in \mathbb{R}^N$ can be written uniquely as the linear combination of the elements of Φ :

$$\begin{aligned} \mathbf{x} &= \mathbf{x}_1 + \mathbf{x}_2 + \dots + \mathbf{x}_N \\ &= \langle \mathbf{x}, \phi_1 \rangle_D \phi_1 + \langle \mathbf{x}, \phi_2 \rangle_D \phi_2 + \dots + \langle \mathbf{x}, \phi_N \rangle_D \phi_N, \end{aligned}$$

where $\mathbf{x}_i = \langle \mathbf{x}, \phi_i \rangle_D \phi_i$, $i = 1, 2, \dots, N$. This decomposition satisfies Conditions 2, 3 and 4. The next theorem will demonstrate that the decomposition also satisfies Condition 1.

THEOREM 3.3 (THE SPECTRAL THEOREM). *For $i(1 \leq i \leq N)$, let W_i be the eigenspace of Δ corresponding to the eigenvalue λ_i , and let Δ_i be the orthogonal projection on W_i . Then the following are true.*

1. $\mathbb{R}^N = W_1 \oplus \dots \oplus W_N$.
2. For $1 \leq i \leq N$, $\Delta_i = \phi_i \phi_i^T D$ and is self-adjoint with respect to the modified inner product $\langle \cdot, \cdot \rangle_D$.
3. For $1 \leq i, j \leq N$, $\Delta_i \Delta_j = \delta_{ij} \Delta_i$.
4. $I = \Delta_1 + \dots + \Delta_N$.
5. $\Delta = \lambda_1 \Delta_1 + \dots + \lambda_N \Delta_N$.

From Theorem (3.3), each $\mathbf{x}_i = \Delta_i(\mathbf{x})$, and hence the decomposition satisfies Condition 1, since we have found a family of operators $\{\Delta_i\}$.

3.2 Linear operators

Given a polynomial f and any square matrix M , the matrix $f(M)$ is well-defined. In our case, the matrix $f(\Delta)$ is a *linear operator* which maps one signal vector to another.

3.2.1 Bandpass filters

If f is a polynomial defined on $[0, 2]$, then for a Laplacian matrix Δ and a signal vector \mathbf{x} , we have

$$\begin{aligned} f(\Delta)\mathbf{x} &= f(\Delta)(\langle \mathbf{x}, \phi_1 \rangle_D \phi_1 + \cdots + \langle \mathbf{x}, \phi_N \rangle_D \phi_N) \\ &= \langle \mathbf{x}, \phi_1 \rangle_D f(\Delta)\phi_1 + \cdots + \langle \mathbf{x}, \phi_N \rangle_D f(\Delta)\phi_N \\ &= \langle \mathbf{x}, \phi_1 \rangle_D f(\lambda_1)\phi_1 + \cdots + \langle \mathbf{x}, \phi_N \rangle_D f(\lambda_N)\phi_N. \end{aligned}$$

Suppose that the value of a polynomial f is 1 at an eigenvalue λ of Δ , and 0 at the other eigenvalues. The existence of this polynomial f is always guaranteed by the Lagrange interpolation formula. For instance, we can write

$$f_\lambda(x) = \frac{\prod_{\substack{\mu \in \sigma(\Delta) \\ \mu \neq \lambda}} (x - \mu)}{\prod_{\substack{\mu \in \sigma(\Delta) \\ \mu \neq \lambda}} (\lambda - \mu)}.$$

for this polynomial f , we have

$$f_\lambda(\Delta)\mathbf{x} = c\phi(\lambda), \quad (8)$$

where $\phi(\lambda)$ is the corresponding eigenvector of λ and $c = \langle \mathbf{x}, \phi(\lambda) \rangle$. Since each signal vector has discrete frequencies, we can regard the linear operator $f(\Delta)$ as a narrow-bandpass filter that passes only vectors of frequency λ . Furthermore, if we multiply by the vector \mathbf{x}^T on both sides, we get

$$c^2 = \langle f_\lambda(\Delta)\mathbf{x}, \mathbf{x} \rangle_D. \quad (9)$$

The implication of this result is that it is not necessary to calculate the eigenvectors of Δ in order to find the square of the spectral coefficient. Instead, we can use elementary operations such as matrix-matrix multiplication and scalar-matrix multiplication.

However, we note that the polynomial f_λ depends heavily on the spectrum $\sigma(\Delta)$. As we already mentioned, our goal is to find an operator that only passes signal vectors with a certain frequency band, without calculating the eigenvalues of Δ directly. We want this operator to be able to deal with as large a class of Laplacian matrices as possible. Thus we search for operators that are independent of the spectrum of a specific Laplacian matrix Δ . We know that the spectrum of all Laplacian matrices will lie in the interval $[0, 2]$.

3.2.2 Linear operators generated by polynomial families

Suppose that a family of functions $\{f_j\}_{j \in J}$ defined on $[a, b]$ satisfies the following condition:

$$\sum_{j \in J} f_j \equiv 1.$$

In this case, we say that $\{f_j\}_{j \in J}$ is a partition of unity. For example, a family of Bernstein polynomials

$$\left\{ \frac{n!}{j!(n-j)!} t^j (1-t)^{n-j} \right\}_{j=0, \dots, n}$$

is a partition of unity on $[0, 1]$. In particular, it a family of polynomials $\{f_j\}_{j \in J = \{1, 2, \dots, N\}}$ consists only of polynomials, then, for a Laplacian matrix Δ and a signal vector \mathbf{x} , all matrices $f_j(\Delta)$ are

well-defined and satisfy the following equality:

$$\begin{aligned} \mathbf{x} &= \sum_{j=1}^N f_j(\Delta)\mathbf{x} \\ &= f_1(\Delta)\mathbf{x} + f_2(\Delta)\mathbf{x} + \cdots + f_N(\Delta)\mathbf{x}. \end{aligned} \quad (10)$$

If f_j is a polynomial for which $f_j(\lambda_k) = \delta_{jk}$ (such as f_λ in the previous section) it follows that

$$\begin{aligned} f_j(\Delta)\mathbf{x} &= f_j(\Delta)(c_1\phi_1 + c_2\phi_2 + \cdots + c_n\phi_n) \\ &= c_j\phi_j. \end{aligned}$$

Equation(10) therefore gives us the decomposition of \mathbf{x} :

$$\begin{aligned} \mathbf{x} &= f_1(\Delta)\mathbf{x} + f_2(\Delta)\mathbf{x} + \cdots + f_N(\Delta)\mathbf{x} \\ &= c_1\phi_1 + c_2\phi_2 + \cdots + c_N\phi_N. \end{aligned}$$

3.3 A special function family: scaling functions

We will now present special functions called *scaling functions*, which appear frequently in wavelet theory.

3.3.1 Orthogonal polynomials

Let V_n be the vector space of polynomials of degree less than $n+1$. This $(n+1)$ -dimensional vector space has a basis $\{1, x, x^2, \dots, x^n\}$. For any $f, g \in V_n$, we define an inner product on V_n as follows:

$$\langle f, g \rangle = \int_{-1}^1 f(x)g(x)w(x)dx, \quad (11)$$

where $w(x)$ is a nonnegative weight function on the interval $[-1, 1]$. In particular, if we set $w(x) = 1/\sqrt{1-x^2}$, $-1 \leq x \leq 1$, we get a family of orthogonal polynomials called Chebyshev polynomials, which are extremely important in approximation theory [2]:

$$T_n(x) = \cos(n \arccos x), \quad n \geq 0.$$

There is a very large literature on these polynomials, including a variety of formulas for them [15].

All the zeros of $T_n(x)$ belong to the interval $[-1, 1]$ and are given by

$$x_j = \cos\left(\frac{2j\pi + \pi}{2n}\right), \quad j = 0, 1, \dots, n-1.$$

Since the $n+1$ polynomials $T_0(x), T_1(x), \dots, T_n(x)$ are mutually orthogonal with respect to the inner product (11), the set $\mathcal{T} = \{T_0(x), T_1(x), \dots, T_n(x)\}$ forms a basis of the inner product space V_n .

Now for a given fixed $\xi \in \mathbb{R}$, let

$$K_n(x; \xi) := \sum_{k=0}^n T_k(x)T_k(\xi).$$

The polynomial $K_n(x; \xi)$ is called the kernel polynomial with respect to $\langle \cdot, \cdot \rangle$ and the parameter ξ . Kernel polynomials are known to satisfy the reproducing property; for $p \in V_n$

$$\langle K_n(\cdot; \xi), p \rangle = \int_{-1}^1 K_n(x; \xi)p(x)w(x)dx = p(\xi).$$

The n th kernel polynomial $K_n(x; \xi)$ is the unique solution of the following constrained approximation problem:

$$\left\| \frac{K_n(\cdot; \xi)}{K_n(\xi; \xi)} \right\| = \min\{\|p\| \in V_n, p(\xi) = 1\}. \quad (12)$$

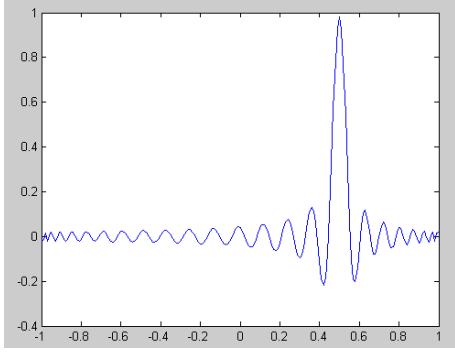


Figure 1: A scaling function localized at $x = 0$.

Equation (12) indicates that the kernel polynomials $K_n(x; \xi)$ are localized around $x = \xi$. $K_n(x; \xi)$ takes its maximum value at $x = \xi$ (see Figure 1).

3.3.2 Scaling functions

For parameters $x_0 < x_1 < \dots < x_n$, we define *scaling functions* as kernel polynomials:

$$\varphi_{n,k}(x) = \varphi_n(x; x_k) := K_n(x; x_k), \quad k = 0, 1, \dots, n.$$

For a suitable set of parameters, we then get the following theorem.

THEOREM 3.4. *If we take the parameters $x_0 < x_1 < \dots < x_n$ as the $n + 1$ zeros of $T_{n+1}(x) = 0$, then*

$$\begin{aligned} \langle \varphi_{n,k}, \varphi_{n,l} \rangle &= \sum_{j=0}^n T_j(x_k) T_j(x_l) \\ &= \varphi_{n,k}(x_k) \cdot \delta_{k,l} \\ &= \varphi_n(x_k; x_k) \cdot \delta_{k,l}. \end{aligned}$$

From the above theorem, we can see that the scaling functions $\varphi_{n,0}, \varphi_{n,2}, \dots, \varphi_{n,n}$ form another basis of the inner product space V_n . Therefore $T_0 \in V_n$ can be written uniquely as the linear combination of the polynomials $\varphi_{n,0}, \varphi_{n,2}, \dots, \varphi_{n,n}$:

$$T_0 \equiv 1 = \frac{1}{n+1} (\varphi_{n,0} + \varphi_{n,1} + \dots + \varphi_{n,n}).$$

3.4 Decomposition of a signal vector

Now we show how to decompose a given signal vector \mathbf{x} . Let $\tilde{\Delta} := \Delta - I$. It follows that $\sigma(\tilde{\Delta}) \subset [-1, 1]$, which is the domain of the scaling functions. For each \mathbf{x}_j , let

$$\mathbf{x}_j = \frac{1}{n+1} \varphi_{n,j}(\tilde{\Delta}) \mathbf{x}.$$

Then we have

$$\begin{aligned} \mathbf{x} &= \mathbf{x}_1 + \dots + \mathbf{x}_n \\ &= \frac{1}{n+1} \varphi_{n,1}(\tilde{\Delta}) \mathbf{x} + \dots + \frac{1}{n+1} \varphi_{n,n}(\tilde{\Delta}) \mathbf{x}. \end{aligned}$$

Clearly this decomposition satisfies Condition 1 in Section 2.2.1. But it depends on both the distribution of the eigenvalues of the Laplacian matrix $\tilde{\Delta}$ and on the spectral distribution of the signal vector \mathbf{x} with respect to the eigenvectors of $\tilde{\Delta}$. It is not certain whether the decomposition satisfies Conditions 2, 3, and 4. As illustrated in Figure 1, the scaling functions used in the decomposition are expected to pass only vectors with narrow frequency bands. Within the range $1 > \delta \geq 0.2$, we use $B_j(\delta)$ to denote the

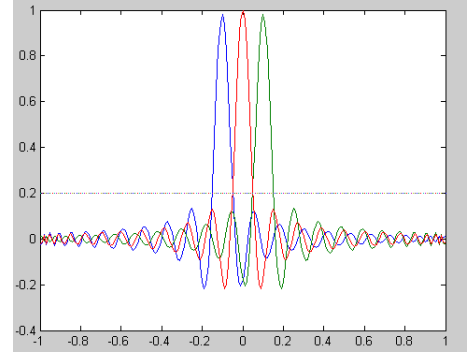


Figure 2: The effective passing band $B_j(\delta)$ with $\delta = 0.3$.

'effective' passing band $\{x \in [-1, 1] \mid |\varphi_{n,j}(x)| \geq \delta\}$ of $\varphi_{n,j}(\tilde{\Delta})$ (see Figure 2). Clearly, $B_j(\delta) \rightarrow \emptyset$ as $\delta \rightarrow 1$. For a fixed n , $\{B_j(\delta)\}_{j=1, \dots, n}$ acts almost like a partition of the interval $[-1, 1]$, i.e.,

$$\bigcup_{1 \leq j \leq n} B_j(\delta) = [-1, 1], \quad m(B_k(\delta) \cap B_l(\delta)) \leq \eta, \quad k \neq l, \quad (13)$$

where m is the Lebesgue measure and η is dependent on δ .

The condition expressed in Equation (13) can be seen as a relaxed version of Condition 3. Although the decomposition may not meet Condition 3, Equation (13) suggests that it will still serve our purpose.

It is inefficient to compute each matrix $\varphi_{n,j}(\tilde{\Delta})$ separately. The performance of $N \times N$ matrix-matrix multiplications is usually $O(N^3)$, and the results are prone to perturbation error. Fortunately, our calculations contain only matrix-vector multiplications, which have a time complexity which is only $O(N^2)$. We now suggest two methods to compute $\varphi_{n,j}(\tilde{\Delta}) \mathbf{x}$.

3.4.1 Method 1

Let $\xi_j, j = 0, 1, \dots, n$ be the $n + 1$ zeros of $T_{n+1}(x) = 0$. These zeros are given by

$$\xi_j = \cos\left(\frac{2j\pi + \pi}{2(n+1)}\right), \quad j = 0, 1, \dots, n.$$

It can easily be proved that the polynomials $\varphi_{n,j}(x) = \varphi_n(x; \xi_j)$, $j = 0, 1, \dots, n$ have n distinct zeros and that they are given by

$$\xi_0, \xi_1, \dots, \xi_{j-1}, \xi_{j+1}, \dots, \xi_n.$$

Since the degree of the polynomial $\varphi_{n,j}(x)$ is n , the polynomial $\varphi_{n,j}(x)$ can be factorized as follows:

$$\varphi_{n,j}(x) = A_j(x - \xi_0) \cdots (x - \xi_{j-1})(x - \xi_{j+1}) \cdots (x - \xi_n)$$

for some $A_j \in \mathbb{R}$. We now go on to compute $\mathbf{x}_j = \varphi_{n,j}(\tilde{\Delta}) \mathbf{x}$ according to the following procedure.

1. Input:
 $\mathbf{x}_j \leftarrow \mathbf{x}$
 2. From $j = 2$ to n except j
 $\mathbf{x}_j \leftarrow (\tilde{\Delta} - \xi_j I) \mathbf{x}_j$
 3. Output:
 $\Psi \leftarrow 1/(n+1) \Psi$

Since matrix-vector multiplication takes place n times in the procedure, its overall complexity is about $O(nN^2)$. As long as the degree n is small in comparison with N , we can get the result faster than by using matrix-matrix multiplication with a complexity of $O(N^3)$.

3.4.2 Method 2

This method uses the triple recursion relation for Chebyshev polynomials. From the definition, $T_0(x) = 1$, $T_1(x) = x$. The relation is given by

$$T_{n+2}(x) = 2xT_{n+1}(x) - T_n(x), n = 0, 1, \dots$$

Using the relation, we compute $\mathbf{x}_j = \varphi_{n,j}(\tilde{\Delta})\mathbf{x}$ according to the following procedure.

```

1. Input:
    $t_0 \leftarrow 0, t_1 \leftarrow \xi_j$ 
    $T_0 \leftarrow$  zero matrix,  $T_1 \leftarrow \tilde{\Delta}$ 
    $\Psi \leftarrow t_0 T_0 + t_1 T_1$ 

2. From  $j = 2$  to  $n$ 
    $t \leftarrow 2\xi t_1 - t_0, t_1 \leftarrow t, t_0 \leftarrow t_1$ 
    $T \leftarrow 2\tilde{\Delta} T_1 - T_0, T_1 \leftarrow T, T_0 \leftarrow T_1$ 
    $\Psi \leftarrow \Psi + t \cdot T$ 

3. Output:
    $\Psi \leftarrow 1/(n+1)\Psi$ 

```

As in Method 1, there are n matrix-vector multiplications in the procedure. However, while Method 1 is a convenient way to compute $\mathbf{x}_j = \varphi_{n,j}(\tilde{\Delta})\mathbf{x}$, a slight modification of Method 2 allows us to compute all of $\varphi_{n,j}(\tilde{\Delta})\mathbf{x}, j = 1, 2, \dots, n$ simultaneously, resulting in a procedure with a time complexity that is almost $O(nN^2)$. This efficient simultaneous computation of $\mathbf{x}_j, j = 1, 2, \dots, n$ is possible because the representations of terms of \mathbf{x}_j share common vectors $\{T_0(\tilde{\Delta})\mathbf{x}, T_1(\tilde{\Delta})\mathbf{x}, \dots, T_n(\tilde{\Delta})\mathbf{x}\}$.

4. EXPERIMENTAL RESULTS

We used scaling functions based on Chebyshev polynomials in these tests. Our scheme is tested on data that has been corrupted by random noise, affine transform and remeshing attacks. We analyzed the execution time and the robustness of our watermarking scheme.

4.1 Resource

We used a personal computer which has a Intel Pentium4 2.6GHz processor and 512 Mbytes of main memory for testing. The models we used were the Stanford Bunny [1], a Dragon [1] and a Skeleton Hand [1] where the number of vertices are 1494, 1257 and 1055 respectively.

Figures 3, 4, 5 show the Stanford Bunny, Dragon and Skeleton Hand model without any watermarks.

4.2 Method

We attacked each watermarked model using random noise, affine transformation and remeshing. We made the displacement vector for a random noise attack in two ways. In the first, each component of the displacement vector is a value between -1 to 1. In the second, the size of the displacement vector is a constant times the watermark vector. For the affine transform attack, we applied



Figure 3: Stanford bunny model.



Figure 4: Dragon model.



Figure 5: Original Skeleton hand model



Figure 6: Stanford Bunny attacked by random noise.



Figure 7: Dragon attacked by affine transformation.

scaling and rotation matrices and a translation vector to each vertex of the watermarked model. Finally, in the remeshing attack, we modified the topology of the watermarked model. this remeshing attack changes shape of watermarked model slightly but it doesn't change the number of edges.

Figures 6, 7, 8 show the Stanford Bunny, Dragon, and Skeleton Hand models after attack by random noise.

To improve the speed of watermarking, we implemented the following mesh partitioning method that uses the BFS (Breadth First Search) method.

1. Choose a feature vertex to become the center of the submesh.
2. Find the child vertices of the feature vertex that are connected by an edge, and add them to the submesh.
3. Each of the child vertices created in step2 now becomes a feature vertex. Steps 1 and 2 are then repeated.

Figures 9, 10, 11 show the Stanford Bunny, Dragon, and Skeleton Hand model after the watermark has been inserted into sub-meshes.

4.3 Result



Figure 8: Skeleton Hand attacked by random noise.

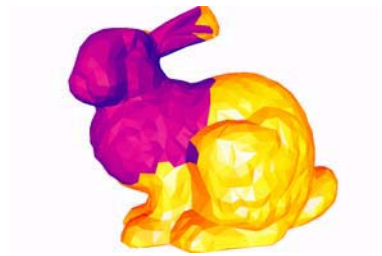


Figure 9: Watermarked Stanford Bunny.



Figure 10: Watermarked Dragon.



Figure 11: Watermarked Skeleton Hand.

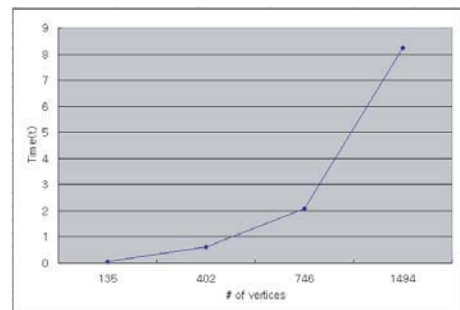


Figure 12: Variation of a execution time against number of vertices.

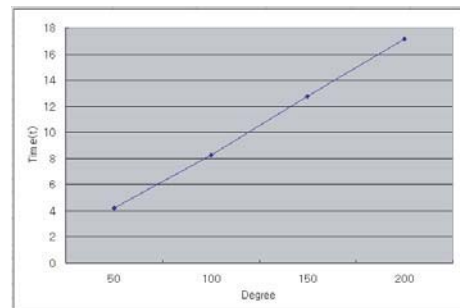


Figure 13: Execution time against scaling function degree.

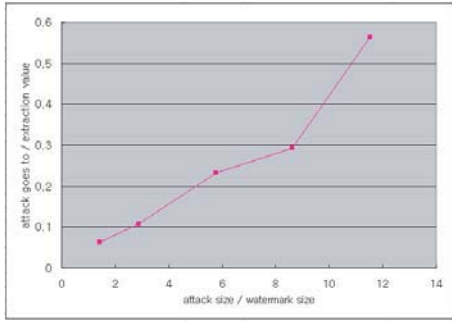


Figure 14: Robustness against intensity of attack.

4.3.1 Execution time

We define execution time as the total time for making a watermarked model from the original model. First, we calculated the time complexity against the number of vertices with a fixed scaling function degree. Fig 12 shows that the time complexity is $O(N^2)$ in this case. It means that we can watermark efficiently using mesh partitioning even if the model is large. We also calculated the time complexity against the scaling function degree. Fig 13 shows that the time complexity is $O(N)$ in this case. It means that if we use a higher-degree scaling function, we can embed more watermark bits, but then we will require a longer execution time.

4.3.2 Robustness

We analyzed the robustness of our watermarking algorithm. In this test, we only used a random noise attack, because we can affine transform and remeshing may be considered as special cases of random noise. We analyzed robustness against changes in the size of the random noise attack vector and of the watermark vector. The x -axis of Fig 14 represents the intensity of the attack, and the y -axis represents robustness. The results shows that robustness can be expected to be inverse proportion to the intensity of an attack.

5. CONCLUSIONS AND FUTURE WORK

We intended to reduce the computational costs of Ohbuchi et al's spectral watermarking scheme, in spite of possible degradation of robustness. The results show that the watermark embedded by our method is still robust against additive random noise attacks. We have chosen scaling functions to generate the linear operators for our scheme. When we use low-degree scaling functions for these linear operators the computational costs associated with processing large matrices are significantly reduced. In this case, the time complexity is about $O(nN^2)$, where N is the order of the Laplacian matrix and n is the degree of the scaling functions. As n grows, more watermark data can be embedded but the time complexity is increased.

A scaling function localized around $x = \xi$ inevitably gives rise to overshoots around $x = \xi$, which may contribute to the degradation of the performance of our watermarking scheme (see Figure 15). The Gibbs phenomenon is usually associated with the Fourier and other eigenfunction series. However much the degree of the scaling function is raised, the Gibbs phenomenon does not disappear. But the orthogonal polynomials in the scaling functions that we use have been studied extensively for applications in approximation of functions. For this reason, we are able to compute the linear operators generated by the scaling functions easily and fast.

In the future, we would like to investigate alternatives to scaling functions for the linear operators. Uniform polynomial approxima-

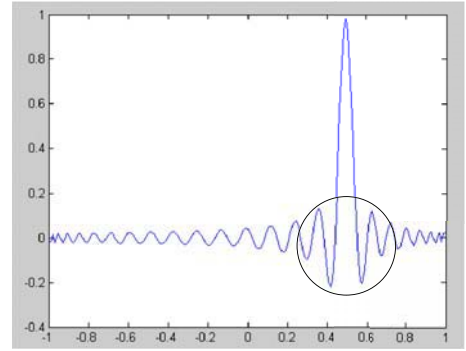


Figure 15: The Gibbs phenomenon.

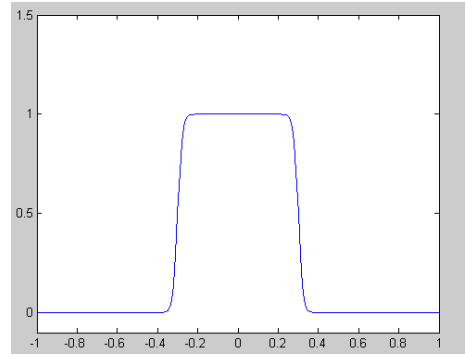


Figure 16: A hat function with local support

tions to hat functions with local supports (see Figure 16) would reduce the magnitude of the Gibbs phenomenon. Rational functions might also be able to approximate hat functions without incurring the Gibbs phenomenon.

6. ACKNOWLEDGMENTS

This research has been accomplished with the support of the Korea Science Foundation Objective Basis Research Contract R01-2001-00396.

The models are used courtesy of the Stanford University Computer Graphics Laboratory, the Georgia Tech Large Models Archive, and the Stereolithography Archive at Clemson University.

7. REFERENCES

- [1] Available at georgia institute of technology. In http://www.cc.gatech.edu/projects/large_models/.
- [2] K. Atkinson. *An Introduction to Numerical Analysis*. John Wiley, 2nd edition, 1989.
- [3] O. Benedens. Geometry-based watermarking of 3d models. *IEEE Computer Graphics and Applications*, 19(1):46–55, 1999.
- [4] F. R. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.
- [5] G. Golub and C. V. Loan. *Matrix Computations*. John Hopkins Univ. Press, 3rd edition, 1996.
- [6] C.-T. Hsu and J.-L. Wu. Hidden digital watermarking in images. *IEEE Trans. Image Process*, 8(1):58–68, 1999.
- [7] K. H. Ko, T. Maekawa, N. M. Patrikalakis, H. Masuda, and F.-E. Wolter. Shape intrinsic fingerprints for free-form object matching. In *Proceedings of the Eighth ACM Symposium on*

Solid Modeling and Applications, pages 196–207, Seattle, USA, 2003.

- [8] R. Ohbuchi, H. Masuada, and M. Aono. Data embedding algorithms for geometrical and non-geometrical targets in three-dimensional polygon models. *Computer Communications*, 21:1344–1354, 1998.
- [9] R. Ohbuchi and H. Masuda. Managing cad data as a multimedia data type using digital watermarking. In *IFIP WG5.2 Fourth Workshop on Knowledge Intensive CAD*, May 2000.
- [10] R. Ohbuchi, H. Masuda, and M. Aono. Watermarking three-dimensional polygon models. In *Proceedings of the ACM Multimedia '97*, pages 261–272, Seattle, Washington, USA, November 1997.
- [11] R. Ohbuchi, H. Masuda, and M. Aono. Watermarking three-dimensional polygon models through geometric and topological modifications. *IEEE Journal on Selected Areas in Communications*, pages 551–559, 1998.
- [12] R. Ohbuchi, S. Takahashi, T. Miyazawa, and A. Mukaiyama. Watermarking 3d polygonal meshes in the mesh spectral domain. In *Proc. Graphics Interface 2001*, pages 9–17, 2001.
- [13] I. Pitas. A method for watermark casting on digital images. *IEEE Trans. Circuit Syst. Video Technol.*, 8(6):775–780, 1998.
- [14] E. Praun, H. Hoppe, and A. Finkelstein. Robust mesh watermarking. *Computer Graphics (Proceedings of SIGGRAPH'99)*, 33:49–56, 1999.
- [15] G. Szegő. *Orthogonal polynomials*. AMS Colloquium Publications XXIII, American Mathematical Society, New York, USA, revised edition, 1959.
- [16] G. Taubin. A signal processing approach to fair surface design. *Computer Graphics (SIGGRAPH'95 Proceedings)*, pages 351–358, 1995.
- [17] M. Wagner. Robust watermarking of polygonal meshes. In R. Martin and W. Wang, editors, *Proceedings of the Conference on Geometric Modeling and Processing*, pages 201–208, 2000.