Robust Mesh Watermarking Based on Multiresolution Processing

[§]Kangkang Yin [§η]Zhigeng Pan [§]Jiaoying Shi [η]David Zhang
[§]State Key Lab of CAD&CG, Zhejiang University, Hangzhou, 310027, PRC
[η] Dept of Computing, Hong Kong Polytechnic University, Hung Hom Kowloon, HK

Abstract: In this paper we propose a new mesh watermarking scheme for triangle meshes, which are widely used in computer graphics systems. Arbitrary meshes lack a natural parameterization for frequency-based signal processing, and thus they are harder to be watermarked. In this paper, we adopt Guskov's multiresolution signal processing method for meshes and use his 3D non-uniform relaxation operator to construct a Burt-Adelson pyramid for the mesh, and then watermark information is embedded into a suitable coarser mesh. Since all the computation and data structures are the same as those used in the multiresolution signal processing, our mesh watermarking algorithm can be integrated naturally with the multiresolution mesh processing toolbox. To detect watermarks, registration and resampling are needed to bring the attacked mesh model back into its original location, orientation, scale, topology and resolution level. Experimental results and attack analysis are given to show that our watermarking algorithm is robust under a variety of attacks, including vertex reordering, noise addition, simplification, filtering and enhancement, cropping, etc. Our main contribution to the field of 3D watermarking is that we propose a generic applicable multiresolution framework in which other proposed techniques may fit in as well.

Keywords: Digital watermarking, Mesh watermarking, Multiresolution analysis, Mesh pyramid, Attack analysis

1. Introduction

Digital media data has become very common on the Internet in recent years. Its prevalence creates an urgent need for appropriate copyright protection schemes in digitized media. Conventional cryptographic systems permit only valid key-holders access to encrypted data, but once such data is decrypted there will be no way to track its reproduction or retransmission. Digital watermarking provides a novel way to achieve effective copyright protection [1,2,3]. Providing digital information security is attracting more and more attention nowadays.

In this paper we will describe a technique for watermarking graphics models, which may be used in Internet-based model shopping systems and collaborative design systems.

1.1 Background

Digital watermarking embeds unobtrusive labels in digital content. The label, also called a digital watermark, is usually imperceptible, but can be extracted or detected by some computing processes. The digital content can be in various data types, including audio, video, still image, text, graphical data, etc. There are many methods in which watermark signals are inserted into a document, including spatial domain approaches, frequency domain approaches, compressed domain approaches, statistical approaches, and approaches utilizing human sensory characteristics. The application domains are classified as ownership assertion, traitor tracing, authentication and integrity verification, labeling and annotation, usage control, and content protection [3]. The robustness requirement of watermarks can vary greatly due to different targetted applications. In ownership assertion or traitor tracing, the more robust the watermark is, the better, because we would possibly like to use them as a proof

for resolving copywrite disputes. In contrast, we do not care too much about robustness if the watermarks are used as labels and annotations which just carry some information to the end users.

In this paper, we aim at embedding robust watermarks into 3D meshes, which means the watermark must be difficult (hopefully impossible) to remove or destroy while remaining imperceptible. Acts of removal can be either unintentional, like common signal processing operations, or intentional, like deliberate and malicious attacks.

1.2 Mesh Watermarking and Previous Research Work

Watermarking provides a mechanism for copyright protection of digital media by embedding information identifying the owner in the data. The bulk of the research on digital watermarking has focused on media such as image, video, audio and text watermarking [4,5,6]. Research about mesh watermarking began in 1998. Ohbuchi *et al.* proposed several watermarking algorithms for polygonal models in [7], including TSQ (Triangle Similarity Quadruple), TVR (Tetrahedral Volume Ratio) and a visible mesh watermarking algorithm. These algorithms provide many insights into mesh watermarking, but the techniques are not yet robust enough. Benedens *et al.* subtly altered surface normals to embed watermark bits, and the watermark can survive simplification attacks [8]. Yeung *et al.* proposed a fragile watermarking algorithm in [9]. The most successful robust mesh watermarking algorithm was proposed by Praun *et al.* in [10]. They generalized spread spectrum techniques to surfaces. Firstly they construct a multiresolution set of scalar basis functions over the mesh, and then perturb the coordinates of the mesh vertices, using the basis functions as weights. Experiments show that this algorithm has high robustness, but it still leaves some space for improvements. For example, their scheme is not essentially a multiresolution one, at least the resolution concept in their context can not be shifted directly into multiresolution mesh processing; the computational cost is high and independent of mesh processing and editing algorithms.

We address the two challenges of mesh watermarking mentioned in [10] in natural and efficient ways:

(i) Arbitrary meshes lack a natural parameterization for frequency-based decomposition, and are thus harder to watermark. We adopt Guskov's relaxation operator [11] to construct Burt-Adelson (B-A) mesh pyramids, which are multiresolution representations for meshes. Then we embed the watermark into a coarser level of the mesh pyramid, which corresponds to the low-frequency components of the original finer mesh.

(ii) Various attacks may modify the geometry and topology of a watermarked mesh. Registration and resampling are needed to bring the attacked mesh model back into its original location, orientation, scale, topology and resolution level. Our registration and resampling methods are quite simple yet effective. More complex methods can be used to enhance the efficiency and accuracy.

The watermarking method proposed in this paper has two major advantages: (a) All the computation and data structures are the same with those used in the multiresolution signal processing methods proposed by Guskov, so our mesh watermarking algorithm can be integrated with his mesh processing toolbox naturally. (b) Watermarks are inclined to survive when mesh models are processed by various multiresolution mesh processing operations, since multiresolution signal processing of meshes mainly operates on the high frequency components while our watermarks are embedded into the low frequency components. However, this watermarking method is not a blind

one, which means that the watermark detection requires access to the original mesh data and the original embedded watermark.

In Section 2, the mesh watermarking method is described in detail. Section 3 and 4 present implementation techniques, experimental results and attack analysis. In Section 5, we draw a conclusion and list some possible future work.

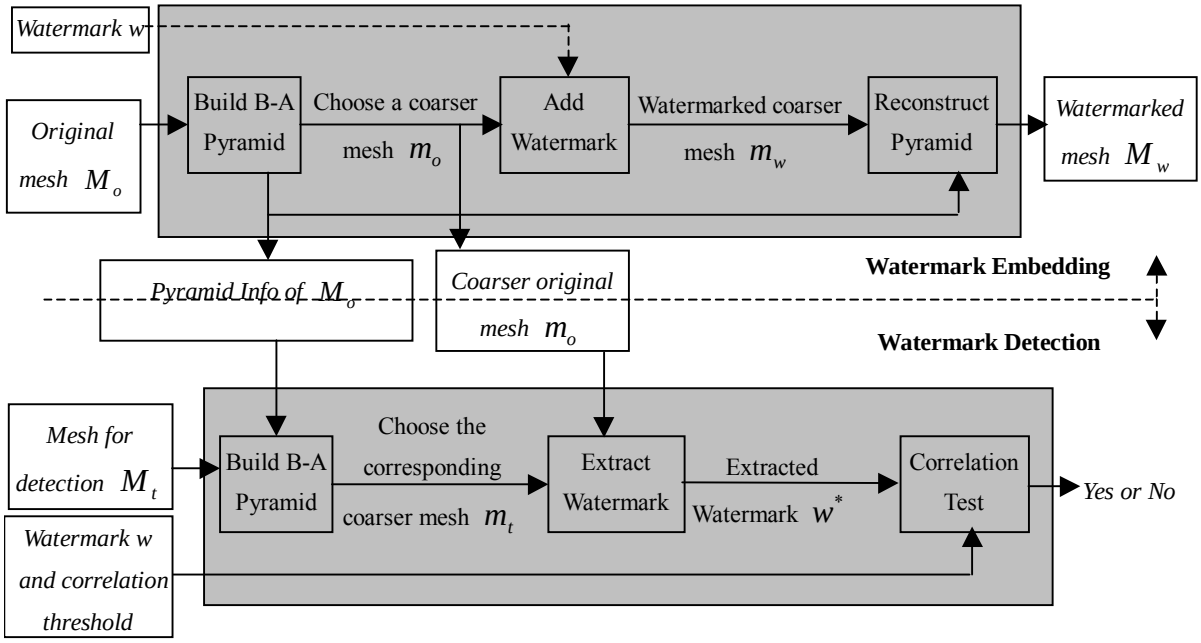


Fig.1 Watermarking Procedures

2. Description of the Mesh Watermarking Algorithm

The basic procedures of our watermarking algorithm are shown in Figure 1. The upper part of Figure 1 stands for the watermark embedding process. The lower part of Figure 1 stands for the watermark detection process. Building a B-A pyramid for a mesh is a common component of both watermark embedding and detection. In addition, the registration and resampling algorithms are used to prepocess an attacked mesh before its watermark detection. These techniques are disscussed in detail in the following sub-sections.

2.1 Burt-Adelson Pyramid Construction

The B-A pyramid is a kind of Laplacian Pyramid scheme proposed by Burt and Adelson for compact image coding [12], and I. Guskov generalized it to a mesh pyramid using his 3D non-uniform relaxation operator [11]. We briefly introduce the construction of B-A pyramid for meshes in the following; more detailed descriptions can be found in [11,12] or in the Appendix of this paper.

Pyramid algorithms belong to multiresolution analysis. A useful multiresolution approach for meshes is Hoppe's Progressive Mesh (PM) [13]. PM approach provides a decimation step by an edge collapse, an upsampling step by a vertex split. Let $M = (P, K)$ represents a triangular mesh, where

$P = \{p_i \in R^3 \mid 1 \le i \le N\}$ is a set of vertices, and $p_i = (x_i, y_i, z_i)$ is the coordinate of a vertex; *K* contains all the topological information, which indicates the adjacency relations. With PM approach we can obtain a sequence of meshes $M^n = (P^n, K^n),\ 1 \le n \le N$, where $P^n = \{p_i \mid 1 \le i \le n\}$, together with a sequence of detail records that indicate how to incrementally refine M^0 exactly back into the original mesh M^N. The vertex that is removed from mesh P^n to get P^{n-1} is numbered *n*. Different methods can be used to determine the sequence of edge collapses and the vertex position after an edge collapse. Garland's quadric error metric [14] is an effective priority criterion for edge collapses. Half-edge collapse, which simply removes one vertex of an edge while leaving the other vertex position unchanged, is a simple and efficient way for edge collapses.

In the PM setting, we may only change the coordinate of the vertex which is left after an edge collapse during a decimation step; while in the B-A pyramid method, the neighborhood vertex coordinates with subdivision connectivity to the vertex involved in the edge collapse can also be changed. The specific procedures for constructing a B-A pyramid are shown in Figure 2. To construct a sequence of meshes $M^n = (S^n, K^n),\ 1 \le n \le N$, we start from the original mesh S^N. There are four stages in obtaining $S^{(n-1)}$ from $S^{(n)}$: presmoothing, downsampling, subdivision and computation of details.

In Figure 2, $s^{(n)}$ denotes the vertex coordinates in $S^{(n)}$, the new coordinates of the remaining vertices after subdivision are denoted as $q^{(n)}$, and the differences between two mesh levels, i.e., the details, are denoted as $d^{(n)}$. In presmoothing and subdivision procedures, a relaxation operator proposed by I. Guskov is used [11]. Half-edge collapse is adopted in the downsampling steps. More detailed description can be found in the Appendix.

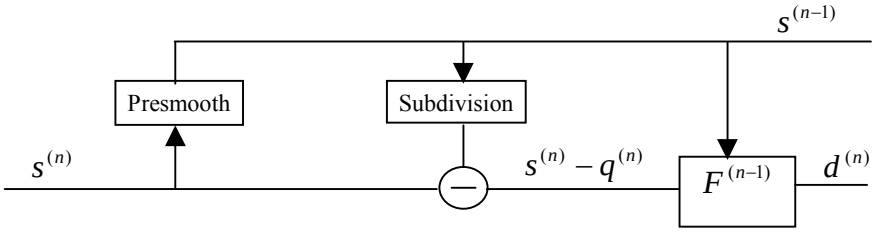


Fig.2 Burt-Adelson style pyramid scheme

Starting from the upper level or coarser level in the B-A mesh pyramid, we can reconstruct the lower level or finer level meshes: use $s^{(n-1)}$, subdivide values $q^{(n)}$, add in the details $d^{(n)}$, and then get the recovered original values $s^{(n)}$.

In pyramid construction, we need to record some useful information which is to be used in pyramid reconstruction, multiresolution signal processing or watermark extraction. The information recorded includes: downsampling vertex sequence {*n*}, weight value sequence $\{w^{(n)}\}$ of relaxation operators (see Appendix), and detail sequence $\{d^{(n)}\}$.

From the above procedures of pyramid construction, we can see that the upper coarser mesh can be viewed as the low frequency components of the lower finer mesh. From the viewpoint of signal processing, the vertices in the coarser mesh are the downsamples of the vertices in the finer mesh, and thus correspond to the low frequency components which retain the most important features of the original mesh while discarding the detail information. Compared with the image watermarking algorithms that embed watermarks into the low frequency components of

static images [5], we can embed watermarks into the coarser mesh. This is the primary difference between our algorithm and the algorithm presented by Praun *et al.* in [10], who found those half-edge collapses which cause the largest geometric changes during the PM construction, and the vertices in those half-edge collapses correspond to low frequency components. In our scheme, we find the low frequency components through pyramid construction. It resembles a low pass filter for images, and conceptually it is easier to understand.

2.2 Watermark Embedding

In [10], Praun *et al.* adopted 50 real numbers as a watermark vector, sampled from a Gaussian distribution with zero mean and variance 1, i.e., *N(0,1)*. Instead, we use $w_k \in \{-1,1\}, k = 1,...,m$ as our watermark signal. It is a binary sequence, so it can be mapped directly to a {0,1} bit stream. This enables the embedding of watermarks with real meaning, such as copyright text or a small logo image.

The watermark is inserted as follows (refer to Figure 1):

(1) Construct a B-A pyramid for the original mesh M_o, select a suitable level of coarser mesh m_o as the object of watermarking.

(2) Choose $\lceil m/3 \rceil$ vertices p_i {i=1…$\lceil m/3 \rceil$} from m_o. Calculate the minimal length of p_i's 1-ring edge neighborhood by $lm_i = \min\{length(e) \mid e \in \varepsilon_1(i)\}$ (see Appendix for neighborhood definitions). Then we can represent the insertion process as three equations:

$$p'_{ix} = p_{ix} + w_k \cdot \alpha \cdot lm_i, \quad (1)$$

$$p'_{iy} = p_{iy} + w_{k+1} \cdot \alpha \cdot lm_i, \quad (2)$$

$$p'_{iz} = p_{iz} + w_{k+2} \cdot \alpha \cdot lm_i, \quad (3)$$

where p_{ix} is the *x* coordinate of vertex p_i, p'_{ix} is the watermarked *x* coordinate, and other symbols have similar meanings. α is the strength coefficient, which controls the energy of the embedded watermark signal. lm_i is the parameter controlling the local watermarking strength; it ensures that the watermark strength adapts to the local feature of the mesh. The watermarked coarser mesh is denoted as m_w.

In practice, we give a threshold *T,* and only embed watermark into vertices whose $lm_i > T$. It can eliminate weak watermark information and ensure the robustness. The higher *T* is, the shorter the embedded watermark is. The vertices p_i can be chosen either randomly or key-dependently, but they should approximately uniformly scatter on the whole model so the robustness to crop attacks can be ensured. In our experiments given in Section 4, we choose p_i randomly for simplicity. To enforce uniform distribution, once we have selected a vertex for watermarking, we do not allow its 1-ring vertex neighbourhood to be watermarked anymore.

(3) Reconstruct the watermarked finer mesh M_w from m_w, using pyramid reconstruction algorithm.

2.3 Watermark Detection

Our watermark detection algorithm has two steps. The first is watermark extraction, in which we need to access the original mesh data. Given a mesh M_t, the extraction algorithm extracts the watermark signal possibly embedded in the mesh. The second is the correlation test, in which we need to access the original embedded

watermark. Then we can decide whether mesh M_t contains this designated watermark by comparing it with the extracted watermark. Typically, the decision is made by the people who hold the original mesh data, i.e., who watermarked the mesh.

Based on the above watermark insertion algorithm, our watermark detection algorithm is as follows (refer to Figure 1). We construct the B-A pyramid for M_t, using the pyramid information of the original mesh M_o. Then we take the coarser mesh m_t that is of the same resolution as m_o, and extract the watermark:

$$w_k^* = sign(p_{ix}^{'} - p_{ix}), \quad (4)$$

$$w_{k+1}^* = sign(p_{iy}^{'} - p_{iy}), \quad (5)$$

$$w_{k+2}^* = sign(p_{iz}^{'} - p_{iz}), \quad (6)$$

where $p_i^{'}$ is vertex in m_t, p_i is vertex in m_o, and *sign* is a function that returns the sign of its parameter.

If the watermarked mesh undergoes some processing or attacks, the extracted watermark may not be exactly the same with the embedded watermark. So we need to define a criterion by which we can decide whether the designated watermark is present in the mesh. We employ linear correlation [15,10] between the extracted watermark and the designated watermark as our decision criterion:

$$corr(w^*, w) = \frac{\sum_{i=1}^{m}(w_i^* - \overline{w^*})(w_i - \overline{w})}{\sqrt{\sum_{i=1}^{m}(w_i^* - \overline{w^*})^2}\sqrt{\sum_{i=1}^{m}(w_i - \overline{w})^2}}, \quad (7)$$

where w^* is the extracted watermark and w is the designated watermark, $\overline{w}$ is the mean of vector w, $\overline{w^*}$ is the mean of vector w^*, m is the vector length. The value of *corr* lies in [-1,1]. If *corr* equals 1, that means the two vectores are completely positively correlated, which in our case of binary sequences means the two watermarks are the same. A value of *corr* near zero indicates that the vectors are uncorrelated. So if the computed correlation exceeds a deliberately chosen threshold, we conclude that the designated watermark w presents in the mesh. Since the extracted watermark from an unwatermarked or differently watermarked mesh is highly impossible to correlated with the designated watermark and can be viewed as a random sequence, the computed correlation tends to be low (see Section 4.1).

2.4 Registration

Similarity transforms, including translation, rotation and uniform scale, are often applied to a 3D model when it is employed within a graphical scene. Often we transform an object during rendering while leaving the object geometry unaltered. Yet an attacker might change the vertex coordinates directly, which puts forward the necessity to bring the object back to its original location, orientation and scale — a process called object registration — before watermark extraction.

Our registration algorithm is quite simple. 7 degrees of freedom are allowed: translation, rotation and uniform scaling of the mesh. The registration is always performed between the attacked mesh and the original unwatermarked mesh, for registration with the watermarked mesh will falsely introduce the watermark

information and increase the false positive possibility. We denote the mesh to be registered as M_t, the original mesh as M_o. The registration procedure is as follows:

Step1. Provide an initial transformation between the two meshes. This is done by users from the Graphical User Interface (GUI) provided by the underlying system. The approximately registered M_t is denoted as M_r.

Step2. Uniformly choose a subset of vertices $v_{ri}, (i = 1...k)$ from M_r.

Step3. Define an energy function: $E = \sum_{i=1}^{k} E_{dist}(v_{ri}, v_{oi})$, where v_{oi} is the nearest vertex of v_{ri} in M_o. E_{dist} measures the distances between the meshes, it is the sum of the squared distances between the corresponding vertex pairs. We then use the Powell method [15] to minimize the energy function E. After this minimization process, M_r is used as the registered version of mesh M_t.

This registration algorithm requires a reasonable initial condition and is somewhat low in efficiency. More complex registration algorithms can be employed to enhance the efficiency, such as the algorithm proposed by Chen *et a*l. in [16] .

2.5 Resampling

When a watermarked mesh undergoes some attacks like simplification, vertex reordering or cropping, the topology of the mesh will be changed and we cannot extract the watermark directly using our watermark detection algorithm. Therefore we need first to resample the attacked mesh to obtain a mesh with the topology of the original mesh but retain the geometry of the attacked mesh. We denote the mesh to be resampled as M_s, the original mesh as M_o. The resampling procedure is as follows:

Step1. For every vertex in M_s, find its nearest vertex in M_o. If their Euclidean distance is less than a given threshold, mark them as a matched vertex pair.

Step2. Project every unmatched vertex in M_o onto its nearest plane in M_s, mark the vertex and its projection as a vertex pair.

Step3. Connect the matched vertices and the projected vertices in M_s, using the adjacency information in M_o. The resulting mesh is denoted as M_r.

Step4. Define an energy function: $E = \sum_{i=1}^{n} (E_{dist}(v_{ri}, v_{oi}) + c_d \cdot E_{deform}(v_{ri}, v_{oi}))$, where v_{ri} is a vertex in M_r, v_{oi} is its corresponding vertex in M_o. E_{dist} measures the distances between the meshes, it is the sum of the squared distances between the corresponding vertex pairs. E_{deform} measures the deformation of the resampled mesh. Each edge in the mesh is viewed as a spring, with rest length equal to its corresponding edge length in M_o. Then the deformation E_{deform} can be measured by the potential energy of the springs. c_d is a weight factor, which is equal to 10^{-3} in our implementation. We then use a conjugate gradient method [15] to minimize the energy function E, which represents the energy difference between M_r and M_o. After minimization process, M_r is used as the resampled version of mesh M_s.

It should be noted that in the matched vertex pairs found out in Step 1, when the distance of the vertex pair is less than a given threshold, we neglect this vertex pair in our energy-minimization process. The reason is that these vertices contain the watermark information.

More complex resampling algorithms can be employed to enhance the accuracy, such as the algorithm used by Praun *et al.* [10,17].

3. Implementation of A Mesh Watermarking System

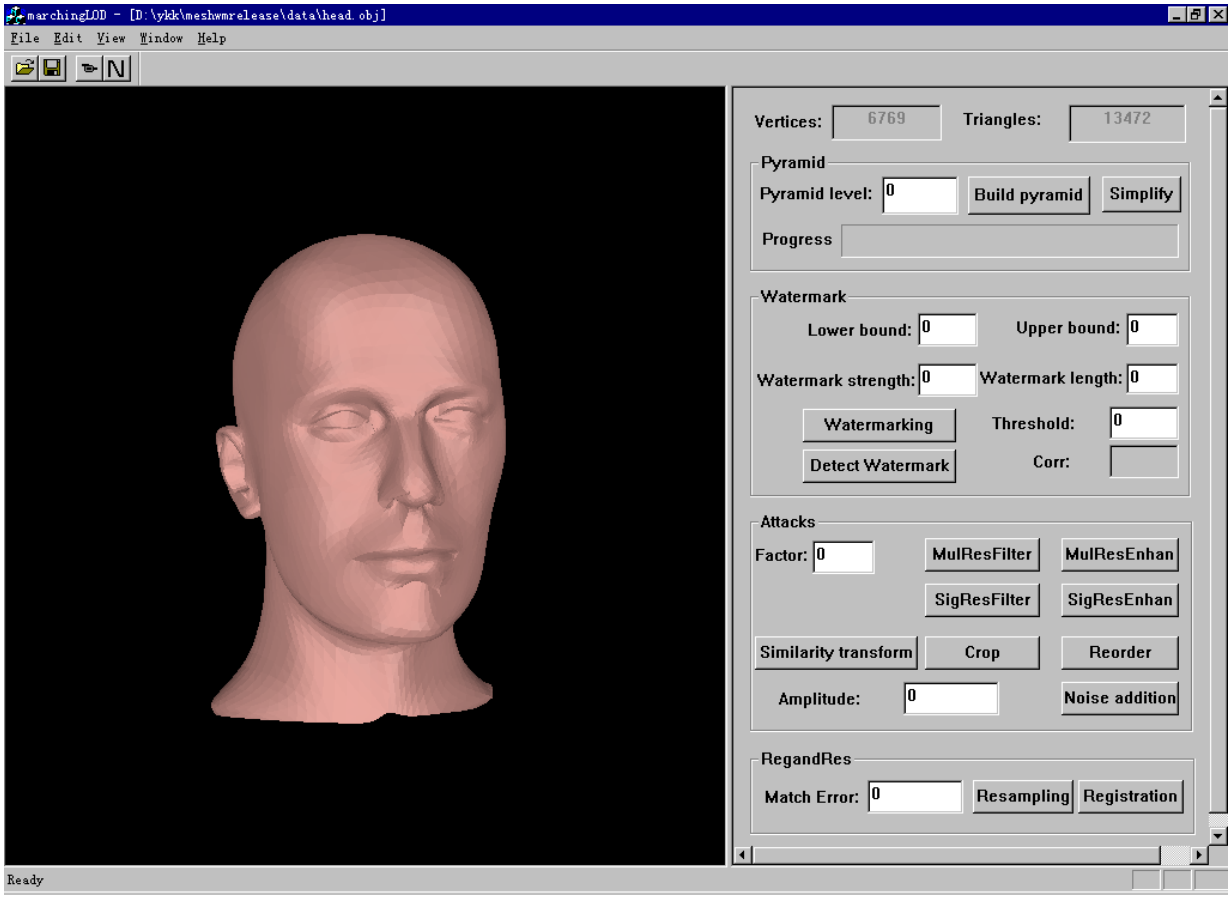
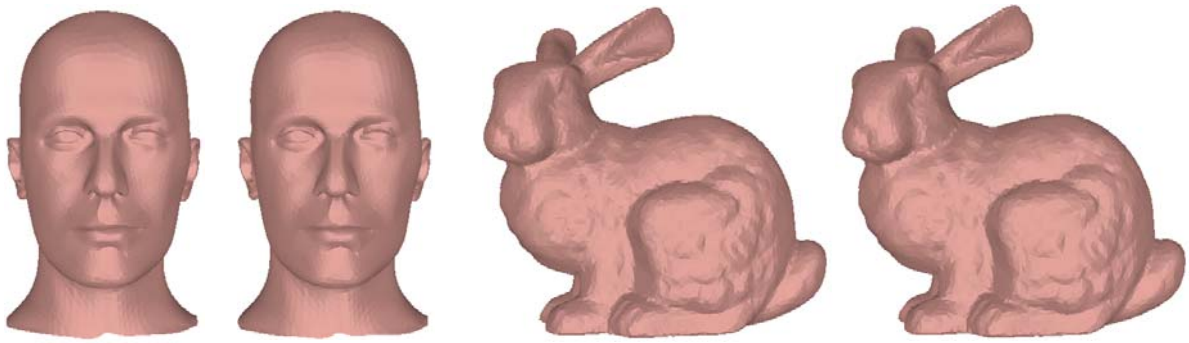


Fig.3 Interface of the watermarking system

Based on the watermarking algorithm described above, we implemented an experimental mesh watermarking system in MSVC++ 6.0 on a PII 350 machine. The interface is shown in Figure 3. The left sub window is for mesh rendering. Users can view and transform the model interactively here. The right sub window is for commands and parameter settings. It contains four group boxes: "Pyramid" is for B-A pyramid building and half-edge simplification; "Watermark" is for watermark embedding and detection; "Attacks" is for applying various mesh attacks; "RegandRes" is for performing necessary registration or resampling operations. Generally speaking, buttons correspond to commands or operations and edit boxes are in charge of parameter settings.

The watermark length, strength and threshold T (threshold of lm_i in Equations (1) to (3)) can be controlled by users. T depends on both the original size of the mesh model and the chosen mesh pyramid level (adjustable by "lower bound" and "upper bound" edit boxes) into which we plan to insert the watermark. The larger T is, the smaller the number of watermark bits that can be embedded into a certain resolution level of a given model. If T is so large that the designated watermark length can not be granted, then we only embed as many bits as possible and return the length of the watermark actually embedded. The watermark strength is controlled by strength coefficient α in Equations (1) to (3). When α exceeds some threshold, the watermark will become perceivable.

In our experiments, a watermark of 250 bits is embedded into a Head model and a watermark of 300 bits is embedded into a Bunny model. These watermark lengths are longer than Praun's watermark length of 50 coefficients [10]. The resulted watermarked models are denoted as Headwm and Bunnywm, shown in Figure 4. We set α=0.08 and T=0.03 in both cases. Similar parameter settings are also used in the following attack analysis. Human visions cannot distinguish the original and the watermarked models, which implies that this watermarking algorithm is transparent. Later attack analysis will show it is robust too. Similar results have been obtained when the watermarking and various attacks are applied to other models.



Head Headwm Bunny Bunnywm

(6769 vertices and 13472 faces) (8704 vertices and 17332 faces)

Fig.4 Original Model and Watermarked Model

4. Experimental Results and Attack Analysis

4.1 Threshold Setting

To perform watermark detection, we should first choose a suitable correlation threshold for correlation test. Then we can assert a watermark's presence by comparing this threshold and the correlation value computed from Equation (7). Since our watermark can be any binary sequence and possesses no distinctive probability distribution, there is no universal way to compute a suitable threshold mathematically in the case of the null hypothesis. Therefore, just as what is usually done in the watermarking literature [5], we adopt an experimental method to choose a threshold. The nature of such experiment is the Monte-Carlo method often used in the probability theory.

In doing this, 10,000 randomly generated watermark vectors of length $k \cdot 50(k = 1,...,9)$ are correlated with the designated watermarks of the same length, and the experiment is run ten times with different sets of designated watermarks. We plot the mean of the mean correlation values, the maximum of the maximal correlation values and the correlation values of the designated watermarks with themselves in Figure 5. This figure can guide us to choose a suitable threshold for a given watermark length. For example, when the watermark length is 250 bits, the maximal correlation between a randomly generated watermark and the designated watermark is very probably below 0.3. So we can set the correlation threshold to 0.3. In order to decrease the possibility of false positive, we can increase the threshold further, but the possibility of false negative will rise as a

result. Here false positive means asserting incorrectly that a document is watermarked when it is not, false negative means failing to detect a watermarked document. Usually a compromise favoring a lower false positive rate needs to be achieved.

After some mesh attacks, the embedded watermark will be partly damaged, so its correlation with the original watermark will be less than 1. However, if our watermark algorithm is roboust enough and can resist attacks to some extent, the extracted watermark should be more "close" to the original watermark than a random watermark, i.e., the correlation is larger than the threshold we choose from the random watermark correlation test. So when the detected watermark has a stronger correlation with the embedded watermark, it is generally sound to conclude that the mesh contains the designated watermark. Although, of course, false positives and negatives do exist due to the probabilistic nature of the watermark detection method. In Section 4.2, we will see that the detector results degrade gracefully after various attacks, which shows this watermark algorithm is robust.

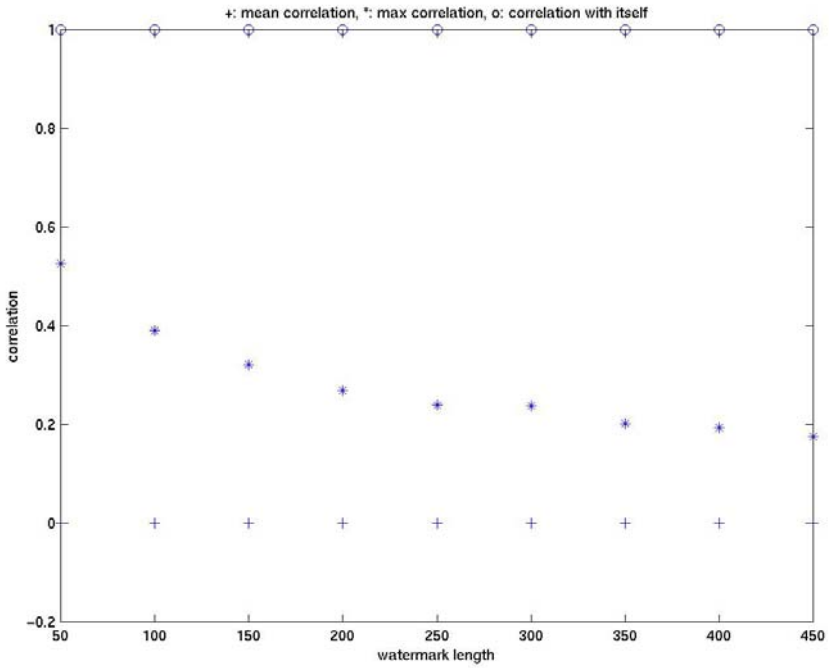


Fig.5 Correlation Threshold Setting

4.2 Attack Analysis

The watermark detection results for a variety of attacked Head models are listed in Table 1, and some of the attacked meshes are shown in Figure 6. Since the watermarking process and some of the attacks depend on random numbers or processes, there is variability in the results. We therefore perform each test five times on Head models with different watermarks embedded in different locations, and report the mean value.

Table 1 Attack Analysis of Mesh Watermarking

ATTACK		REG	RES	ATTACK PARAMETERS AND DESCRIPTIONS		CORR
Vertex Reorder			√	Shuffle vertex number randomly		0.7856
Noise Addition				Uniform noise amplitude is 0.45% of the largest dimension of the object		0.5184
Multiple Water-marking	Type A			Embed a first watermark of 250 bits, a second watermark of 350 bits	Detect the first watermark	0.7504
					Detect the second watermark	0.7844
	Type			Embed a first watermark of 250	Detect the first watermark	0.5600

	B			bits, a second watermark of 250 bits	Detect the second watermark	0.7440
Simplifi-cation	Type A		√	Half-edge collapse simplification, Remove 7/8 vertices, 847 vertices left		0.6656
	Type B		√	Merge near-coplanar faces simplification[18], M^3 level, 1555 vertices left on average		0.5600
Multiresolution Filtering				Set the details of level 1269<n<=6769 in the mesh pyramid to zero		0.6464
Multiresolution Enhancement				Multiply the details of level 1269<n<=6769 in the mesh pyramid by two		0.6258
Single Resolution Filtering				15 relaxation steps		0.6752
Single Resolution Enhancement				Multiply all details by two after 15 relaxation steps		0.7264
Cropping			√	1077 vertices cropped on average		0.6512
Similarity Transform		√		Users input transformations randomly from the GUI		0.7792
Combined Attacks		√	√	Second watermarking, vertex reorder, simplification, noise addition, single resolution filtering, cropping and similarity transform		0.4256

If registration (noted as REG in Table 1) or resampling (noted as RES in Table 1) needs to be performed between the attacked mesh and the original mesh before the watermark detection, then we put a √ in the corresponding row. In the CORR column, the correlation values computed by Equation (7) are listed.

To demonstrate the resilience of the watermark under noise addition, we add a noise vector to each vertex. The amplitude of the noise vector is 0.45% of the largest dimension of the object. Although the resulting mesh model suffers from severe quality degradation, the correlation is still high above the threshold.

For multiple watermarking, we add a second watermark to the watermarked model. In type A, we embed a second watermark of 350 bits with α=0.04 into a different resolution coarser mesh. In type B, we embed a second watermark of 250 bits with α=0.08 into the same resolution coarser mesh. Both can pass watermark detection.

We adopt two different simplification algorithms to test the watermark robustness to simplification attacks. The type A simplification is simply half-edge collapses using Garland's quadric error metric [14] as the priority criterion. 7/8 vertices are removed. The type B simplification is based on merging near-coplanar faces [18], which encompasses merging faces and retriangulations. The third level of detail M^3 of the watermarked mesh is used for watermark detection. Both simplification methods cannot destroy the watermark embedded in the original meshes.

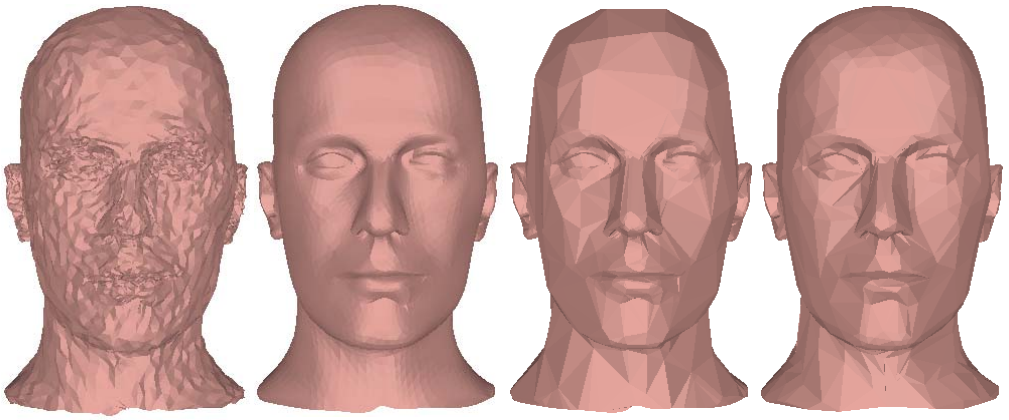
Multiresolution filtering and enhancement are typical operations in Guskov's multiresolution processing toolbox for meshes [11]. We operate on the detail coefficients in the range of $1269<n<=6769$, i.e., the first 5500 detail coefficients in the B-A pyramid. When multiplying these detail coefficients by a value k, $0<=k<1$, multiresolution filtering is achieved. If $k>1$, multiresolution enhancement is achieved. In our experiments, we set detail coefficients to zero when filtering, multiply them by two when enhancing. Single resolution filtering is achieved by using Guskov's relaxation operator on all vertices simultaneously. In our tests we run the relaxation operator 15 times. By multiplying the differences between the original mesh and the single resolution relaxed mesh by two, we get the single resolution enhanced mesh. Our watermarks can survive all these processes.

Five crop attacks are applied, in which we discard separately all vertices in the top, bottom, front, back and left fourth of the object's bounding box. On average, 1077 vertices are deleted. The watermark information in the cropped meshes is still strong enough for detection. Around half of the vertices can be removed before the correlation value drops down to the threshold.

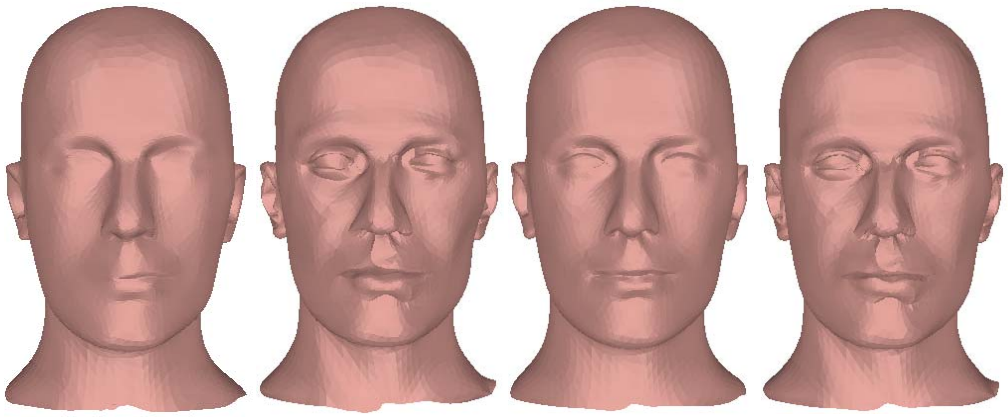
Similarity transformations are performed from the user interface of the experimental watermarking system. One of them is shown in Figure 6.

Finally, combined attacks in the last row of Table 1 include: second watermarking, vertex reordering, simplification, noise addition, single resolution filtering, cropping and similarity transform. In the five tests, the amplitude of noise is 0.2% of the largest dimension of the mesh. Single resolution filtering runs the relaxation step once. On average, 2630 vertices are left after simplification and cropping. Both types of multiple watermarking are tried. Vertex reorder and similarity transform are done randomly, either by the watermarking system or by the end users. As we can see, the watermarks are still detectable in the meshes after the series of attacks.

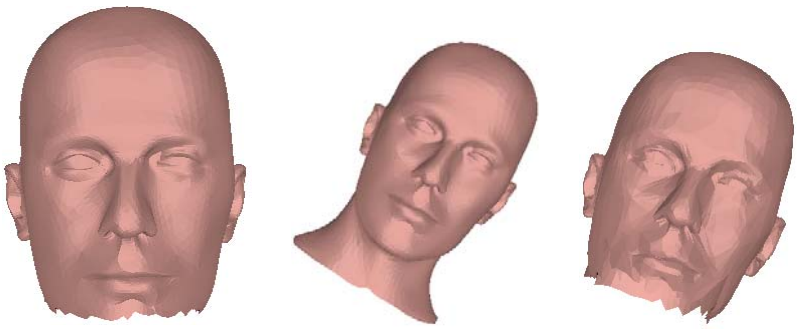
The experimental results show that our watermarking algorithm is robust under a variety of signal processing, including vertex reordering, noise addition, multiple watermarking, mesh simplification, multiresolution filtering and enhancement, single resolution filtering and enhancement, cropping, similarity transform and combination of the above attacks.



0.45% Noise | Second watermark | Simplification type A | Simplification type B



Multiresolution filtering | Multiresolution enhancement | Singleresolution filtering | Singleresolution enhancement



Cropping Similarity transform All attacks

Fig.6 Attacked Models

5. Conclusion and Future Work

Our watermarking algorithm has proven to be robust against a wide variety of attacks, including noise addition, multiple watermarking, simplification, filtering, enhancement, cropping, similarity transforms and combined attacks. Under the same level and kind of attacks, the robustness of our algorithm is comparable to the algorithm presented by Praun *et al.* in [10]. Also, our watermarking can resist some multiresolution processing attacks which Praun *et al.* did not test. Higher robustness can be achieved by increasing the watermark strength coefficient. Furthermore, our watermarking algorithm has the following characteristics:

(1) The watermark signal is embedded into the low frequency components of a mesh, as in many watermarking algorithms for static images [5,19,20]. Most mesh processing and attacks manipulate the high frequency components of a mesh. After all, it is the low frequency components that determine the main features of the mesh. So low frequency components become the natural host for watermarks.

(2) The algorithm can be naturally integrated with Guskov's multiresolution signal processing toolbox for meshes, hence no extra computation and data structures are needed. Another gain is, when processed by the mesh toolbox, the watermark can be well retained. We believe any robust watermarking scheme requires (to be integrated in) some kind of multiresolution framework, otherwise watermarks will not be robust against standard signal processing operations.

(3) The registration and resampling algorithms are simple yet effective, which enable the watermark detection of an attacked mesh whose geometry and topology have been changed.

(4) The watermark signal is a binary sequence, which enables the embedding of watermarks with real meaning.

Future work may include the following aspects:

- To apply other kinds of attacks and test possible failures of our algorithms. It is feasible to extend our registration method to undo general affine transformations, although it can only undo similarity transformations at present. But to remove complex non-linear/non-affine transformations is problematic, even with user assistance.
- To build a statistical model to choose the suitable threshold for the correlation detector to satisfy certain false positive and negtive requirements. For empirical claims would not work in copyright disputes. The possible

method could be either to convert to another correlation detector, or to exert some sort of probability constraints on the embedded watermarks so to have some hypothesis in developing a statistical model.

- To change our watermarking algorithm into a blind watermarking algorithm. As we can see, our watermark extraction requires access to the original mesh data. While in a blind watermarking algorithm, the original mesh is not needed in the watermark detection process. It will not only facilitate watermark detection (registration and resampling are not needed anymore), but also enhance the security of the original data. What's more, in blind watermarking, the advantage of our long binary watermarks over short real number watermarks will become obvious. Since we can provide readable ownership assertion to the end users directly. The biggest hurdle to revise 3D watermarking to blind watermarking still lies at the fact that 3D models lack a natural parametrization for frequency-based decomposition. We have to invent a multiresolution signal analysis method for meshes, which leaves the low frequency components generally unaffected by resampling, transformation and noise addition etc. The multiresolution mesh processing method presented by Guskov *et al.* [11] can not guarantee a stable coarser mesh after such processing and attacks, so we have to resort to the original mesh data for frequency information. One possible frequency decomposition technique we are considering is to exert some constraints in pyramid construction to get a "regular" coarser mesh, say by requiring one vertex in each volumetric cubic element. This vertex may correspond to the center of gravity of all the surfaces in the finer mesh enclosed in a cubic element. Such low frequency components should be somewhat stable so we can apply many blind watermarking schemes proposed in the image watermarking literature [19,20,21] to address blind watermarking for 3D meshes.
- To integrate various watermarking algorithms for different media types, including text, images, audio, video, and 3D models. The integrated watermarking method can find applications in virtual environments.

Acknowledgements: This research work is supported by the National Natural Science Foundation of China. We would like to thank Dr. Weiqun Cao and Dr. Xingguo Liu for their valuable discussions and supports, and Dr. Jim Little for his review and editing. We would also like to give our special thanks to the anonymous reviewers for their valuable comments and suggestions.

References

1. R.J.Anderson, editor. Information Hiding: First International Workshop. Lecture notes in computer science, Vol.1174, Springer Verlag, Berlin, May 1996.
2. Jian Zhao, Eckhard Koch, and Chenghui Luo. In Business Today and Tomorrow. *Communications of the ACM*, 1998, 41(7): 67-72.
3. Nasir Memon, Ping Wah Wong. Protecting Digital Media Content. *Communications of the ACM*, 1998, 41(7): 35-43.
4. W.Bender, D.Gruhl, N.Morimoto. Techniques for Data Hiding. *IBM Systems Journal*, 1996, 35(3&4): 313-335.
5. I.J.Cox, J. Kilian, T. Leighton, T. Shamoon. Secure Spread Spectrum Watermarking for Multimedia. *IEEE Trans. on Image Processing*, 1997, 6(12): 1673-1687.
6. Frank Hartung, Martin Kutter. Multimedia Watarmarking Techniques. *Proceedings of the IEEE*, 1999, 87(7): 1079-1107.
7. Ryutarou Ohbuchi, Hiroshi Masuda, Masaki Aono. Watermarking Three-Dimensional Polygonal Models Through Geometric and Topological Modifications. *IEEE Journal on Selected Areas in Communications*, 1998, 16(4): 551-560.
8. Oliver Benedens. Geometry-based Watermarking of 3D Models. *IEEE Computer Graphics and Applications*, 1999: 46–55.
9. Minerva Yeung, Boon-Lock Yeo. Fragile Watermarking of Three-Dimensional Objects. *Proceedings of International*

Conference on Image Processing, Vol. 2. IEEE Computer Society, Los Angeles, 1998: 442-446.

10. Emil Praun, Hugues Hoppe, Adam Finkelstein. Robust Mesh Watermarking. *SIGGRAPH'99 proceedings*, ACM Press, 1999: 49-56.
11. Igor Guskov, Wim Sweldensy, Peter Schroder. Multiresolution Signal Processing for Meshes. *SIGGRAPH'99 proceedings* ACM Press, 1999: 325-334.
12. P.J.Burt, E.H.Adelson. Laplacian Pyramid as a Compact Image Code. *IEEE Transactions on Communications*. 1983, 31(4): 532–540.
13. Hugues Hoppe. Progressive Meshes. *SIGGRAPH'96 Proceedings*, ACM Press, 1996: 99-108.
14. Michael Garland, Paul S. Heckbert. Surface Simplification Using Quadric Error Metrics. *SIGGRAPH'97 Proceedings*, ACM Press, 1997: 119-128.
15. W. H. Press, S. A. Teulkolsky, W. T. Vetterling, B. P. Flannery. Numerical Recipes in C, 2nd ed. Cambridge University Press, London, 1996.
16. Y. Chen, G. Medioni. Object Modeling by Registration of Multiple Range Images. *Image and Vision Computing*, 1992, 10(3): 145–155.
17. Jerome Maillot, Hussein Yahia, Anne Verroust. Interactive Texture Mapping. *SIGGRAPH'93 Proceedings*, ACM Press, 1993: 27-34.
18. Weiqun Cao, Hujun Bao, Qunsheng Peng. A Level of Detail Modeling by Merging Near-Coplanar Faces based on Gauss Sphere. *Journal of Software*, accepted. (in Chinese)
19. Jiri Fridrich, 2 Lt Arnold C. Baldoza, and Richard J. Simard. Robust Digital Watermarking Based on Key-Dependent Basis Functions. *Proc. of the International Information hiding Workshop, April.* 1998:143-157.
20. A. Piva, M. Barni, F. Bartolini, and V. Cappellini**.** DCT-based watermark recovering without resorting to the uncorrupted original image. *IEEE Signal Processing Society 1997 International Conference on Image Processing (ICIP'97)*, Santa Barbara, California, October 1997: volume I, 520-523.
21. J.'O.Ruanaidh, and Thierry Pun. Rotation, translation and scale invariant digital image watermarking. *IEEE Signal Processing Society 1997 International Conference on Image Processing (ICIP'97)*, Santa Barbara, California, October 1997: volume I, 536-539.

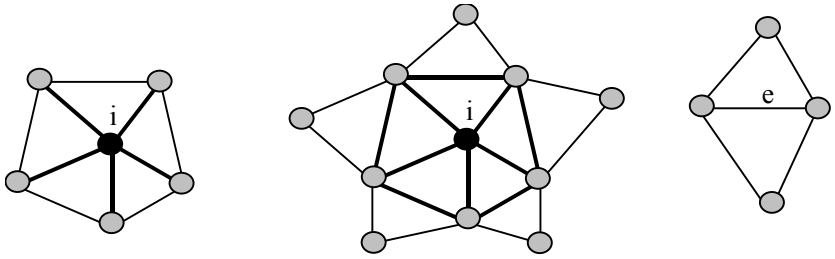
Appendix: Relaxation Operators and Burt-Adelson Pyramid

Most of the content of this Appendix is derived from [11], in which Guskov generalized the Burt-Adelson pyramid scheme to a mesh pyramid using his 3D non-uniform relaxation operator. We hope this will facilitate the understandings of our method.

(1) Some Definitions

First we define some notations for triangular meshes and topological neighborhood. Let $M = (P, K)$ represent a triangular mesh, where $P = \{p_i \in R^3 \mid 1 \le i \le N\}$ is a set of vertices, and $p_i = (x_i, y_i, z_i)$ is the coordinate of a vertex; *K* contains all the topological information, which indicates the adjacency relations.

Given a vertex *i*, an edge *e*, $V_1(i)$ denotes *i*'s 1-ring vertex neighborhood, $\varepsilon_1(i)$ denotes *i*'s 1-ring edge neighborhood, $V_2(i)$ denotes *i*'s 1-ring vertex neighborhood with flaps, $\varepsilon_2(i)$ denotes *i*'s 1-ring edge neighborhood with flaps, $w(e)$ denotes *e*'s vertex neighborhood. In Figure A1, gray vertices constitute vertex neighborhoods, and bold edges constitute edge neighborhood.



$V_1(i)$ $\varepsilon_1(i)$ $V_2(i)$ $\varepsilon_2(i)$ $w(e)$

Fig.A1 Neighborhood definitions

(2) Relaxation Operators

In [11], Guskov *et al.* presented a 3D relaxation operator for meshes:

$$Rp_i = \sum_{j \in V_2(i)} w_{i,j} p_j \,, \tag{a1}$$

where $w_{i,j}$ is computed as

$$w_{i,j} = -\frac{\sum_{\{e \in \varepsilon_2(i) | j \in w(e)\}} c_{e,i} c_{e,j}}{\sum_{e \in \varepsilon_2(i)} c_{e,i}^2}. \tag{a2}$$

According to the adjacency relations in Figure A2, the coefficients $c_{e,i}$ is calculated in one of the four ways

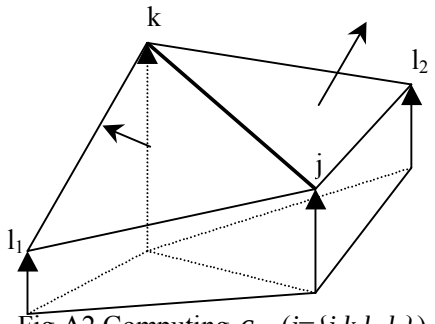


Fig.A2 Computing $c_{e,i}$ (i=$\{j,k,l_1,l_2\}$)

$$c_{e,l_1} = \frac{L_e}{A_{[l_1,k,j]}},\ \ c_{e,l_2} = \frac{L_e}{A_{[l_2,j,k]}},\ c_{e,j} = -\frac{L_e A_{[k,l_2,l_1]}}{A_{[l_2,k,j]} A_{[l_2,j,k]}},\ c_{e,k} = -\frac{L_e A_{[j,l_1,l_2]}}{A_{[l_2,k,j]} A_{[l_2,j,k]}}, \tag{a3}$$

where L_e is the length of the common edge e (the bold edge in Figure A2); A represents the signed area of the triangle formed by corresponding subscript vertices; $A_{[k,l_2,l_1]}$ and $A_{[j,l_1,l_2]}$ are the triangle areas after Δkl_2l_1 and Δjl_1l_2 are rotated into one plane, using the common edge e as a hinge.

(3) The Burt-Adelson pyramid and its construction

The Burt-Adelson (B-A) pyramid is a kind of Laplacian Pyramid scheme proposed by Burt and Adelson for compact image coding [12]. It is generalized into a mesh pyramid using the above relaxation operator in [11].

According to Figure 2 (in the main text), the following stages are needed to obtain $S^{(n-1)}$ from $S^{(n)}$:

(a) **Presmoothing**: Compute the new coordinates for n's 1-ring vertex neighborhood,

$$\forall j \in V_1^n(n): \qquad s_j^{(n-1)} = \sum_{k \in V_2^n(j)} w_{j,k}^{(n)} s_k^{(n)} \ ; \tag{a4}$$

Other vertex coordinates pass from $S^{(n)}$ to $S^{(n-1)}$ unchanged

$$\forall j \in V^{n-1} \setminus V_1^n(n): \qquad s_j^{(n-1)} = s_j^{(n)}. \tag{a5}$$

(b) **Downsampling**: Remove vertex *n* by a half-edge collapse.

(c) **Subdivision**: Use the coordinates in $S^{(n-1)}$ to compute subdivided coordinates $q_j^{(n)}$ for the vertices whose positions have been changed. The coordinates of vertex *n* just removed can be evaluated as

$$q_n^{(n)} = \sum_{j \in V_2^n(j)} w_{n,j}^{(n)} s_j^{(n-1)}. \tag{a6}$$

The coordinates of *n*'s 1-ring vertex neighborhood can be evaluated as

$$\forall j \in V_1^n(n): \qquad q_j^{(n)} = \sum_{k \in V_2^n(j) \setminus \{n\}} w_{j,k}^{(n)} s_k^{(n-1)} + w_{j,n}^{(n)} q_n^{(n)}. \tag{a7}$$

(d) **Detail computation**: Compute the details for vertex *n* and its 1-ring vertex neighborhood in a local frame $F^{(n-1)}$.

$$\forall j \in V_1^n(n) \cup \{n\}: \qquad d_j^{(n)} = F_j^{(n-1)}(s_j^{(n)} - q_j^{(n)}), \tag{a8}$$