Progressive Coding and Illumination and View Dependent Transmission of 3D Meshes Using R-D Optimization

Wei Guan, Jianfei Cai, *Senior Member, IEEE,* Juyong Zhang, Jianmin Zheng

***Abstract*— For transmitting complex 3D models over bandwidth-limited networks, efficient mesh coding and transmission are indispensable. The state-of-the-art 3D mesh transmission system employs wavelet-based progressive mesh coder, which converts an irregular mesh into a semi-regular mesh and directly applies the zerotree-like image coders to compress the wavelet vectors, and view-dependent transmission, which saves the transmission bandwidth through only delivering the visible portions of a mesh model.**

We propose methods to improve both progressive mesh coding and transmission based on thorough rate-distortion (R-D) analysis. In particular, by noticing that the dependency among the wavelet coefficients generated in remeshing is not being considered in the existing approaches, we propose to introduce a preprocessing step to scale up the wavelets so that the inherent dependency of wavelets can be truly understood by the zerotree-like image compression algorithms. The weights used in the scaling process are carefully designed through thoroughly analyzing the distortions of wavelets at different refinement levels. For the transmission part, we propose to incorporate the illumination effects into the existing view-depend progressive mesh transmission system to further improve the performance. We develop a novel distortion model that considers both illumination distortion and geometry distortion. Based on our proposed distortion model, given the viewing and lighting parameters, we are able to optimally allocate bits among different segments in real time. Simulation results show significant improvements in both progressive compression and transmission.

***Index Terms*— Progressive mesh coding, progressive mesh transmission, rate-distortion analysis, view dependent, illumination dependent, semi-regular meshes.**

I. Introduction

Three-dimensional (3D) mesh compression has become more and more essential in 3D technology especially for storage and transmission of complex models. With the increased popularity of networked graphics applications, 3D mesh model transmission has received more and more attention in the past few years. Considering the time-varying characteristic of wireless channels, progressive mesh coding is highly desired. With progressive compression techniques, a complex 3D mesh model only needs to be encoded once and can be transmitted and decoded at multiple bit rates.

In literature, many progressive mesh compression schemes have been proposed including [3]–[7]. In our research, we consider progressively compressing triangular meshes. A triangular mesh is typically irregular in connectivity. The latest progressive mesh coding technique is to convert irregular mesh into semi-regular meshes. Such a conversion process is called remeshing. For example, Lee *et al.* provided an efficient remeshing technique called the MAPS algorithm [8]. In that scheme, an irregular mesh is first simplified into a base mesh. Each triangle in the original mesh can be mapped into an “internal” triangle within a base triangle. Then, the base mesh is subdivided, and the new vertices obtained through subdivision are mapped back to the vertices in the original mesh. Finally, the base mesh and these mapped vertices form a semi-regular mesh. In addition, the generated semi-regular mesh can be efficiently compressed using wavelet based image coding schemes [7].

Based on the generated semi-regular mesh that has subdivision connectivity, Sim *et al.* further developed a rate-distortion (R-D) optimized view-dependent mesh streaming system [9]. The system consists of three parts: preprocessing, progressive mesh coding, and view-dependent transmission. In preprocessing, an irregular mesh is converted into a semi-regular mesh using the MAPS algorithm. Then, the mesh is partitioned into many segments. Each segment is progressively encoded using the SPIHT algorithm. Finally, given the view direction, the system optimally allocates bits for each segment based on the developed R-D model and progressively transmits them to the receiver side. Hoppe [10] also proposed a view-dependent refinement algorithm, where a mesh is represented as a PM (progressive mesh). Initially, a coarsest mesh is rendered. Then the algorithm iteratively checks each vertex whether it needs to be split (refined) or not. A vertex will only be refined if it is within the viewing frustum, facing towards the viewer, and the screen-space error is larger than a predefined threshold.

In this paper, we first propose an improved wavelet-based 3D progressive mesh coder. We notice that the vector wavelet coefficients generated during remeshing are dependent, which is different from wavelet coefficients in image compression. Such a characteristic of semi-regular meshes is not being considered in the existing progressive mesh coders. Our basic idea is to introduce a preprocessing step to scale up the vector wavelets generated in remeshing so that the inher-

Manuscript received Nov. 28, 2008; revised July 14, 2009. This research was partially supported by Singapore A*STAR SERC Grant (062 130 0059). Parts of this paper were presented in [1], [2]. This paper was recommended by Associate Editor L. Onural.

The authors are with the School of Computer Engineering, Nanyang Technological University, BLK N4, #02c-75 Nanyang Avenue, Singapore 639798 (Email: {guan0006, asjfcai, s070051, asjmzheng}@ntu.edu.sg; Phone: +65 6790 6150; Fax: +65 6792 6559). Contact author: Jianfei Cai.

ent dependency of wavelets can be truly understood by the zerotree-like image compression algorithms. Although the idea is simple, the weights used in the scaling process are obtained by thoroughly analyzing the distortions of wavelets at different refinement levels. Experimental results show that compared with the state-of-the-art wavelet based 3D mesh compression scheme, our proposed mesh coder can achieve significant quality improvement with only slight complexity increase.

Moreover, we further propose an illumination and view dependent progressive mesh transmission system. Although the view dependency has been considered in previous work such as [9]–[12], they did not take into account the illumination effects, which results in rendering the perceptually unimportant over-dark or over-bright portions and thus wastes the transmission bandwidth. In this paper, we consider both illumination distortion and geometry distortion. Through thorough derivation and simplification, we reach a practically feasible distortion model, which can be calculated in real time. Experimental results show that significant gain can be achieved when the illumination effects are being considered in addition to the view dependency.

The rest of the paper is organized as follows. Section II reviews the related work. Section III presents our proposed progressive mesh coder. Section IV describes our proposed illumination and view dependent progressive mesh transmission scheme emphasizing on distortion modeling. We show the simulation results in section V and conclude the paper in section VI.

II. RELATED WORK

This research can be regarded as an enhancement to the state-of-the-art progressive mesh coding and transmission framework proposed in [9]. In this section, we provide the background information related to the the major components used in [9].

A. Remeshing

In [9], the MAPS algorithm [8] is adopted to convert an irregular triangular mesh into a semi-regular mesh with subdivision connectivity, which is called a remeshing process as illustrated in Fig. 1. In particular, the original mesh is simplified into a base mesh and the base mesh is then refined into a semi-regular mesh through a series of subdivision and vertices adjustments. The butterfly subdivision (see Fig. 2(a)) is used in [9] to predict the next level vertices from the current level. The prediction errors, i.e. the difference between original vertices and predicted vertices, are treated as wavelets (see Fig. 2(b)). Encoding wavelets will lead to higher compression ratio than encoding the original vertices. This is because the wavelets are usually small and less correlated. In summary, the remeshing process generates a base mesh and a sequence of wavelets, based on which the semi-regular mesh can be fully reconstructed.

B. Wavelet Based Progressive Mesh Coding

In the coding process, the base mesh and the wavelets are encoded separately. Typically, the base mesh is encoded losslessly by a single-rate mesh compression scheme such as the MPEG-4 3DMC algorithm while the wavelets are encoded progressively by a wavelet-based image coder such as the zerotree coder [13] or the SPIHT coder [14]. In particular, in [9], vertices are organized into edge-based trees to have the hierarchical parent-offspring relationship so that zerotree-like coders can be applied. Each vertex on base mesh edges becomes a root node of an edge-based tree. Fig. 3 shows the structure of a edge-based tree. In this way, all the vertices can be covered by edge-based trees without overlapping.

Recall that a wavelet here is the displacement between the predicted vertex and the corresponding vertex on the original model. In other words, each wavelet is actually a displacement vector $(\Delta x, \Delta y, \Delta z)$. In [7], three independent zerotree coders are applied to encode the three components separately. In [9], the SPIHT algorithm is applied to first encode the scalar of $sgn(\Delta x) \cdot \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2}$, followed by the encoding of Δy and Δz if the scalar is significant.

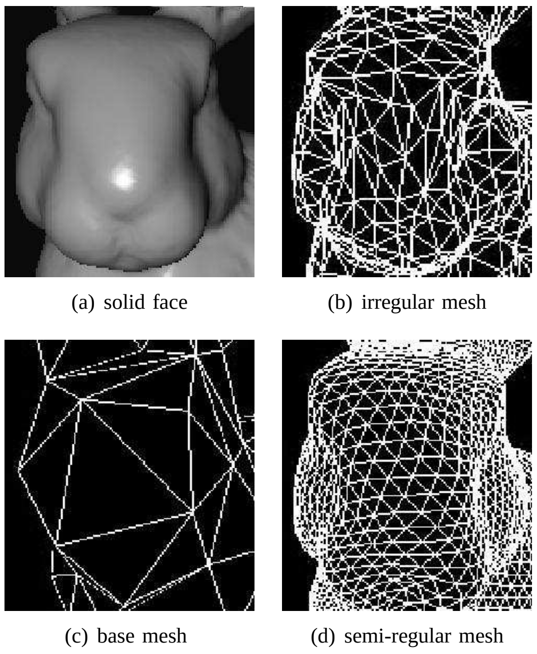


Fig. 1. The remeshing process of bunny head.

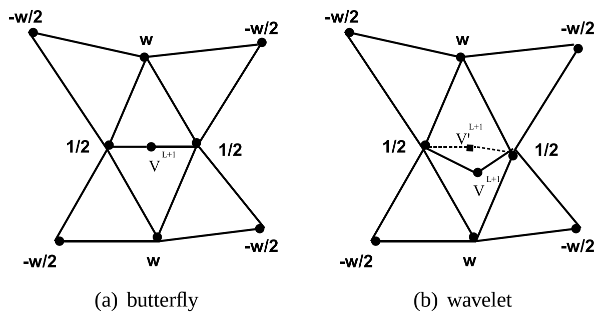


Fig. 2. The butter-fly subdivision. In (b), the wavelet is the displacement between original and predicted vertex position, i.e. $V^{L+1} - V'^{L+1}$.

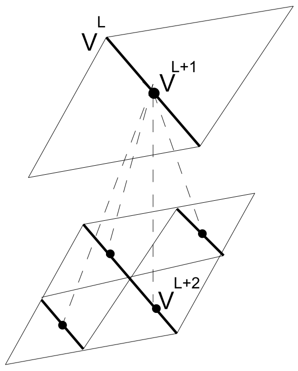


Fig. 3. An example of an edge-based tree, where V^{L+2} is one of the four offsprings of V^{L+1}.

C. R-D Optimized Transmission

In [9], each edge-based tree is treated as a segment and an R-D function is developed for each segment. We will discuss the R-D modeling in detail in the subsequent sections. Suppose there is an R-D function for each segment, denoted as $D_k = f_k(R_k), k = 1, 2, ..., n$, where D_k and R_k are the distortion and the rate for the kth segment respectively, and n is the total number of segments. During the transmission, the key question is how to optimally allocate bits to each segment or tree so that the total available bit budget can be well utilized. Such a bit allocation problem can be formulated as follows: given a certain bit budget R, how to find the optimal R_k values that minimize $\sum_{k=1}^{n} D_k = \sum_{k=1}^{n} f_k(R_k)$ subject to $\sum_{k=1}^{n} R_k = R$.

This is an n-dimension constrained optimization problem. A common solution is to use the Lagrange multiplier method. In particular, minimizing the distortion is the same as minimizing the following equation with some value of λ,

$$\sum_{k=1}^{n} f_k(R_k) + \lambda(\sum_{k=1}^{n} R_k - R). \tag{1}$$

By differentiating the E.q. (1) with respect to $R_k, k = 1, 2, ..., n$, we obtain n equations: $f_k^{'}(R_k) + \lambda = 0, k = 1, 2, ..., n$. Together with the constraint: $\sum_{k=1}^{n} R_k = R$, theoretically the $n + 1$ variables, i.e. λ and $R_k, k = 1, 2, ..., n$, can be derived. In practice, there is usually no derivable formula for the above optimization problem. Instead, the greedy algorithm is typically used to search the optimal solution in a discrete solution space.

D. Quality Measurement

For 3D mesh coding, one of the key challenges is how to measure the quality. A popular metric is the Hausdorff distance, which measures the distances between geometric shapes. However, this type of metric is not the same as the perceived visual quality since it is defined purely from the geometry point of view. Several research studies have been conducted to find more appropriate quality metrics. In particular, Hoppe [10] proposed a view-dependent mesh metric called screen-space geometric error, which measures the geometric error after projecting geometry models in the viewing direction. Luebke *et al.* [15] pointed out that the silhouette regions are more important than interior regions and employed a tighter screen-space error threshold in the simplification process. Reddy [16] proposed to select levels of detail (LODs) according to a fundamental perceptual model, and an updated version of the perceptual model was further proposed in [17]. Although the perceptual model is not comprehensive, it does achieve some desirable effects such as preserving silhouette boundaries and high contrast details. However, the perceptual model requires tremendous computations.

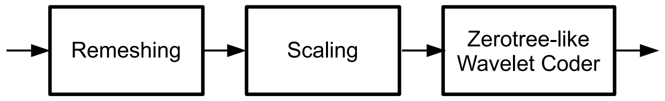


Fig. 4. The diagram of our proposed wavelet based progressive mesh coder.

III. R-D Optimized Wavelet-based Progressive Mesh Coding

A. Problem Statement

As described in Section II-B, the existing wavelet based 3D semi-regular mesh coders directly extend the state-of-the-art wavelet-based image coders to compress the wavelet vectors generated in the remeshing process. The main problem of these approaches is that the particular properties of semi-regular meshes are not being taken into consideration. Specifically, in image compression, wavelet coefficients of an image at different levels are independent from each other. A distortion of a coefficient at a coarser level does not affect the distortion of any other coefficient at a finer level. However, in 3D semi-regular mesh compression, the situation becomes totally different. That is, the wavelets in the lth level affect the vertex positions in the $(l + 1)$th level and the subsequent levels due to the subdivision operations. Thus, with the same error, a wavelet at a lower level contributes more to the total distortion than a wavelet at a higher level.

It is well known that the principle of zerotree-like coders is to send the higher order bits of the larger magnitude coefficients first. Directly applying zerotree-like coders to encode semi-regular meshes only considers the wavelet magnitudes without taking into account the dependency and the inherent unequal importance among different levels of wavelets. Thus, the generated progressive bitstream is not embedded since it is not guaranteed that the earlier portions of the progressive mesh bitstream are always more important than the later parts.

B. R-D Optimized Progressive Coding

Fig. 4 shows the diagram of our proposed wavelet based progressive mesh coder. Basically, we add one new component, "scaling", between the remeshing component and the coding component. The purpose of the scaling component is to give different weights to wavelets in different levels so that their magnitudes can truly represent their importance. One advantage of such a system is that it can directly employ the exiting remeshing and zerotree-like coding algorithms.

The key question in our proposed mesh coder is how to set the weights for different wavelets. In order to address this question, we need to analyze the distortion. In our research, we consider using geometric distortion to measure the reconstructed mesh quality. The geometry distortion is typically defined as the mean square error between original vertices and reconstructed vertices, i.e.

$$D(M', M) = \frac{1}{|V|} \sum_{v_i \in M, v_i' \in M'} \|v_i - v_i'\|^2 \tag{2}$$

where $|V|$ is the total number of vertices, and v_i and v_i' denote the vertices in the original semi-regular mesh M and the reconstructed semi-regular mesh M', respectively.

Since in the progressive representation vertices are represented by wavelets, it is highly desired to compute the distortion through wavelets so that repeatedly reconstructing the vertices can be avoided. As pointed out in [9], the mean square error of vertices $\|v_i - v_i'\|^2$ can be approximated by the mean square error of wavelets $\|w_i - w_i'\|^2$. However, they are not exactly equal due to the dependency among the vertices at different levels.

In [9], a distortion model was proposed, which considers the error propagation effect due to subdivision. In particular, as shown in Fig. 5(a), the distortion for the center solid circle in the lth level will propagate to the other three types of vertices in the $(l+1)$th level with the weights of 1/2, w and $-w/2$, respectively. Considering the valence of six feature for a semi-regular mesh, the error propagation effect (including itself) from a particular level to the next level is approximated by a weighting factor [9]

$$\begin{aligned} W &= 1^2 + 6 \times (\frac{1}{2})^2 + 6 \times w^2 + 12 \times (-\frac{w}{2})^2 \\ &= 2.5 + 9w^2 \end{aligned} \tag{3}$$

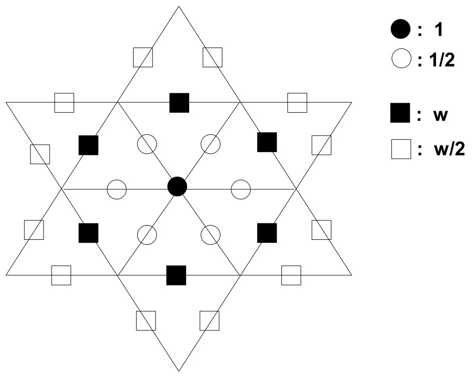
where w ranges from 0 to 1/8. Further taking into account the error propagation among multiple levels, the weight for a distortion in the lth level is defined as [9]

$$W_l = W^{L-l}, \quad l = 1, 2, \ldots, L \tag{4}$$

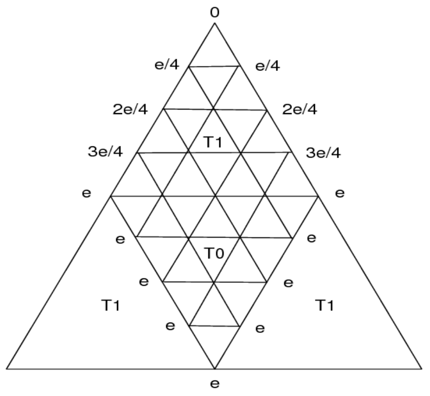
where L is the total number of subdivision levels.

Note that although the authors in [9] did consider the dependency among the wavelets at different levels in their distortion model, they only applied it for the distortion estimation and did not use it to improve the coding performance. Moreover, their derived weights shown in Eqs. (3) and (4) are rough approximation because they assume the error propagations of individual vertices are the same. The subsequent analysis further details the inaccuracy of such simple weights.

We first consider the simplest case, i.e. $w = 0$, and assume all the wavelets at level 1 (base mesh is at level 0) have an uniform error e (see Fig. 5(b)). We study how the distortions at level 1 are propagated to subsequent levels. In particular, considering a vertex on an edge is shared by two adjacent triangles, it is clear that the distortion at level 1 for this particular triangle is $3e^2/2$. As shown in Fig. 5(b), after the first level subdivision, the base triangle is divided into two types of triangles: T_0 and T_1. It can be observed that the vertices generated at different subdivision levels in T_0 always have the same error e while the new vertices in T_1 are of



(a) error propagation



(b) wavelet errors after three times subdivision

Fig. 5. The error propagation weights and wavelet errors.

different wavelet errors, having a pattern of $(k+1)$ vertices with an error of $(k/2^{L-1})e$, $k = 1, 2, \ldots, L$. Adding all the distortions together and removing the overlapping, we derive the total distortion for this particular triangle as

$$d_L = (\frac{5}{16}4^L + \frac{1}{4})e^2 \tag{5}$$

It also means that the distortion of $3e^2/2$ introduced at level l will result in a total distortion of d_{L-l+1}. Therefore, the weight for a distortion at the lth level becomes

$$W_l = \frac{d_{L-l+1}}{3e^2/2} = \frac{5}{6}4^{L-l} + \frac{1}{6}, \quad l = 1, 2, \ldots, L \tag{6}$$

Finally, the overall distortion is calculated as

$$D = \sum_{l=1}^{L} \sum_{w_i \in M^l} W_l \cdot \|w_i - w_i'\|^2 \tag{7}$$

where $w_i \in M^l$ means w_i is used in the lth level refinement.

We would like to point out that although the weight in Eq. (6) is derived in the case with the subdivision parameter $w = 0$, it can be directly used for the other cases with $w \neq 0$. This is because w ranges from 0 to 1/8 and the corresponding error propagation is insignificant as shown in Eq. (3).

In our implementation, with the derived weight in Eq. (6), the scaling component in Fig. 4 changes a wavelet w_i, $w_i \in M^l$, into $\sqrt{W_l}w_i$ before the SPIHT encoding. To encode a vector wavelet coefficient ($\triangle x$, $\triangle y$, $\triangle z$) for a vertex, we propose the following method. We first encode the scalar of $\triangle l = sgn(\triangle x) \cdot \sqrt{(\triangle x)^2 + (\triangle y)^2 + (\triangle z)^2}$, where $sgn(\triangle x)$ denotes the sign of $\triangle x$. After the scalar is encoded, $\triangle y/\triangle l$ and $\triangle z/\triangle l$ are lossless encoded with entropy coding. In this way, during decoding, the error caused by compression is distributed to $\triangle x$, $\triangle y$, and $\triangle z$ in proportion.

IV. R-D Optimized Illumination and View Dependent Transmission

A. Progressive Transmission System

Now, we consider applying the developed 3D mesh coder for progressive 3D mesh transmission. Our key idea is to consider the view and illumination dependencies. Specifically, we try to avoid the transmission of the invisible portions and avoid the refinement of the perceptually unimportant over-dark or over-bright portions of a 3D model. In order to do this, a 3D mesh needs to be partitioned into many segments. Fortunately, the adopted 3D mesh coder naturally organizes vertices into edge-based trees, each of which can be considered as a segment.

In particular, in our research, a semi-regular mesh is divided into two components: base mesh and edge-based forest. The vertices from the first time subdivision of the base mesh are taken as roots for each edge-based tree. Clearly, the number of trees in the forest is the same as the number of edges in the base mesh. We use the MPEG-4 3DMC (3D mesh coding) algorithm to encode the base mesh while employing our proposed wavelet-based progressive mesh coder to encode each edge-based tree separately. In this way, the base mesh is losslessly transmitted to the receiver, while the edge-based trees are progressively delivered according to the available bit budget.

The key question is how to optimally allocate bits to each segment or tree so that the total available bit budget can be well utilized, as discussed in section II-C. In other words, we need to decide which segment should be refined at which level. In order to achieve that, we need to develop the R-D function for each segment that takes the illumination and view dependencies into consideration.

B. Illumination and View Dependent Distortion Modeling

In our research, we define the overall distortion as a combination of geometry distortion and illumination distortion, i.e.

$$D = D^u + \lambda D^s, \tag{8}$$

where we use the screen-space distortion D^s to measure the geometry distortion, D^u denotes the illumination distortion, and λ is a user defined weight. In the following, we describe our developed models for D^s and D^u in detail.

1) Geometry Distortion: In section III, we have derived the geometry distortion in Eq. (7). Now, we consider the factor of viewing direction. In our research, we assume that the viewer is located at a far distance and thus orthogonal projection is used, i.e., the viewing direction is the same for all the vertices.

Let V denote the normalized viewing direction. We use the screen-space error metric, which calculates the geometric error

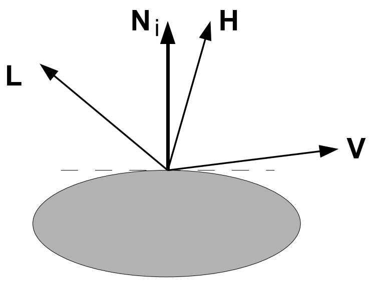


Fig. 6. The Blinn-Phong shading model.

projected on the screen. Specifically, the total view-dependent screen space error for the kth segment is

$$\begin{aligned} D_k^s &= \sum_{l=1}^{L} \sum_{v_i \in M_k^l} a_i \cdot |(v_i - v_i^{'}) \times V|^2 \\ &= \sum_{l=1}^{L} \sum_{v_i \in M_k^l} a_i \cdot W_l \cdot |(w_i - w_i^{'}) \times V|^2 \end{aligned} \tag{9}$$

where a_i is either 1 or 0, indicating whether the vertex is visible or not, W_l is the weight derived in Eq. (6), and $v_i \in M_k^l$ means v_i is a new vertex introduced in the lth refinement of the kth segment.

We can easily obtain a_i by checking the angle between the vertex normal N_i with the view direction V. If the angle is less than $\pi/2$, the vertex is visible; otherwise, it is invisible. To simplify the computation, for all the vertices within one segment k, we assign the same normal direction N^k since the vertices in one segment are around the the neighborhood of a base mesh edge and their normal directions are close to each other. We calculate N^k as the average of two neighboring base triangle normals. In this way, the computation of a_i can be done as follows:

$$a_i = \begin{cases} 1, & (N^k \cdot V) > 0; \\ 0, & (N^k \cdot V) \leq 0. \end{cases} \tag{10}$$

2) Illumination Distortion: In this study, we use the Blinn-Phong shading model [18] as the illumination model (shown in Fig. 6). Let L be the normalized lighting vector for a directional light source, and N_i be the normal vector of the surface at v_i. The intensity I_i at vertex v_i is given by

$$I_i = \begin{cases} I_a k_a + I_d k_d (N_i \cdot L) + I_s k_s (N_i \cdot H)^m, & (N_i \cdot L) > 0; \\ I_a k_a, & (N_i \cdot L) \leq 0; \end{cases} \tag{11}$$

where k_a, k_d and k_s are the ambient, diffuse and specular material coefficients, I_a, I_d and I_s are the corresponding luminance, and H_i, called the halfway vector, is defined as $\frac{L+V}{|L+V|}$.

The vertex position error of v_i will affect the normal vector at v_i, and thus the illumination of its neighborhood will also be affected. We define the illumination distortion d_i at v_i as

$$d_i = b_i \cdot S_i \cdot (I_i - I_i^{'})^2 \tag{12}$$

where $I_i^{'}$ is the corresponding intensity for the reconstructed vertex $v_i^{'}$, S_i is the surrounding triangle area, i.e. $S_i =$

$\sum_{j=1}^{6} S_{i,j}$, and b_i is the projection of S_i on the screen, which is defined as

$$b_i = \begin{cases} N^k \cdot V, & (N^k \cdot V) > 0; \\ 0, & (N^k \cdot V) \le 0. \end{cases} \tag{13}$$

Based on the illumination model in Eq. (11), the illumination error can be derived as follows:

$$\begin{aligned} I_i - I_i^{'} &= I_d k_d (N_i \cdot L) + I_s k_s (N_i \cdot H)^m - I_d k_d (N_i^{'} \cdot L) \\ &\quad - I_s k_s (N_i^{'} \cdot H)^m \qquad (14) \\ &= I_d k_d [(N_i - N_i^{'}) \cdot L] + I_s k_s [(N_i \cdot H)^m - (N_i^{'} \cdot H)^m] \\ &\approx I_d k_d [(N_i - N_i^{'}) \cdot L] + I_s k_s m [(N_i - N_i^{'}) \cdot H](N_i \cdot H)^{m-1} \end{aligned}$$

where $N_i^{'}$ is the corresponding normal for the reconstructed vertex $v_i^{'}$ and the last step approximation is based on first-order Taylor expansion.

In order to compute the illumination error in Eq. (14), we need to compute the normal vector change $N_i - N_i^{'}$. The common way to compute a vertex normal is through averaging its surrounding triangle normals, which requires to reconstruct vertex positions. Such a method is computationally prohibitive since we need to consider all the possible reconstructed vertex positions. In order to avoid that, we simply approximate a vertex normal as a weighted average of the surrounding edges, i.e.

$$N_i \approx \frac{1}{6} \sum_{j=1}^{6} \frac{(v_i - v_{i,j})}{E_{i,j}} \tag{15}$$

where $v_{i,j}, j = 1, 2, ..., 6$ are the six surrounding vertices for a vertex v_i, and $E_{i,j}$, the edge length of $(v_i - v_{i,j})$, is to reduce the sensitivity of the calculation by the edge length. According to Eq. (15), the normal vector change can be approximately derived as

$$\begin{aligned} N_i - N_i^{'} &\approx \frac{1}{6} \sum_{j=1}^{6} \frac{(v_i - v_{i,j}) - (v_i^{'} - v_{i,j})}{E_{i,j}} \\ &= \frac{1}{6} \sum_{j=1}^{6} \frac{v_i - v_i^{'}}{E_{i,j}} \\ &\approx \frac{w_i - w_i^{'}}{E_i} \qquad (16) \end{aligned}$$

where $\frac{1}{E_i} = \frac{1}{6} \sum_{j=1}^{6} \frac{1}{E_{i,j}}$.

Note that Eq. (15) is by no means an accurate method for normal calculation. However, it works fine for our particular purpose, i.e. computing the illumination distortion, due to the following reasons. First, it is a good normal approximation for the cases where each vertex has a valence of six and the vertices are distributed more or less uniformly, which is quite common for semi-regular meshes. Second, although the approximation of Eq. (15) causes problems in the cases of locally planar regions, where it produces normal vectors lying in the planes, and concave regions, where we obtain the normals in opposition direction, it does not have a great effect on computing the magnitude of the normal difference, which plays the key role for computing the illumination distortion.

Substituting Eq. (16) back to Eq. (14) together with using $(N^k \cdot H)^{m-1}$ to approximate $(N_i \cdot H)^{m-1}$, and further substituting Eq. (14) back to Eq. (12), we derive

$$\begin{aligned} d_i &= b_i \frac{S_i}{E_i^2} \{ c_1^2 [(w_i - w_i^{'}) \cdot L]^2 + c_2^2 [(w_i - w_i^{'}) \cdot H]^2 \\ &\quad + 2c_1 c_2 [(w_i - w_i^{'}) \cdot L][(w_i - w_i^{'}) \cdot H] \} \qquad (17) \\ &= b_i \frac{S_i}{E_i^2} \{ c_1^2 [(w_i - w_i^{'}) \cdot L]^2 + c_2^2 [(w_i - w_i^{'}) \cdot H]^2 \\ &\quad + 2c_1 c_2 [(w_i - w_i^{'}) \cdot H^{'}]^2 - c_1 c_2 |w_i - w_i^{'}|^2 [1 - (L \cdot H)] \}, \end{aligned}$$

where $H^{'} = \frac{L+H}{|L+H|}$, $c_1 = I_d \cdot k_d$, $c_2 = I_s \cdot k_s \cdot m (N^k \cdot H)^{m-1}$, and the derivation of the last step is based on the property of dot product derived in Appendix.

Finally, the total illumination distortion for the kth segment can be expressed as

$$D_k^u = \sum_{l=1}^{L} \sum_{v_i \in M_k^l} W_l \cdot d_i, \tag{18}$$

where W_l is to compensate the approximation of $\|v_i - v_i^{'}\|^2$ by $\|w_i - w_i^{'}\|^2$.

3) Distortion Model Simplification: For practical applications of 3D mesh transmission, given the viewing and lighting parameters, we need to perform optimal bit allocation among different segments in real-time. The distortion model described by Eqs. (9) and (18) are computationally intensive. This is because for one set of viewing and lighting parameters, we need to compute the distortions for each vertex under different bit rates, which is impractical. Therefore, in this subsection, we further develop a simplified distortion model. The basic idea is to separate the wavelet distortion from the viewing and lighting parameters. In particular, we approximate any dot product in the form of $[(w_i - w_i^{'}) \cdot A]^2$ into $|w_i - w_i^{'}|^2 \cdot (N^k \cdot A)^2$, where A denotes a vector and N^k is the normal vector for the kth segment.

With such a simplification, the screen-space distortion for the kth segment becomes,

$$D_k^s = a_k |N^k \times V|^2 \sum_{l=1}^{L} \sum_{v_i \in M_k^l} W_l |w_i - w_i^{'}|^2. \tag{19}$$

The illumination distortion for the kth segment becomes,

$$\begin{aligned} D_k^u &= b_k \{ c_1^2 (N^k \cdot L)^2 + c_2^2 (N^k \cdot H)^2 + 2c_1 c_2 (N^k \cdot H^{'})^2 \\ &\quad - c_1 c_2 [1 - (L \cdot H)] \} \cdot (\sum_{l=1}^{L} \sum_{v_i \in M_k^l} W_l \frac{S_i}{E_i^2} |w_i - w_i^{'}|^2), \end{aligned} \tag{20}$$

where a_k and b_k are defined in Eqs. (10) and (13).

To compute S_i and E_i, we make use of the two base triangles and their edges since in the one-to-four subdivision the shape of the four off-spring triangles in the next level is similar to that of the two base triangles. In particular, we denote the two base triangle areas as $S_{k,1}$ and $S_{k,2}$, and the surrounding edge length as $E_{k,j}, j = 1, 2, .., 6$, as shown in Fig. 7. For a particular vertex v_i, we have the following approximations:

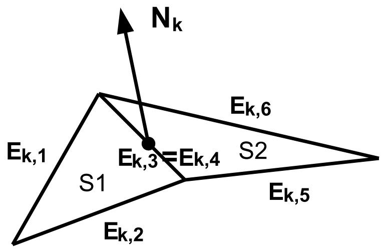


Fig. 7. The estimations of edge length and triangle area.

$\frac{1}{E_i} = \frac{1}{6}\sum_{j=1}^{6}\frac{1}{E_{k,j}} \times 2^l$ and $S_i = \frac{S_{k,1}+S_{k,2}}{2} \times \frac{1}{4^l}$. Thus, we derive

$$\frac{S_i}{E_i^2} = \frac{S_{k,1}+S_{k,2}}{72 \times (\sum_{j=1}^{6}\frac{1}{E_{k,j}})^2}, \quad (21)$$

which is a constant for all the vertices within one segment.

Based on the developed simplified distortion model, for each segment we only need to compute $\sum_{l=1}^{L}\sum_{v_i \in M_k^l} W_l|w_i - w_i^{'}|^2$ once, which is independent of view and lighting parameters and can be pre-computed. During the online transmission, only the view and lighting dependent portions in Eqs. (19) and (20) need to be calculated. In this way, the view and lighting dependent transmission can be achieved in real time.

V. Simulation Results

A. Results of Progressive Mesh Compression

We implement the proposed wavelet based 3D mesh coder by adopting the MAPS algorithm [8] for remeshing and SPIHT for encoding wavelet coefficients. We test on two 3D graphics models, "Venus" and "Armadillo".

First, we evaluate the accuracy of our proposed distortion model. Fig. 8 shows the results of the distortion estimation, where our proposed distortion model uses the derived weight given in Eq. (6) while the previous model in [9] uses the weight given in Eq. (3). We would like to point out that the distortion model is a function of the wavelet distortions. Given a bit rate, it is needed to perform SPIHT decoding in advance to obtain the individual wavelet distortions. It can be seen from Fig. 8 that the estimated distortions by our proposed model closely match the actual distortions at each bit rate, much more accurate than using the previous model. This is mainly because we assign more accurate weights to the wavelet distortions at each level. Note that all the distortion results here are normalized with respect to the distortion of the base mesh.

Second, we compare the wavelet based 3D mesh coders with and without the scaling components. The rate-SNR performance is shown in Fig. 9 and the reconstructed models at a particular bit rate are shown in Fig. 10. From Fig. 9, we can see that our proposed coder significantly outperforms the state-of-the-art coder proposed in [9] with $2 \sim 3$ dB gain at most of the bit rates. The reconstructed models in Fig. 10 further demonstrate that our proposed coder achieves much better perceived visual quality.

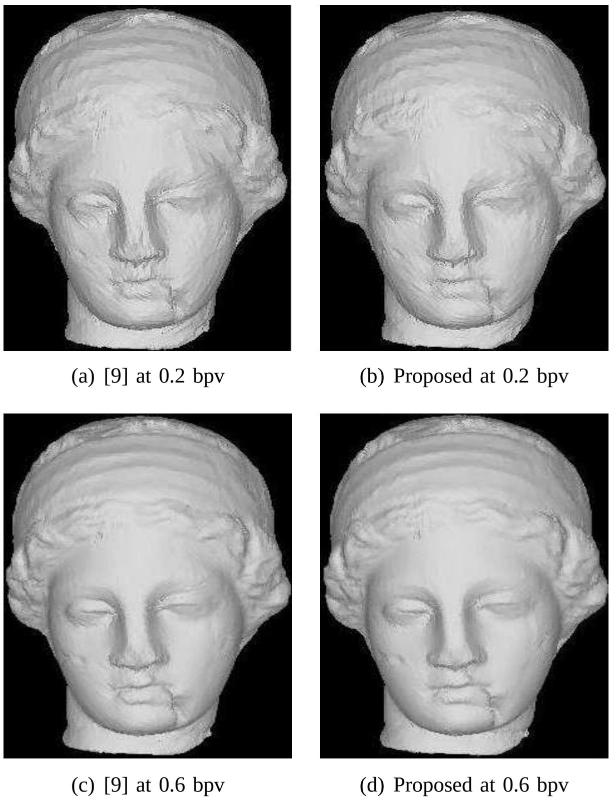


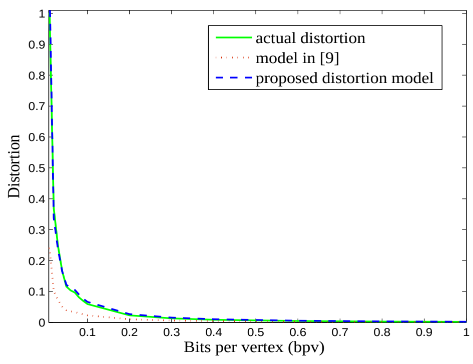
Fig. 10. The reconstructed models for Venus.

B. Results of View and Illumination Dependent 3D Mesh Transmission

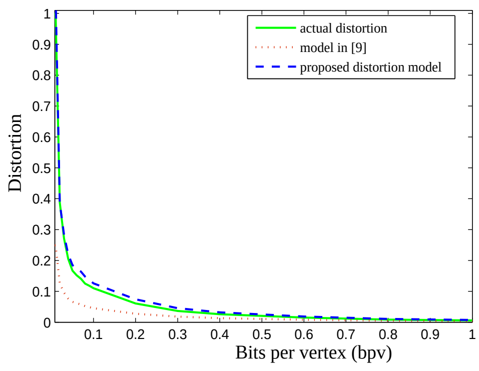
In this section, we test the performance of our proposed distortion model that considers both view and illumination dependencies. To illustrate the importance of the illumination distortion, we compare the approaches with and without the consideration of the illumination distortion. In particular, one is based on Eq. (19) and the other is based on both Eqs. (19) and (20). For simplicity, we detach the mesh coding part and directly select vertices for transmission.

Fig. 11 shows the comparison results. It can be seen that under the same number of vertices the approach considering illumination can achieve much better visual quality. This is because it gives more refinement for the important parts while allocating less number of vertices for unimportant parts. On one hand, both approaches consider the view dependency and thus do not refine the invisible parts, as shown in Fig. 11(a). On the other hand, the approach considering illumination gives higher priorities to the parts with high contrast, which leads to the better performance. In particular, the invisible back side is not refined at all for a front view direction (see Fig. 11). For the front side, the approach without considering the illumination heavily refines the parts that contribute to the boundaries equally throughout the face. However, with shading enabled, some boundaries become visually more important than others. The approach considering illumination gives more refinement for these visually important boundary parts. The experimental results for other models are shown in Fig. 12.

Fig. 13 gives the progressive transmission performance with

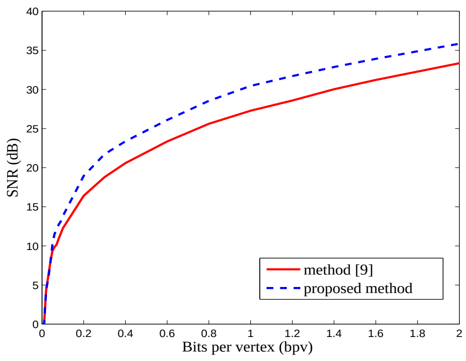


(a) Venus

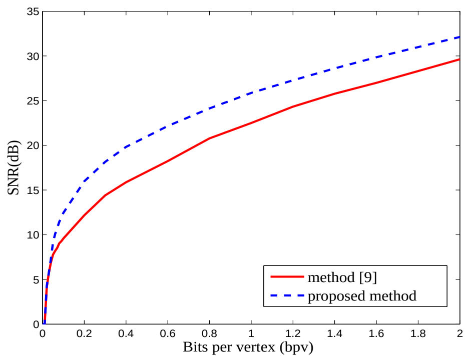


(b) Armadillo

Fig. 8. Distortion estimation.

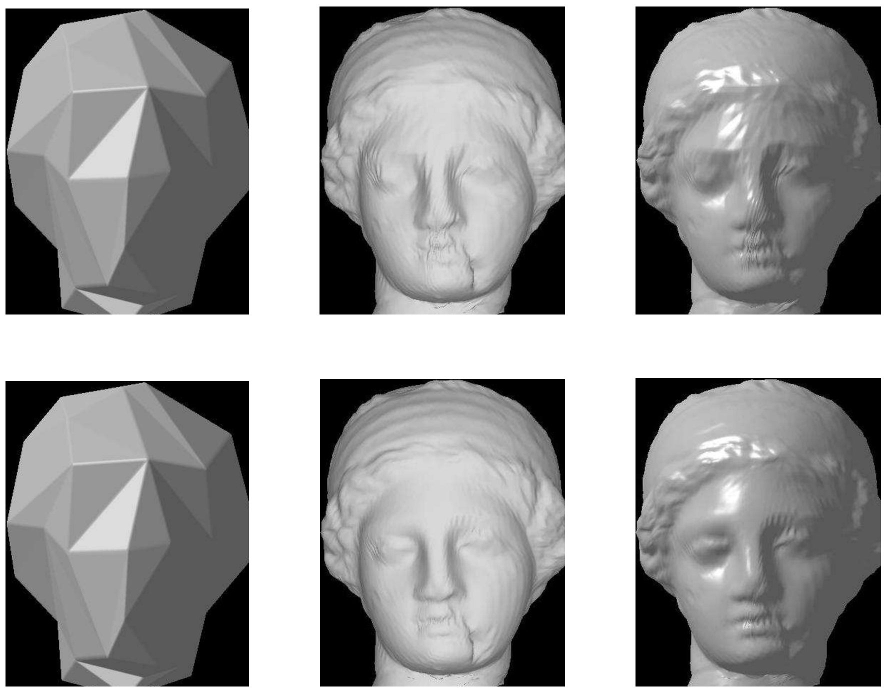


(a) Venus



(b) Armadillo

Fig. 9. The comparison of the rate-SNR performance.



(a) invisible back side (b) visible front side with illumination shading disabled (c) visible front side with illumination shading enabled

Fig. 11. The performance comparison for Venus with a bandwidth of 20% vertices. Top: without considering illumination. Bottom: considering illumination.

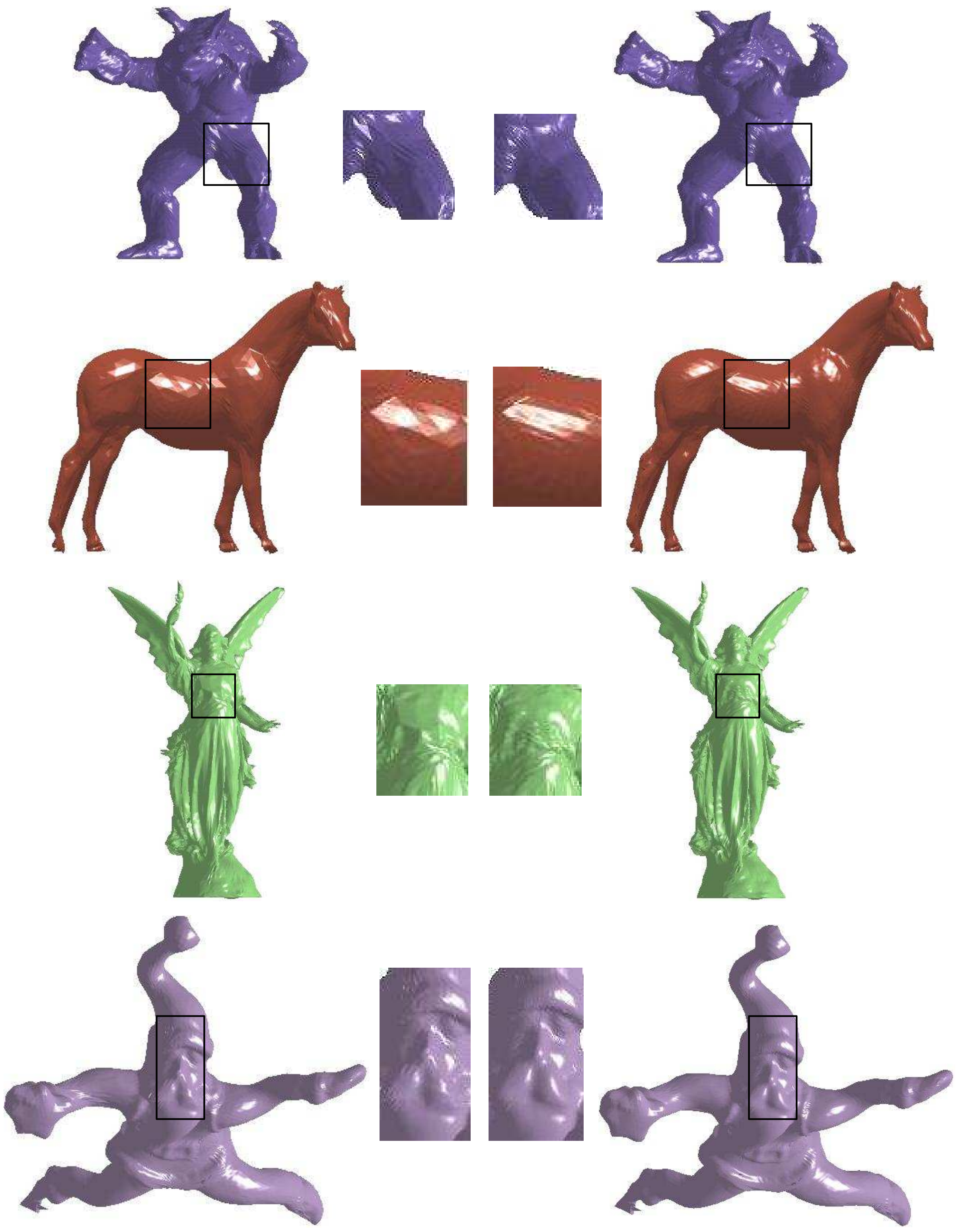


Fig. 12. From top to bottom: Armadillo, Horse, Lucy, Santa. Leftmost: without considering illumination. Rightmost: with the consideration of illumination. Middle: corresponding zoom-in parts. For all the models, we set $\lambda = 0.5$ and the bandwidth to be 20% vertices.

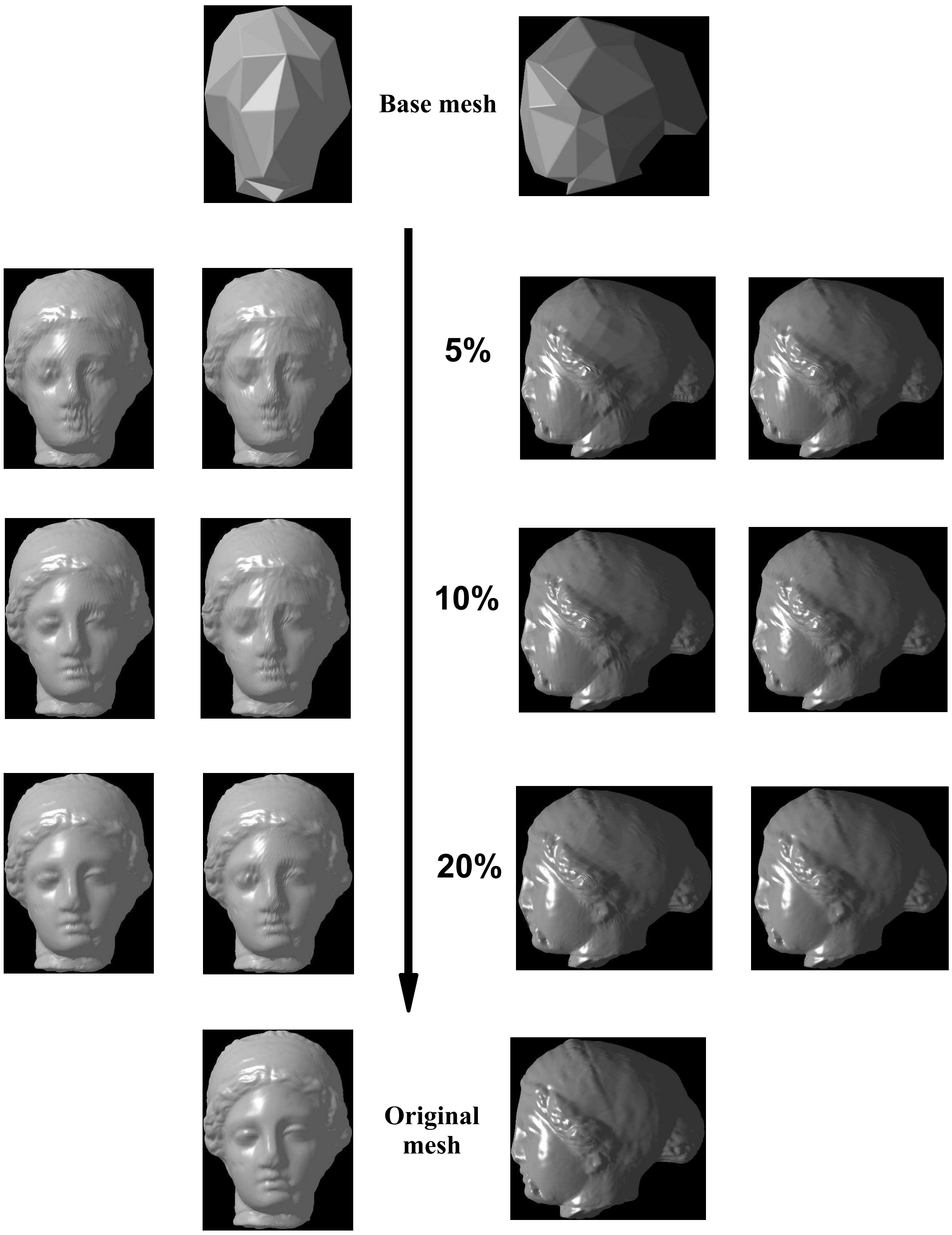


Fig. 13. The progressive transmission performance with $I_a = I_d = I_s = 1.0, k_a = 0.2, k_d = 0.8, k_s = 1.0$ and $m = 10$. The four columns from left to right: considering illumination (front-view), not considering illumination (front-view), not considering illumination (side-view), and considering illumination (side-view).

$5\% \sim 20\%$ of the total number of vertices. It can be seen that the performance gain of our proposed system is more prominent at lower total available bandwidth.

VI. Conclusion

We have made two main contributions in this research. The first contribution is that we have derived the weighting function shown in Eq. (6), which can accurately reflect the importance of wavelets at different refinement levels. Based on the derived weight, we proposed to scale up the vector wavelets generated in remeshing so that the inherent importance of wavelets can be truly utilized by the zerotree-like compression algorithms. Experimental results have demonstrated that our proposed mesh coder significantly outperforms the existing one. Our second major contribution is the illumination distortion model derived in Eq. (20). Combined with the geometry distortion model in Eq. (19), we obtained an overall distortion model that considers both illumination and view dependencies. This is a significant contribution since to the best of our knowledge none of the existing progressive mesh transmission systems has considered the illumination effects. Based on our derived distortion model, given the viewing and lighting parameters, we are able to optimally allocate bits among different segments in real time. Simulation results show that significant quality improvement can be achieved by taking into account the illumination effects in addition to the view dependency.

R-D analysis has been extensively used in the areas of image/video coding and transmission and various R-D models have been developed. Although there are a few studies on R-D analysis for reconstructing 3D meshes [9] or 3D scenes [19], the overall research is falling behind. This is mainly because of the complexity of 3D models and the unpopularity of networked 3D applications in the past. Our research only sheds some lights on the R-D analysis for 3D mesh coding and transmission. There are various ways to extend our work such as developing more accurate R-D models, considering robustness issues and extending to dynamic 3D meshes. The main focus for the research in this direction is not about applying the techniques for image and video to meshes in a straightforward manner but about designing methods that consider the unique features of 3D mesh coding and transmission.

Appendix

Here, we derive the expression of $(N \cdot L)(N \cdot V)$ used in Eq. (17), where N is an arbitrary vector and L and V are two arbitrary unit vectors. Suppose the geometry relationship among N, L, V is the same as that among N_i, L, V in Fig. 6.

Since L and V are unit vectors, $(N \cdot L)(N \cdot V)$ can be written as $A = (|N|\cos\alpha)(|N|\cos\beta)$, where α and β are the angles between the corresponding two vectors. Then, we can have

$$\begin{aligned} A &= |N|^2\frac{\cos(\alpha+\beta)+\cos(\alpha-\beta)}{2} \\ &= |N|^2[\frac{(L\cdot V)}{2}+\cos(\frac{\alpha-\beta}{2})^2-\frac{1}{2}] \\ &= |N|^2[\frac{(L\cdot V)}{2}+\frac{(N\cdot H)^2}{|N|^2}-\frac{1}{2}] \\ &= (N\cdot H)^2-|N|^2\frac{1-(L\cdot V)}{2} \end{aligned} \tag{22}$$

where $H = \frac{L+V}{|L+V|}$. For the case that the geometry relationship among N, L, V is different from that illustrated in Fig. 6, we can perform similar derivation and the final expression is the same.

Acknowledgment

The authors would like to thank Dr. Jae-Young Sim and Prof. Chang-Su Kim for sharing their source codes for the work in [9]. Some 3D models are obtained from the Stanford 3D Scanning Repository. The authors would also like to thank anonymous reviewers for their valuable comments and suggestions.

References

[1] J. Zhang, J. Cai, W. Guan, and J. Zheng, “Re-examination of applying wavelet based progressive image coder for 3D semi-regular mesh compression,” in *IEEE ICME*, 2008, pp. 617–620.

[2] W. Guan, J. Zhang, J. Cai, and J. Zheng, “Illumination and view dependent progressive transmission of semiregular meshes with subdivision connectivity,” in *ACM Graphite*, 2007, pp. 219–226.

[3] H. Hoppe, “Efficient implementation of progressive meshes,” *Computers & Graphics*, vol. 22, pp. 327–336, 1998.

[4] G. Taubin, A. Gueziec, W. Horn, and F. Lazarus, “Progressive forest split compression,” in *ACM SIGGRAPH*, 1998, pp. 19–25.

[5] R. Pajarola and J. Rossignac, “Compressed progressive meshes,” *IEEE Trans. Vis. Comput. Graph.*, vol. 6, no. 1, pp. 79–93, 2000.

[6] J. Li and C.-C. Kuo, “Progressive coding of 3-d graphic models,” *Proc. IEEE*, vol. 86, no. 6, pp. 1052–1063, Jun. 1998.

[7] A. Khodakovsky, P. Schroder, and W. Sweldens, “Progressive geometry compression,” in *ACM SIGGRAPH*, 2000, pp. 271–278.

[8] A. W. F. Lee, W. Sweldens, P. Schroder, L. Cowsar, and D. Dobkin, “MAPS: Multiresolution adaptive parameterization of surfaces,” in *ACM SIGGRAPH*, 1998, pp. 95–104.

[9] J.-Y. Sim, C.-S. Kim, C.-C. Kuo, and S.-U. Lee, “Rate-distortion optimized compression and view-dependent transmission of 3D normal meshes,” *IEEE Trans. Circuit and Syst. Video Technol*, vol. 15, no. 7, pp. 854–868, Jul. 2005.

[10] H. Hoppe, “View-dependent refinement of progressive meshes,” in *ACM SIGGRAPH*, Aug. 1997, pp. 189–198.

[11] S. Yang, C.-S. Kim, and C.-C. Kuo, “A progressive view-dependent technique for interactive 3D mesh transmission,” *IEEE Trans. Circuit and Syst. Video Technol*, vol. 14, no. 11, pp. 1249–1264, Nov. 2005.

[12] W. Guan, J. Cai, J. Zheng, and C. W. Chen, “Segmentation based view-dependent 3D graphics model transmission,” *IEEE Trans. Multimedia*, vol. 10, no. 5, pp. 724–734, Aug. 2008.

[13] J. M. Shapiro, “Embedded image coding using zerotrees of wavelet coefficients,” *IEEE Trans. Signal Processing*, vol. 41, no. 12, pp. 3445 – 3462, Dec. 1993.

[14] A. Said and W. A. Pearlman, “A new fast and efficient image codec based on set partitioning in hierarchical trees,” *IEEE Trans. Circuit and Syst. Video Technol*, vol. 6, no. 3, pp. 243–250, June 1996.

[15] D. Luebke and C. Erikson, “View-dependent simplification of arbitrary polygonal environments,” in *ACM SIGGRAPH*, 1997, pp. 199–208.

[16] M. Reddy, *Perceptually Modulated Level of Detail for Virtual Environments*, Ph.D. Thesis, University of Edinburgh, 1997.

[17] ——, “Perceptually optimized 3D graphics,” *IEEE Computer Graphics and Applications*, vol. 21, no. 5, pp. 68–75, 2001.

[18] J. F. Blinn, “Models of light reflection for computer synthesized pictures,” in *ACM SIGGRAPH*, 1977, pp. 192–198.

[19] E. Imre, A. A. Alatan, and U. Gudukbay, “Rate-distortion efficient piecewise planar 3-D scene representation from 2-D images,” *IEEE Transactions on Image Processing*, vol. 18, no. 3, pp. 483–494, March 2009.