Mesh Segmentation Schemes for Error Resilient Coding of 3-D Graphic Models

Zhidong Yan, *Member, IEEE*, Sunil Kumar, *Senior Member, IEEE*, and C.-C. Jay Kuo, *Fellow, IEEE*

***Abstract*—Most existing coding techniques for three-dimensional (3-D) graphic models focus on coding efficiency. Due to irregular structure of the 3-D mesh and the use of variable length entropy codes, channel errors often propagate in the coded bitstream and severely distort the decoded model. By segmenting a 3-D graphic model (or its connected components) into small pieces, the impact of channel errors is confined to directly corrupted pieces rather than the whole mesh. The mesh segmentation schemes should be compatible with the underlying encoding techniques to achieve the low computational and coding overhead. In this research, we examine four mesh segmentation schemes, i.e., multiseed traversal, threshold traversal, morphing-based volume splitting, and content-based segmentation, and apply them to the context of error resilient mesh coding based on the constructive traversal coding technique. The advantages and shortcomings of each segmentation method are discussed. These schemes segment a mesh into pieces according to the target piece size that is determined according to the channel error rate.**

***Index Terms*—Constructive traversal, error resiliency, mesh partitioning, mesh segmentation, 3-D graphic model, 3-D mesh.**

I. Introduction

Models of three-dimensional (3-D) objects generally consist of triangular and/or polygonal meshes. Since the resulting meshes usually consist of tens of thousands of vertices and triangles, it is necessary to compress them with tolerable distortion while maximizing the degree of data reduction. A 3-D graphic model is represented by *topological and attribute* data. Topological data specify the connectivity information among vertices (e.g., the adjacency of vertices, edges and faces) while attribute data describe the position, surface normal, color and other application-specific information of each vertex. We will concentrate on manifold polygonal models with only the position attribute information in this paper.

Early research on 3-D mesh processing focused on simplification of graphic models [1], [2]. Recent work has emphasized more on the compression of graphic models [2]–[4]. As the third-generation (3G) telecom system and the wireless local area networks (LANs) become more mature, wireless transmission of 3-D graphic models will be in place in the near future. Errors often occur in wireless channels because of fading and interference. Due to the irregular structure of the 3-D mesh and the use of variable length entropy codes, the resulting errors often propagate to a large area. This affects the decoding of subsequent connectivity information and leads to severe distortion of the decoded mesh structure. Traditional mesh coding techniques (e.g., [3], [4]) are very sensitive to channel errors since the graphic model is encoded as a whole [5].

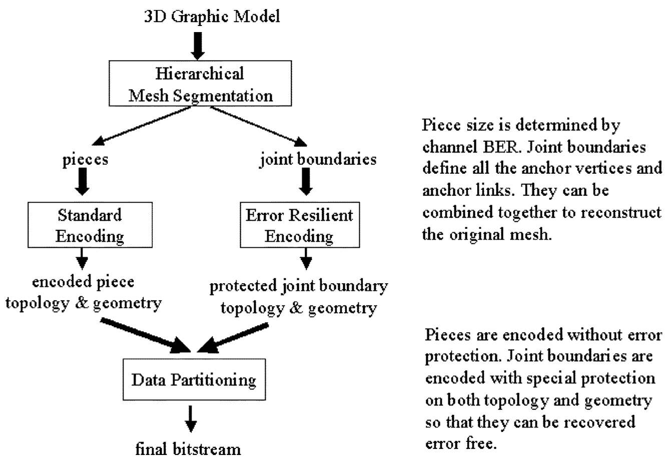
3D Graphic Model
Hierarchical Mesh Segmentation
pieces
joint boundaries
Standard Encoding
Error Resilient Encoding
encoded piece topology & geometry
protected joint boundary topology & geometry
Data Partitioning
final bitstream
Piece size is determined by channel BER. Joint boundaries define all the anchor vertices and anchor links. They can be combined together to reconstruct the original mesh.
Pieces are encoded without error protection. Joint boundaries are encoded with special protection on both topology and geometry so that they can be recovered error free.

Fig. 1. Block diagram of the encoder for a 3-D graphic model.

Little research has been done in the error resilient coding of 3-D graphic models before, e.g., [5], [6]. Based on Li and Kuo's constructive traversal coding technique in [4], we proposed an efficient error resilient coding technique in [5] which can achieve high error resiliency as well as incremental rendering at a reasonable bit rate overhead. As shown in Fig. 1, the mesh structure of each connected component of a 3-D graphic model is segmented into a set of smaller pieces, according to the target channel bit-error rate (BER). The relationship between two neighboring pieces of a connected component is represented by the "joint boundary," which contains information of common vertices and links. The joint boundary and the piece data of each connected component are separately encoded. Since the joint boundary information is necessary to stitch pieces back to obtain the original mesh, it is protected by using Bose–Chaudhuri–Hocquenghem (BCH) error correcting code. At the decoder, the joint boundary information of a connected component is first extracted. After that, pieces are decoded to reconstruct the connected components. Since the decoding procedure can start as soon as coded data of the first connected component is received, the model can be incrementally rendered.

Mesh segmentation plays an important role in the error-resilient coding scheme presented in [5]. Several mesh segmentation schemes exist for efficient handling or parallel processing to address issues such as scientific computing, fast graphics rendering, and so on [7]–[9]. Their results are not necessarily suitable for error-resilient coding. In this research, we propose four mesh

Manuscript received June 25, 2002; revised July 17, 2003. This work was supported in part by the Integrated Media Systems Center, an NSF Engineering Research Center, and in part by the Annenberg Center for Communication, University of Southern California, Los Angeles, CA.

Z. Yan is with J2 Global Communications, Hollywood, CA 90028 USA (e-mail: zyan@usc.edu).

S. Kumar is with the Department of Electrical and Computer Engineering, Clarkson University, Potsdam, NY 13699 USA (e-mail: skumar@clarkson.edu).

C.-C. J. Kuo is with the Integrated Media Systems Center and the Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089-2564 USA (e-mail: cckuo@sipi.usc.edu).

Digital Object Identifier 10.1109/TCSVT.2004.837023

segmentation schemes that are specifically designed for error resilient coding of 3-D graphic models, based on the constructive mesh traversal idea as used in [4]. The constructive traversal scheme starts from a chosen seed (i.e., vertex) and traverses vertices and polygons with a uniform depth approach, where the closest vertices and polygons are traversed first, followed by the next closest ones, until the traversal process is completed. The coded bitstream size increases with the traversal depth.

The features of the proposed mesh segmentation schemes include the following.

- Coding efficiency for a given piece size (in terms of number of vertices) is improved by minimizing the depths from the seed. Other mesh segmentation schemes may not perform well under this constraint.
- Relatively uniform size pieces are obtained, as determined by the channel BER. For a BER of 10^{-3}, the target piece size can be as small as 32 vertices. Applications addressed by other mesh segmentation schemes usually do not demand graphic models to be divided into pieces of such a small size.
- The number of vertices in each joint boundary is kept very small, to ensure very low overhead in the coding of the joint boundary.

Even though Li and Kuo's encoding scheme [4] can work with other mesh segmentation schemes as presented in [7]–[9], we have observed that coding efficiency decreases considerably when the segmentation method results in nonuniform pieces, containing small isolated pieces or long boundaries.

This paper is organized as follows. Four mesh segmentation schemes are described in Section II. Experimental results and conclusion are given in Sections III and IV, respectively.

II. Proposed Mesh Segmentation Schemes

We first determine the piece size for a given channel BER, followed by mesh segmentation algorithms by considering coding efficiency and error resiliency.

A. Piece Size Determination

The topology structure of 3-D graphic models usually contain many connected components of different size, in terms of vertices and polygons. Each connected component can be treated as an independent unit of the original graphic model. Large connected components should be segmented into pieces for better error resiliency. We determine the piece size (in terms of the number of vertices) based on the target BER.

$$\mathrm{piece\_size} \approx \left(\frac{1}{\mathrm{BER} \cdot \mathrm{bits\_per\_vertex} \cdot \mathrm{piece\_group}}\right). \quad (1)$$

Here, $\mathrm{bits\_per\_vertex}$ represents the number of bits required to code a vertex. The coding of topology data requires an average of 2 b/vertex in [4]. Therefore, statistically, there would be 1 bit error for every 500 vertices of a coded mesh, at a BER of 10^{-3}. The $\mathrm{piece\_group}$ is the number of pieces (i.e. ≥ 10 pieces in our scheme) whose coded topology data will have only 1 bit error in total. The minimum piece size is set to 32 to avoid high bit rate overhead. For a piece size of 50, there would thus be an average of 1 bit error in 10 pieces. Since each piece consists of complete polygons, the actual piece size will be slightly different. Now, we can determine the target number of pieces (N) as a ratio of number of vertices in the mesh component to the $\mathrm{piece\_size}$.

B. Mesh Segmentation

Since no two pieces share the same polygon, the union of polygon sets of all pieces form the complete polygon set of the original model. We represent the original 3-D polygonal mesh M by using set V of all vertices (geometry position and vertex index) and set P of all polygons (indices of vertices) of the mesh. We segment mesh M into N pieces $M_1, M_2, \ldots, M_N$, where M_i is characterized by V_i and P_i.

Four mesh segmentation schemes are discussed below: multiseed traversal, threshold traversal, morphing-based volume splitting, and the content-based segmentation method.

1) Multiseed Traversal Algorithm: The basic idea of this algorithm is similar to that of simultaneously burning fires from multiple starting seeds, which would grow uniformly toward their respective neighborhood until all polygons and vertices of the whole mesh are covered. Each traversed region is called a piece.

The algorithm can be described by the following steps.

1) Initialize N polygon sets P_i and N vertex sets V_i, where $i = 1, \ldots, N$. All sets are empty in the beginning and implemented as a FIFO queue structure.
2) Choose N starting seeds by using the seed selection algorithm described later in this section. Push each seed in its corresponding vertex set V_i, and mark the seed as unvisited. Set the size of this vertex set to 1.
3) Get the first unvisited seed in vertex set V_i that has the smallest size. If there are multiple vertex sets that satisfy the smallest size requirement, choose the one with the smallest index.
4) Traverse the polygons associated with the chosen seed in a counterclockwise order, and find the first unvisited polygon. Push all its unvisited vertices into the corresponding vertex set V_i exclusively. Increase the size of V_i by the number of vertices pushed in. Mark the polygon as visited, and push it into the corresponding polygon set P_i. Otherwise, mark the seed as visited, if no unvisied polygon can be found.
5) Repeat Steps 3–4 until no unvisited vertex and polygon can be located.

Seed selection: To obtain pieces of uniform size, starting seeds should be carefully selected in the mesh structure, by using the **far distance** approach, as explained below.

1) Calculate the geometrical center of all the vertices of the connected component, and choose the vertex with the largest Euclidean distance from the center point as the "first seed."
2) After choosing $i-1$ seeds, the ith seed is chosen as the vertex with the largest distance from the set of $i-1$ seeds that satisfies max (min (distance(vertex k, seed j))). Here, seed j is a vertex in the set of $i-1$ already chosen seeds, and vertex k is a vertex from the remaining mesh component vertices.
3) Repeat the above step until all N seeds are selected.

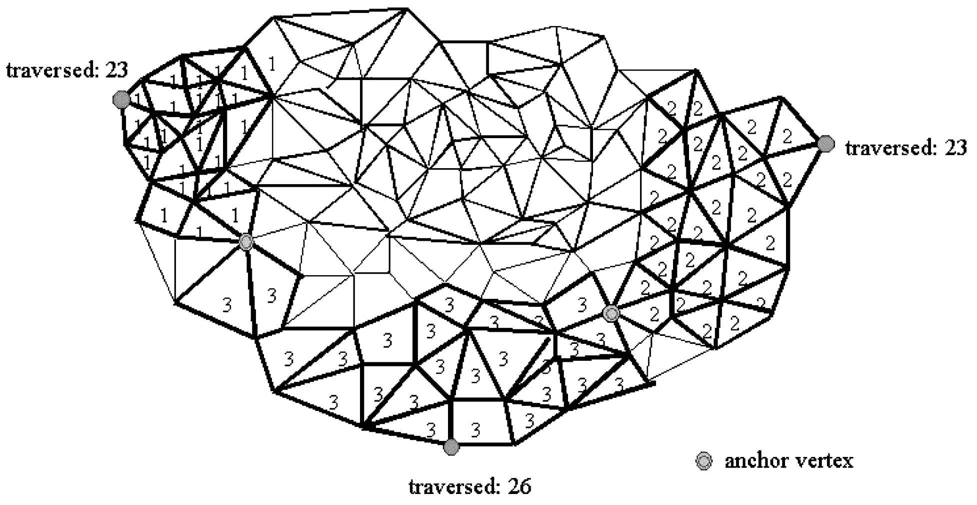

Fig. 2. Illustration of the middle stage in the multiseed traversal scheme by using three starting seeds. Generation of two anchor-vertices is shown here.

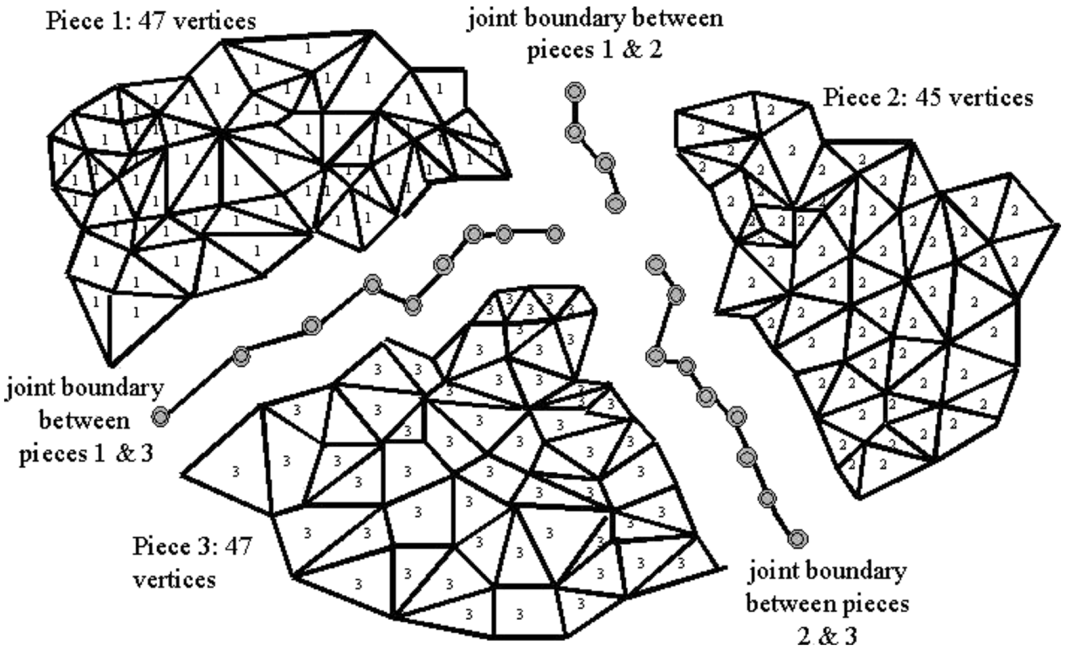
Piece 1: 47 vertices
joint boundary between pieces 1 & 2
Piece 2: 45 vertices
joint boundary between pieces 1 & 3
Piece 3: 47 vertices
joint boundary between pieces 2 & 3

Fig. 3. Illustration of the final stage in multiseed traversal scheme by using three starting seeds, where the three joint boundaries are shown, after joint boundary smoothing.

In Fig. 2, we show a mesh of 119 vertices, with a target piece size of 40 vertices. By using the far distance approach, we select three seeds denoted by "1," "2," "3." Starting from each seed, we traverse polygons as described above, and assign their corresponding seed number to each of them. For each seed, 23 to 26 traversed vertices associated with traversed polygons are shown. It should be pointed out that the boundaries of Pieces 1 and 3 (and also Pieces 2 and 3) start to merge at a common vertex, which we call "anchor vertex." As the traversal progresses, more anchor vertices will be generated. Fig. 3 shows the three resulting pieces and their corresponding joint-boundaries, along with anchor vertices.

Two adjacent pieces share a number of vertices and links on their *joint boundary*. We do not allow any polygon in the joint boundary. Care has also been taken in mesh segmentation to make sure that the joint boundary is composed by vertices which are linked one by one to form a 3-D curve. The joint boundary information is critical as it is used to help stitch different pieces of a connected component together, when one or more pieces are corrupted by channel errors. Since, the coding of joint boundary information contributes to the bit rate overhead, we described a simple *smoothing scheme* in [5], which reduces the number of vertices of the joint boundary while keeping the mesh segmentation result nearly the same.

2) Threshold Traversal Algorithm: Similar to the multiseed traversal algorithm, the threshold traversal algorithm also chooses the starting seed, burns the fire and lets it grow. However, the threshold traversal algorithm selects only one seed at a time, and traverses its neighboring polygons and vertices. The traversal process stops, when the required number of vertices, determined by a threshold (i.e., the target piece size), have been traversed. Traversed polygons and vertices are extracted from the model and assigned to the current piece. A new seed is then selected from the remaining mesh and the traversal process is

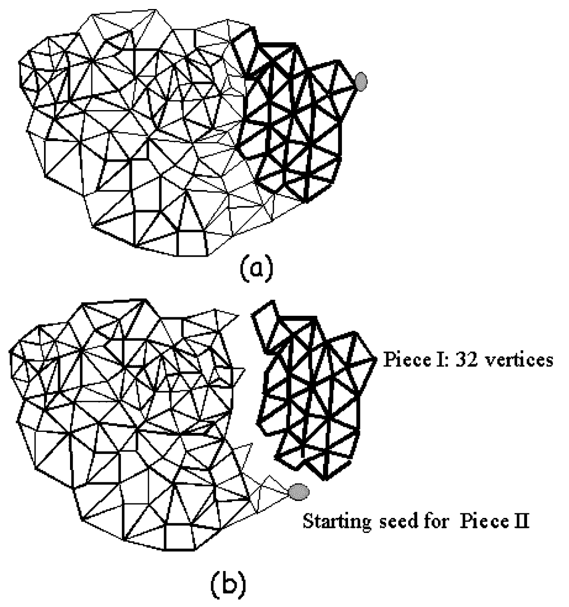

Fig. 4. Illustration of the threshold traversal scheme.

repeated. This is how the whole mesh is traversed and pieces are obtained. The threshold traversal algorithm works as follows.

1) Denote the total number of vertices in the connected component by N_v. If the desired number of pieces is N, the vertex threshold is $V_s = N_v/N$.
2) Initialize $N-1$ polygon sets P_i and a vertex set V, where $i = 1, \ldots, N-1$. All sets are empty in the beginning and implemented as a FIFO queue structure.
3) Choose one starting seed from the model and push it into V, mark the seed as unvisited. Set the size of V to 1. Set the index i starting from 1.
4) Get the first unvisited seed in V, traverse the polygons associated with it in a counterclockwise order, and find the first unvisited polygon. Push all its unvisited vertices into the vertex set V exclusively. Increase the size of V by the number of vertices pushed in. Mark the polygon as visited, and push it into the corresponding polygon set P_i. Otherwise, mark the seed as visited, if no unvisited polygon is found.
5) Repeat Steps 3–4 until the size of V exceeds threshold V_s or no unvisited vertex and polygon can be located.
6) Delete polygon set P_i from the model. Empty set V, and increase i by 1.
7) Repeat Steps 3–6 until i reaches $N-1$. Put all remaining polygons in a new polygon set P_N.

The resulting N polygon sets $P_1, P_2, \ldots, P_N$ and their associated vertices form N pieces. Fig. 4(a) shows the starting seed for Piece I and its 32 traversed vertices. With Piece I having been extracted, the starting seed for Piece II is shown in Fig. 4(b).

3) Morphing-Based Volume Splitting: It is relatively easy to divide a regular 3-D object, such as a ball or candy bar, into geometrically equal sized pieces. The morphing-based volume splitting scheme carries out segmentation by first mapping the mesh to a regular object (i.e., morphing) and then partitioning it into pieces geometrically (i.e., volume splitting), as discussed below.

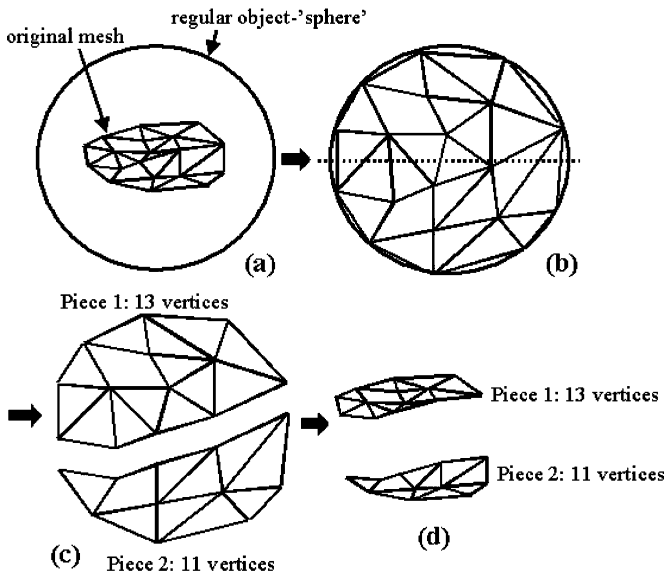

Fig. 5. Illustration of the morphing-based volume splitting scheme.

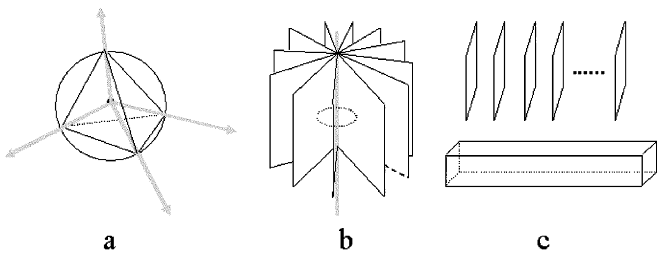

Fig. 6. Volume splitting applied to three predefined structures. (a) Sphere. (b) Cylinder. (c) Long bar.

Morphing: Many meshes can usually be mapped to a regular 3-D object, such as a sphere, cylinder or long bar, by using 3-D morphing. We explain the scheme with the help of an example where we map our mesh to a sphere that can hold all its vertices and polygons. First, we place the mesh inside a sphere as shown in Fig. 5(a). Then, mesh vertices are moved outward to the sphere surface. The morphing is complete when all vertices are uniformly distributed on the sphere surface in the geometry sense as shown in Fig. 5(b). Connectivity between vertices (i.e., topology) remains untouched in this procedure. Note that not all graphic models can be morphed to a sphere, cylinder or long bar, and a more complicated 3-D object may be needed. Since, we segment each connected component separately, most of them can be mapped to a regular object. However, this technique of morphing/moving vertices can also be generalized to use implicit surfaces that are more complex than regular 3-D objects, as discussed in [10] and [11].

Volume splitting: Since all vertices are uniformly distributed on the sphere surface, cutting the sphere into geometrically equal sized regions will result in segmenting the mesh into topologically equal sized pieces. Fig. 5(c) shows the volume splitting of an object into two pieces. After volume splitting, mesh vertices are moved back to their original positions as shown in Fig. 5(d). Again, connectivity between vertices remains unchanged.

In Fig. 6, we show the application of this scheme to three predefined structures, i.e., sphere, cylinder and long bar. In

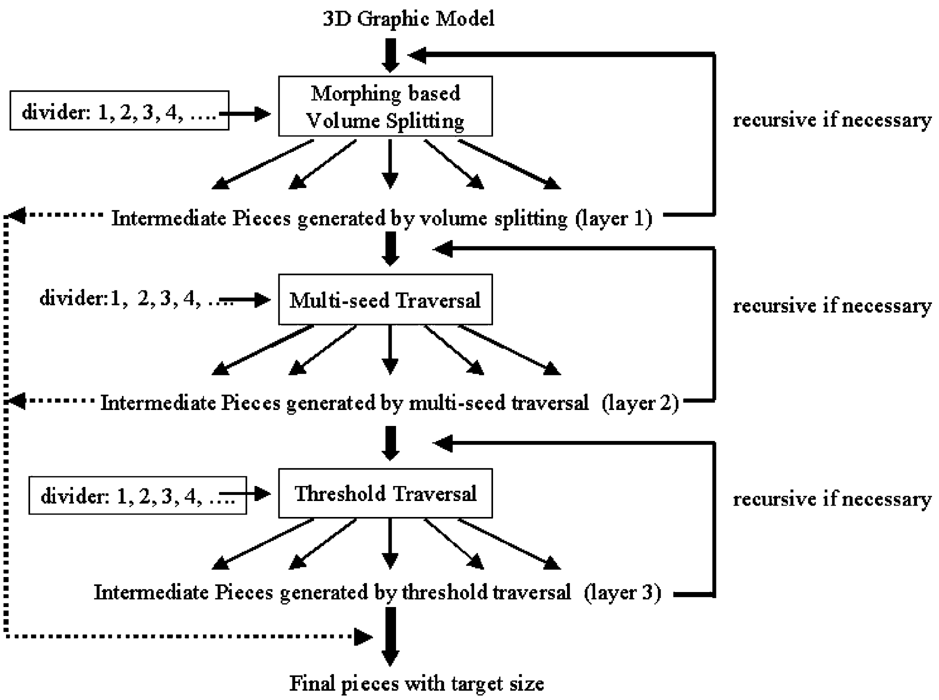
3D Graphic Model
divider: 1, 2, 3, 4, ...
Morphing based Volume Splitting
recursive if necessary
Intermediate Pieces generated by volume splitting (layer 1)
divider:1, 2, 3, 4, ...
Multi-seed Traversal
recursive if necessary
Intermediate Pieces generated by multi-seed traversal (layer 2)
divider: 1, 2, 3, 4, ...
Threshold Traversal
recursive if necessary
Intermediate Pieces generated by threshold traversal (layer 3)
Final pieces with target size

Fig. 7. Three-layer hierarchical mesh segmentation scheme.

Fig. 6(a), the sphere is partitioned into four uniform regions. For this, a group of half-infinite lines are built, starting from the center of the sphere. Any two adjacent lines form a partial plane that cut into the sphere. The directions of these lines are calculated to ensure that they divide the 360° 3-D solid angle into surfaces of the same size. In Fig. 6(b), the cylinder is partitioned into uniform regions. Basically, starting from the central vertical line of the cylinder, a group of half-infinite planes are built, each of which cuts into the cylinder through the central line. The directions of the planes are calculated to ensure uniform partitioning. In Fig. 6(c), the long bar is partitioned into uniform regions. Here, a group of infinite planes is built, where each plane cuts into the long bar. The distance between two consecutive planes is set to ensure uniform splitting.

4) Content-Based Segmentation: In many applications, it may be useful if a model can initially be segmented in a few semantically meaningful parts. Each part can then be further divided in smaller pieces by using one or more segmentation schemes. It is possible to choose smaller sizes for semantically important pieces and provide them higher protection against channel errors by using "unequal error protection." Since our error resilient coding scheme supports incremental rendering, pieces belonging to semantically more important regions can be transmitted first. In this case, a few corrupted pieces will not introduce much distortion in the reconstructed model, and some scheme(s) could be used to conceal them at the decoder.

Since it is difficult to train a computer for extracting the semantic meaning of a specific graphic model, we extract pieces by using the user-input about the approximate location of piece boundary, which includes the following information.

1) Several vertices on the piece boundary are selected. For example, to segment the head from the "dinosaur" model, several vertices around the neck should be selected.
2) A closed path is formed passing through these vertices to divide the mesh. For this, the order of vertex selection is important. Let us denote n vertices by x_1, x_2 up to x_n, according to the order in which they are selected. The closed path is constructed by forming the path x_1 to x_2, x_2 to $x_3, \ldots, x_{n-1}$ to x_n, and x_n to x_1, in that order. The path from x_i to x_{i+1}, $i = 1, 2, \ldots, n-1$, should satisfy the shortest path requirement, i.e., traversing the minimum number of vertices between them.

Three pieces of the "dinosaur" model are shown in Fig. 11, where the "head" piece is constructed by selecting five vertices around the neck, and the "middle body" piece is constructed by selecting 12 vertices between this and the "tail" piece.

5) Discussion: We tested our proposed segmentation algorithms on MPEG-4 test models. The multiseed traversal algorithm achieves better results when the mesh is divided in two to four pieces. The threshold traversal algorithm is more efficient when the model is divided in six or more pieces. This may be due to the fact that selection of seeds in the multiseed traversal algorithm becomes more critical when the number of pieces is large. It is quite likely that two seeds are selected much closer than other seeds, which would result in nonuniform pieces. The morphing-based volume splitting scheme is computationally more complex $O(N^2)$, because it requires mapping and redistribution of vertices according to the reference object, followed by volume splitting and reverse mapping. Here N refers to the number of vertices in the mesh. This scheme should be primarily used to initially partition the mesh into a small number of pieces. Furthermore, the multiseed and threshold traversal algorithms are more efficient, with computational complexity $O(N)$. The traversal information is recorded in these two segmentation schemes that can later be used in the piece encoding process. However, the system complexity also depends on the target BERs, piece size, layered hierarchies, etc. Nevertheless,

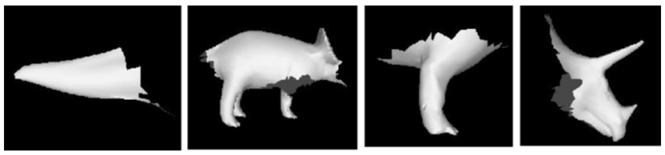
Fig. 8. Segmentation of the dinosaur model by multiseed traversal algorithm, with the "far distance" seed selection approach. Only four pieces are shown for the purpose of illustration.

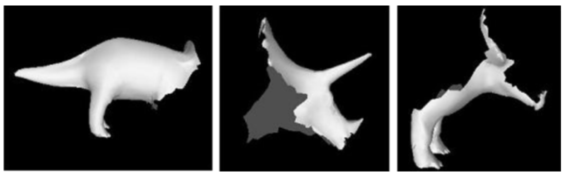
Fig. 9. Extracted pieces of the "dinosaur" model by threshold traversal algorithm.

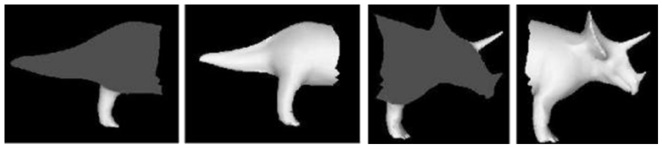
Fig. 10. Segmentation of the "dinosaur" model into four pieces by line-based volume splitting.

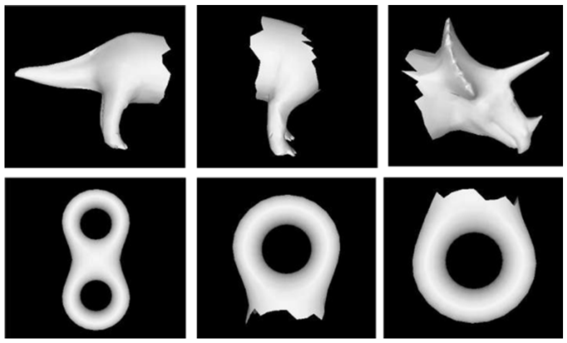
Fig. 11. Content-based segmentation: "dinosaur" model segmented into three pieces (upper row); "eight" model segmented into two pieces (bottom row).

we found in our research that the most computational complexity comes from the encoding and assembling of the segmented pieces.

To achieve the best performance, we should use a segmentation scheme repetitively or in a combination of other scheme(s), to exploit the strength of each scheme. A *hierarchical segmentation* scheme using the three-layer structure is shown in Fig. 7, where the value of divider indicates the number of pieces to be obtained by the segmentation process at that stage. Each segmentation scheme can be applied repetitively. The segmented piece from layer 1 becomes the input data to layer 2 for further segmentation. For example, the "spock" graphic model has 16 386 vertices. Target piece size of 32 vertices requires it to be divided in about 500 pieces. We can first use the morphing-based volume splitting algorithm twice to divide the model in four pieces each time. Then, the multiseed traversal algorithm can be used to further divide each piece into four pieces, leading to a total of 64 pieces. Finally, the threshold-traversal algorithm can be applied to further divide each of the 64 pieces into six to eight pieces.

III. Experimental Results

Figs. 8–10 show three to four pieces extracted from the "dinosaur" graphic model by using the *multiseed traversal*, *threshold traversal*, and *morphing-based volume splitting* schemes. These schemes give pieces of reasonably uniform size, with no isolated or broken pieces. We show in Fig. 11 the *content-based segmentation* of the "dinosaur" and "eight" models. Each of these pieces can be morphed to three structures (i.e., sphere, cylinder, and long bar) and thus morphing-based volume splitting can be further applied.

The multiseed and threshold traversal algorithms allow pieces to grow uniformly around the seeds. As a result, we get small joint boundary with an average of seven and ten joint boundary vertices for a piece size of 32 and 64, respectively, for the dinosaur model. The pieces obtained by using these scheme can be simply or multiply connected, depending on the original mesh structure of the connected component. The morphing-based volume splitting scheme however does not guarantee small boundaries.

The proposed segmentation schemes achieve relatively low bit rate overhead and high success rate (i.e., error resiliency) at different BERs. The bit rate overhead normally increases as the size of the graphic model becomes smaller and/or the BER increases. In [5], we discussed the result of the error resilient coding scheme applied to graphic models at different BERs. If the topology of one or more pieces cannot be decoded successfully, there would be some holes in the reconstructed model. If the topology of all pieces is decoded correctly while the geometry data of a piece is not correctly decoded, the corresponding piece is decoded with some distortions.

IV. Conclusion and Future Work

Existing mesh segmentation schemes may not be suitable for the constructive traversal-based error resilient encoding scheme, since coding efficiency may decrease considerably when one obtains nonuniform, very small and isolated pieces, with long boundaries. In this research, four mesh segmentation schemes were studied to obtain smaller and uniform sized pieces, for error resilient encoding based on the constructive mesh traversal technique. The piece size is determined according to the channel BER. The proposed mesh segmentation schemes ensure a small number of vertices on the joint boundary between two neighboring pieces of a connected component, which is vital for high error resiliency and low bit rate overhead.

The proposed segmentation algorithms were primarily developed for manifold polygonal models. We observed that their performance is not well defined for more generic models including the "unclean" ones. It should be interesting to modify them for more generic models. It may also be challenging but worthwhile to study a robust automatic method to hierarchically apply the different proposed algorithms in an "optimal" way for an arbitrary mesh.

References

[1] D. P. Luebke, "A developer's survey of polygonal simplification algorithms," *IEEE Comput. Graph. Appl.*, pp. 24–35, May–Jun. 2001.

[2] H. Hoppe, "Progressive meshes," in *Proc. Comput. Graph. Annu. Conf. Series, ACM SIGGRAPH*, Aug. 1996, pp. 99–108.

[3] G. Taubin and J. Rossignac, "Geometry compression through topological surgery," *ACM Trans. Graphics*, vol. 17, pp. 84–115, 1998.

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

[5] Z. Yan, S. Kumar, and C.-C. J. Kuo, "Error resilient coding of 3-D graphic models via adaptive mesh segmentation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 7, pp. 860–873, Jul. 2001.

[6] M.-J. Han, M.-S. Song, S.-J. Kim, and E. S. Jang, "Results of M5 core-experiments," ISO/IEC JTC1/SC29/WG11, MPEG98/M4516, Mar. 1999.

[7] G. Karypis and V. Kumar. (1998) MeTis: A software package for partitioning unstructured graphs, partitioning meshes, and computing fill-reducing orderings of sparse matrices. Univ. Minnesota. [Online]. Available: http://www-users.cs.umn.edu/~karypis/merits/metis.html

[8] J. R. Gilbert, G. L. Miller, and S.-H. Teng, "Geometric mesh partitioning: implementation and experiments," *SIAM J. Sci. Comput.*, vol. 19, pp. 2091–2110, 1998.

[9] F. Guerinoni. (1995) Mesh partitioning techniques and new observations for 3-regular graphs. IRISA, France. [Online]. Available: http://www.irisa.fr/bibli/publi/pi/1995/947/947.html

[10] L. H. Figueiredo, J. Gomes, D. Terzopoulos, and L. Velho, "Physically-based methods for polygonization of implicit surfaces," in *Proc. Graph. Interface*, 1992, pp. 250–257.

[11] R. Zonenschein, J. Gomes, L. Velho, and L. H. Figueiredo, "Controlling texture mapping onto implicit surfaces with particle systems," in *Proc. Implicit Surfaces*, 1998, pp. 131–138.