Robust Compression and Reconstruction of 3-D Mesh Data

Sung-Bum Park *, Chang-Su Kim **, and Sang-Uk Lee *

*Signal Processing Lab., School of Electrical Engineering and Computer Science, Seoul National University, Seoul 151-741, Korea

**Department of Electronics and Computer Engineering, Korea University, Seoul, Korea

ABSTRACT

An algorithm for robust transmission of compressed 3-D mesh data is proposed in this work. In the encoder, we partition a 3-D mesh adaptively according to the surface complexity, and then encode each partition separately to reduce the error propagation effect. To encode joint boundaries compactly, we propose a boundary edge collapse rule, which also enables the decoder to zip partitions seamlessly. In the decoder, an error concealment scheme is employed to improve the visual quality of corrupted partitions. The concealment algorithm utilizes the information in neighboring partitions and reconstructs the lost surface based on the semi-regular connectivity reconstruction and the polynomial interpolation. Simulation results demonstrate that the proposed algorithm provides a good rendering quality even in severe error conditions.

Keywords: 3-D mesh compression, error-resilient coding, error concealment, surface reconstruction.

1. INTRODUCTION

Compression of 3-D mesh data has been an important issue in computer graphics, since the size of a mesh is often too large to store and transmit efficiently. Many algorithms have been proposed to encode mesh data at a single resolution1 or in a progressive way.2–6 However, compressed bit streams of both single-resolution and progressive coders are vulnerable to transmission errors. Reconstructed mesh surfaces can be severely damaged by transmission errors, which propagate in the bitstreams due to redundancy reduction techniques, such as predictive coding and variable length coding.

Relatively little effort has been made for robust transmission of 3-D mesh data. To transmit 3-D mesh data over lossy networks, Yan et al.7, 8 proposed the multi-seed based partitioning technique to alleviate the error propagation in compressed bitstreams. However, it does not consider the concealment of corrupted regions caused by remaining errors. In [9, 10], the channel coding techniques were proposed to improve the error resilience of progressive bitstreams. However, the channel coding cannot guarantee the recovery of all transmission errors, and it is still possible that the quality of 3-D models is severely degraded by the remaining errors.

In this work, we propose a robust 3-D mesh coding algorithm, based on the shape-adaptive partitioning and the error concealment. First, we divide a mesh surface into several partitions according to the surface smoothness. We then employ the progressive coder in [4] to encode each partition separately. We also propose an encoding scheme for joint partition boundaries, which enables the decoder to zip the boundaries of segments at different levels of details (LODs) seamlessly. Finally, we present an error concealment algorithm, which can reconstruct corrupted regions faithfully. The error concealment algorithm employs the semi-regular connectivity reconstruction and the polynomial interpolation. It is shown that the proposed algorithm provides a good image quality even at severe error-prone environments.

This paper is organized as follows. Section 2 describes the encoding algorithm based on the mesh partitioning method and the progressive partition coding. Section 3 presents the robust mesh reconstruction algorithm in the decoder. Section 4 evaluates the error robustness performance of the proposed algorithm. Then, we conclude the paper in Section 5.
2. ENCODING ALGORITHM

Figure 1 shows the block diagram of the proposed algorithm. An input mesh is partitioned into \( N \) partitions, \( \{ S_i : 0 \leq i < N \} \). We encode each partition separately, using the progressive mesh coder in [4], in order to suppress the error propagation. Each partition is composed of the inner surface region and the joint boundaries. The joint boundaries are encoded with a special rule and the anchor vertex information is transmitted to the decoder, so that the decoder can zip the partitions seamlessly. If the whole information for a partition is lost during the transmission, we hide its effect using the error concealment scheme.

2.1. Shape Adaptive 3-D Mesh Partitioning

Yan et al.\(^7\) proposed the multi-seed based partitioning to alleviate the error propagation in 3-D mesh bitstreams. It first selects a number of seed vertices and then assigns each triangle to the partition of the nearest seed vertex. To obtain partitions of uniform sizes, the seeds are sequentially selected so that a new seed is farthest from the set of the already chosen seeds. The uniform partitioning can prevent transmission errors from influencing specific partitions more severely. However, the seed selection scheme cannot guarantee the uniform distribution of seed vertices. Moreover, it does not consider the shape of mesh surfaces.

In this work, we overcome these problems with a two-step partitioning scheme. First, we classify the surface into smooth regions and detailed regions. Second, we further divide these regions into smaller partitions using the generalized Lloyd algorithm (GLA)\(^11\). Due to the first step, the surface smoothness tends to be homogeneous within each partition. This facilitates faithful concealment of erroneous partitions at the decoder side. Due to the second step, all partitions tend to have similar sizes. Let us describe these two steps subsequently.

We observe the surface smoothness of a given 3-D mesh and classify the mesh into smooth and detailed regions. The surface smoothness can be measured in many ways, e.g., using the Gaussian curvature.\(^12\) In this work, we adopt the normal variation to measure the smoothness. The normal variation of a vertex \( v_i \) is defined as

\[
C_{v_i} = \left[ \frac{1}{N} \sum_{\{t_k, t_l\} \subset F_{v_i}} \mathbf{n}_{t_k} \cdot \mathbf{n}_{t_l} \right]^{-1},
\]

where \( F_{v_i} \) is the set of triangles incident to \( v_i \), \( \{t_k, t_l\} \) is a pair of adjacent triangles in \( F_{v_i} \), \( N \) is the number of adjacent triangle pairs in \( F_{v_i} \), and \( \mathbf{n}_t \) denotes the unit normal vector of a triangle \( t \). The normal variation is easily computable and indicates the surface complexity. In general, a vertex with a high curvature also has a high normal variation. If more than two vertices of a triangle have the normal variations larger than a threshold, the triangle is declared to belong to a detailed region. Otherwise, it is classified into a smooth region. In this work, the threshold is set to \( \cos(\pi/12) \).
After the classification, we adopt the GLA method,\textsuperscript{11} which is useful in clustering unorganized data, to divide each region into partitions. It minimizes the mean square distance between sample points and their corresponding cluster centers. The GLA iteration is summarized as follows.

1. Choose an initial set of centroids $C = \{y_i : 0 \leq i < N\}$.
2. Determine the Voronoi region for each $y_i$.
3. Compute the centroid of each Voronoi region.
4. Go to step 2, if the changes in centroid locations are not negligible. Stop, otherwise.

Note that, in this work, a sample point is a triangle and a Voronoi region corresponds to a mesh partition. Also, the distance between two sample points is measured as the Euclidean distance between the centers of the corresponding triangles.

Figure 2 illustrates the classification and partitioning of the ‘Venus’ model. Figure 2 (b) is the normal variation map, where brighter pixels denote higher normal variances. By thresholding the normal variance map, we obtain the classification result in Figure 2 (c). Blue and red colors depict smooth and detailed regions, respectively. Then, the GLA method provides the final partitioning result in Figure 2 (d). Notice that the normal variance map captures the surface complexity well, and the partitions within each region have similar sizes.

### 2.2. Simplification and Compression of Joint Boundaries

After the partitioning, each partition is encoded separately into an embedded bitstream. At the decoder side, the partitions can be reconstructed at different LODs according to network conditions or user requirements. This complicates the reconstruction procedure. Suppose that two partitions share a joint boundary. To avoid cracks, the boundary should be decoded identically in the two partitions, even if they have different LODs. In this section, we propose an algorithm for the simplification and compression of joint boundaries, which facilitates the seamless reconstruction of the surface in the decoder.

Figure 3 shows the collapse operation for a boundary edge, where the triangle adjacent to $v_tv_u$ and one vertex are erased. For the split operation, we encode the indices of the split vertex $v_{\text{split}}$ and the pivot vertex $v_q$ and the offset vector from $v_{\text{split}}$ to $v_u$. Notice that the boundary is encoded twice, if it is shared by two partitions. In this work, we simplify the joint boundary using the same rule, called the boundary edge collapse rule, so that the decoder can connect the two partitions successfully without additional information.
Figure 3. Edge collapse and vertex split for boundary edges, where the bold-lined edges are boundary edges.

Figure 4. Boundary edge collapse rule. The starting vertices for indexing boundary edges are depicted by solid circles, and the collapsing edges are depicted by bold line segments.

Figure 4 illustrates the boundary edge collapse rule, which successively simplifies the boundary edges between two anchor vertices $v_i^*$ and $v_j^*$. An anchor vertex means a vertex that is shared by more than three partitions. First, we decide the number $e_l$ of boundary edges to be collapsed at level $l$. If the simplification ratio is $r$, $e_l$ is given by

$$e_l = \lfloor b_l(1-r) \rfloor,$$

where $b_l$ is the number of boundary edges at level $l$, and $\lfloor x \rfloor$ denotes the largest integer that is smaller than or equal to $x$. The edges incident on the anchor vertices are not collapsed during the simplification to avoid the position changes of the anchor vertices, and they are not included in counting $b_l$ either.

Next, we select $e_l$ edges to be collapsed. At each level, the boundary edges are indexed from the starting vertex as shown in Figure 4, where the index is labeled above each edge. The collapsing edges are selected so that they are uniformly separated between the two anchor vertices. For instance, if $b_l = 10$ and $e_l = 2$, the 3rd and 6th edges are selected. In general, the index set of collapsing edges is given by

$$\{n : n = \lfloor b_l/(e_l+1) \rfloor \cdot k, \ k = 1, \ldots, e_l\}.$$

Then, the starting vertex at the next level is set to the vertex which is merged from the lastly collapsed edge at the current level. The shift of the starting vertex prevents the concentration of collapsed edges on a specific region and thus yields better visual quality of the simplified boundary.
Figure 5. An example of anchor vertex information. The first element in the anchor vertex information indicates that the partition has three anchor vertices, the next three elements are the positions of the anchor vertices, and the remaining elements are the numbers of boundary vertices between two adjacent anchor vertices.

From the boundary edge collapse rule, both the encoder and the decoder can know the index of $v_{\text{split}}$ without requiring any side information. Therefore, we need to encode the index of the pivot vertex $v_q$ only. Note that the edge connecting the split vertex and the pivot vertex is obtained by merging the triangle in Figure 3. The index prediction scheme in [4] is used to encode the index of the pivot vertex more effectively. In the example of Figure 3, it estimates the index of the pivot vertex as follows. $v_p, v_q,$ and $v_r$ are the candidates for the pivot vertex, and the corresponding angles $\angle(v_o, v_p, v_q), \angle(v_p, v_q, v_r),$ and $\angle(v_q, v_r, v_s)$ are measured. Since $\angle(v_p, v_q, v_r)$ is larger than $\angle(v_o, v_p, v_q)$ and $\angle(v_q, v_r, v_s),$ $v_q$ is selected as the estimated pivot vertex. Then, we encode the index offset between the estimated pivot and the real pivot using an arithmetic coder. The geometry offset vector from $v_{\text{split}}$ and $v_u$ is also encoded using the arithmetic coder.

2.3. Anchor Vertex Information

In the simplification of joint boundaries, the anchor vertices and the number of boundary vertices between two anchor vertices play an important role in deciding the collapsing edges. Moreover, they are used for the decoder to zip adjacent partitions. Thus, they are encoded as the anchor vertex information. Figure 5 shows an example, where the partition has three anchor vertices $v_0^*, v_1^*, \text{ and } v_2^*$. Also, there are 3, 5, and 2 vertices between $v_0^*$ and $v_1^*$, $v_1^*$ and $v_2^*$, and $v_2^*$ and $v_0^*$, respectively. Therefore, the encoder transmits the ordered data

$$(3, p(v_0^*), p(v_1^*), p(v_2^*), 3, 5, 2),$$

where the first element is the number of anchor vertices, and the next three elements are the positions of the anchor vertices in the counterclockwise order, and the remaining elements are the numbers of boundary vertices between two adjacent anchor vertices.

3. DECODING ALGORITHM

As shown in Figure 1, the progressively encoded bitstream and the anchor vertex information are transmitted to the decoder over an error-prone channel. Using the received bitstream, the decoder reconstructs each partition. Then, using the anchor vertex information, the decoder zips the partitions to provide a smoothly connected surface.

The compressed bitstream, however, is vulnerable to transmission errors. If only parts of a partition is corrupted, the decoder still can reconstruct the partition by zipping it with the neighboring partitions. On the other hand, if the whole information of a partition is lost, we perform the concealment of the lost surface based on the polynomial interpolation and the semi-regular connectivity reconstruction.
3.1. Zipping of Joint Boundary

It is straightforward to zip the partitions with the same LOD, since their joint boundary are exactly the same. Figure 6 (a) illustrates the zipping in such a case. The matching of anchor vertices can be easily done with the anchor vertex information. In this example, the zipping of two adjacent partitions starts at $v_a - v_b$ and ends at $v_c - v_d$.

When the upper LOD information in one of the partitions is lost during the transmission, the proposed algorithm should zip the two partitions with different joint boundaries. The zipping can be done easily, since the same boundary edge collapse rule is used to simplify the joint boundary. Suppose that two adjacent partitions are reconstructed up to the $i$th and $(i-1)$th LODs as shown in Figure 6 (b). From the boundary edge collapse rule, the decoder can locate the split vertex $v_{\text{split}}$ within the right partition. Then, a pivot vertex $v_m$ indicating the split edge is also estimated by the index prediction scheme in Section 2. We cut the split edge and make a new triangle to obtain the same joint boundary. Finally, we zip the two partitions seamlessly.

3.2. Concealment of Lost Surface

The base LOD of a partition can be also corrupted. In such a case, the whole information of the partition is useless. In this work, to reconstruct the surface of the missing partition, we propose a 3-D surface concealment algorithm.

We combine the semi-regular connectivity reconstruction and the polynomial interpolation to recover the mesh connectivity and geometry. The surface geometry of a semi-regular mesh is visually pleasing, since its most vertices have valence 6. The polynomial interpolation provides a smooth surface using the information in the neighboring partitions.

Before the reconstruction of the lost partition, we map the neighboring partitions into a 2-D domain to apply the 2-D image processing technique. We adopt the harmonic mapping\textsuperscript{13} in this work. Let $D$ be the neighboring partitions. The harmonic mapping yields an one-to-one map $h : D \to H \subset \mathbb{R}^2$, where $H$ is a polygonal region in $\mathbb{R}^2$. For example, the local 3-D surface in Figure 7 (a) is mapped onto the 2-D domain in Figure 7 (b). The connectivity relations of both Figure 7 (a) and Figure 7 (b) are identical. However, the edge lengths are modified by the harmonic mapping, so that the incurred spring energy is minimized.
Figure 7. Harmonic mapping: (a) domain and (b) range.

Table 1. The number of boundary edges in terms of the base polygon type (q-gon) and the subdivision level l.

<table>
<thead>
<tr>
<th>Subdivision level (l)</th>
<th>Base polygon (q-gon)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4 5 6 7</td>
</tr>
<tr>
<td>0</td>
<td>4 5 6 7</td>
</tr>
<tr>
<td>1</td>
<td>8 10 12 14</td>
</tr>
<tr>
<td>2</td>
<td>16 20 24 28</td>
</tr>
<tr>
<td>3</td>
<td>32 40 48 56</td>
</tr>
<tr>
<td>4</td>
<td>64 80 96 112</td>
</tr>
</tbody>
</table>

After the harmonic mapping, the connectivity of the missing region is reconstructed. We find out the number of boundary edges $B$ using the information in the neighboring partitions. Table 1 contains the number of boundary edges in a semi-regular patch, which is obtained by iteratively applying the 1-to-4 subdivision to a q-gon. We find the minimum value $\hat{B}$ in Table 1, which is larger than $B$, and decide the corresponding polygon type and the subdivision level. Then we split $(\hat{B} - B)$ edges on the joint boundaries so that they match the chosen semi-regular connectivity.

To interpolate the geometry of the lost region, we use the polynomial interpolation. The polynomial interpolation is easy to implement and provides a smooth surface. Note that the harmonic mapping gives us the parametrization from the 2-D polygon to the mesh surface. On the 2-D polygon, we consider three third-order polynomials, which approximate the $x$, $y$ and $z$ components of the mesh surface, respectively. The polynomial coefficients are determined to minimize the mean square error between the approximated surface and the original surface over the non-corrupted area. Then, the missing region is replaced by the approximated polynomial surface. In this way, we retrieve a smooth and semi-regular surface for the missing region.

4. SIMULATION RESULTS

The performance of the proposed algorithm is evaluated on the ‘Venus’ (18,519 vertices), ‘Santa’ (17,661 vertices), and ‘Rabbit’ (15,657 vertices) models in Fig. 8. The ‘Venus’ and ‘Rabbit’ models are relatively smooth, while the ‘Santa’ model has a detailed and complex shape. Each model is pre-quantized, partitioned, and progressively encoded. The pre-quantization converts the original vertex coordinates in real numbers into integer values. In this work, the geometry coordinates are quantized with 12 bits per each coordinate. The compressed
Table 2. The compression performance of the proposed algorithm. ‘♯vtx’ = number of vertices. ‘♯part’ = number of partitions. ‘int bits’ = bits for the interior region encoding. ‘bnd bits’ = bits for the boundary encoding. ‘anchor bits’ = bits for the anchor vertex information. ‘bpv’ = bits per vertex. ‘conv’ = conventional algorithm.4

<table>
<thead>
<tr>
<th>Data</th>
<th>♯vtx</th>
<th>♯part</th>
<th>int bits</th>
<th>bnd bits</th>
<th>anchor bits</th>
<th>proposed bpv (a)</th>
<th>conv bpv (b)</th>
<th>a/b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Venus</td>
<td>18,519</td>
<td>72</td>
<td>431,511</td>
<td>29,624</td>
<td>7,632</td>
<td>25.3</td>
<td>22.7</td>
<td>1.12</td>
</tr>
<tr>
<td>Santa</td>
<td>17,661</td>
<td>70</td>
<td>421,787</td>
<td>28,003</td>
<td>7,400</td>
<td>25.9</td>
<td>21.4</td>
<td>1.21</td>
</tr>
<tr>
<td>Rabbit</td>
<td>15,657</td>
<td>62</td>
<td>369,301</td>
<td>26,784</td>
<td>6,416</td>
<td>25.7</td>
<td>21.4</td>
<td>1.20</td>
</tr>
</tbody>
</table>

The bitstream is transmitted over an error prone channel. In the decoder, the partitions are reconstructed and then zipped. When the base LOD of the partition is corrupted, a surface patch is reconstructed by the error concealment algorithm. The objective distortion between the original 3-D data (S) and the decoded data (S’) is evaluated using the Metro tool,14 which measures the Hausdorff distance between S and S’.

Table 2 summarizes the overall compression results. For each model, there are about 15 LOD levels, and the simplification ratio from the original data to the base LOD is set to 20%. We need the average of 25.6 bits per vertex (bpv) to encode the 3-D data progressively. Compared with the coding performance of [4], the proposed algorithm consumes about 20% more bits to encode the joint boundaries and the anchor vertex information. However, using these additional bits, the proposed algorithm alleviates the error propagation effect. Figure 9 illustrates the progressive reconstruction of the mesh models. We see that the proposed algorithm reconstructs the surface geometry faithfully, even when the bitrate is as low as 5 bpv.

The channel condition is modeled by the corruption states of partitions. In other words, the reconstruction level of each partition is provided by the corruption scenario, where the zero level means the loss of the base layer. In this test, it is assumed that one segment is totally lost and several segments are partially corrupted. Figure 10 shows the totally corrupted partitions, depicted by black regions, and their concealed results. It is observed that the corrupted regions are smoothly connected to the neighboring partitions, yielding an acceptable visual quality. These simulation results indicate that the proposed algorithm is an effective approach to transmit mesh surfaces over error-prone channels.

5. CONCLUSION

In this paper, we proposed a progressive and error-resilient coding algorithm for 3-D meshes. In the encoder, the shape adaptive partitioning scheme was employed to divide a mesh into several partitions. The boundary edge collapse rule was proposed to reduce the amount of bits for joint boundaries and enable the decoder to zip partitions seamlessly. We also developed the polynomial interpolation scheme to hide the effect of missing
partitions. Simulation results demonstrated that the proposed algorithm provides a good quality reconstruction in error-prone environments at the cost of a moderate bitrate increase.

ACKNOWLEDGMENTS

This work was supported in part by the Ministry of Information and Communications, Korea, under the Information Technology Research Center (ITRC) support program.

REFERENCES

Figure 9. Progressively decoded 3-D mesh data at different bitrates. From left to right, the ‘Venus,’ ‘Santa,’ and ‘Rabbit’ models.
Figure 10. Error concealment of missing partitions.