MULTIPLE-DESCRIPTION GEOMETRY COMPRESSION FOR NETWORKED INTERACTIVE 3D GRAPHICS

Pavel Jaromersky
Polytechnic University
Brooklyn, New York, U.S.A.
email: jpavel@cis.poly.edu

Xiaolin Wu
Polytechnic University
Brooklyn, New York, U.S.A
email: xwu@poly.edu

Yi-Jen Chiang
Polytechnic University
Brooklyn, New York, U.S.A
email: yjc@poly.edu

Nasir Memon
Polytechnic University
Brooklyn, New York, U.S.A
email: memon@poly.edu

ABSTRACT
In this paper we propose a novel *multiple-description geometry compression* framework in order to improve the error resilience of streaming 3D graphics contents over lossy packet-switched networks. A 3D polygonal mesh is split into two or more sub-meshes, each of which is compressed *independently* of each other into a so-called *co-description*. If any one of the co-descriptions is received, it can be decoded on its own to an approximation of the original mesh. If several co-descriptions are available to the decoder, they can be decoded in collaboration with each other to refine the approximation. To facilitate the multiple-description geometry compression, we also propose a new surface-based quadratic prediction scheme for 3D polygonal meshes. The prediction residuals are compressed by optimized context-based arithmetic coding. This new geometry coding approach achieves competitive compression performance compared with existing single description and multi-resolution geometry compression methods, with added features of high error resilience and progressive transmission capability.

KEY WORDS
geometric algorithms, geometry compression, multiple-description coding, networked 3D graphics, quality of service, adaptive predictive coding

1 Introduction

In web-based virtual reality and visualization applications, such as virtual presence in electronic collaboration, internet computer games, e-commerce, tele-medicine, object-based video compression, and so on, 3D geometric data are typically communicated in a distributed and networked environment. Real-time streaming of 3D graphics contents over lossy networks such as the Internet and wireless channels, has to combat with adverse transmission conditions such as network congestions, bandwidth fluctuations, the delay and loss of packets, and channel errors. Although the TCP/IP network protocols can provide reliable transmission, they often introduce unpredictable delays. On the other hand, users in these applications can tolerate graceful degradation of rendering quality when network conditions deteriorate, but not excessive delay or stoppage of animation sequences.

To address this issue, a major challenge is to code 3D geometric data in such a way that they become error resilient to packet losses and transmission bit errors. Furthermore, high visual quality of streaming 3D graphics has to be achieved without incurring excessive delays. An existing technique for robust streaming of 3D graphics contents over lossy networks is multi-resolution representation and coding of 3D geometry [21, 19, 16, 2, 7]. The central idea of such approach is to use a layered representation of 3D geometric meshes. A base layer and several enhancement layers are encoded and transmitted in a progressively refinable way. The base (the coarsest) layer offers a low but acceptable level of rendering quality, while each successive enhancement layer refines the quality. An advantage of this approach is that multiple clients with different bandwidths and rendering power can be served by a unified scalable code stream from a single server. For instance, while a high-end workstation client with ethernet connection can stream all layers and obtain high visual quality in real time, a relatively low-bandwidth wireless client may stream only the base layer to get a quick overview. If the wireless channel condition improves later, the latter may also receive the enhancement layers as well. This rate-quality scalability of multi-resolution geometry coding supports streaming graphics in heterogeneous environments.

However, a scalable multi-resolution coding of geometry creates a dependency between refinement layers, called *prefix condition*. Successful decoding of a given layer requires the complete knowledge of all the previous (coarser) layers. In other words, the decoder needs the complete prefix of the code stream to proceed. In the worst case, a problem in the base layer reception interrupts the streaming all together and voids the remaining layers even though they are received perfectly. This results in an undesirable waste of network resources.

To overcome this drawback, we propose in this paper an alternative approach to multi-resolution geometry coding, called *Multiple-Description Coding (MDC)* of 3D geometry. Instead of organizing code stream into embedded layers, MDC generates several separate descriptions of a geometric object, called *co-descriptions*. Unlike the inter-dependent layers in multi-resolution coding, each co-description of MDC can be independently decoded without any knowledge of other co-descriptions. Each extra successfully received co-description improves the fidelity of reconstructed geometry regardless of what has been received so far or in what order. As long as a network client receives one co-description intact at all, no matter which one, a geometric model of a minimally acceptable quality can be reconstructed irrespective of other clients. If a network client receives all co-descriptions without any error, then the maximum-quality geometry reconstruction is achieved. This high code-modularity makes MDC very attractive for streaming graphics contents via packet-switched and wireless networks, where no prioritization is assigned to the packets and all packets are delivered on a best-effort basis without guaranteed reception.

Recently, Al-Regib, Altunbasak and Rossignac proposed to protect multi-resolution compressed 3D mesh data against packet loss errors by uneven error protected packetization (UEP) of Reed-Solomon code [1]. This technique can be viewed as a multiple-description geometry compression scheme because any subset of the resulting packets can be decoded to an approximation of the original 3D mesh. The precision of the approximation is proportional to the number of packets received. However, the UEP technique requires us to add an extra layer of packet erasure channel code on top of the multi-resolution geometry compression scheme. In contrast, our MDC geometry compression technique produces multiple descriptions of a geometric object by partitioning the input dataset into mosaic-like disjoint subsets without explicit channel coding. This design offers the advantage of lower decoding complexity.

Our MDC geometry coding achieves the above mentioned error resilience and adaptability to network dynamics in geometry data transmission at the expense of some reduction in the compression efficiency. This is because in multiple-description coding of discrete geometric objects, a surface or volume dataset is partitioned into several subsets. Some statistical redundancy between the subsets exists and may not be completely removed at the decoder side. Moreover, in order to combine multiple co-descriptions, certain amount of side information is required to describe the connectivity between different data subsets (subsets of vertices). Nevertheless, the improved quality of service in web-based/networked real-time graphics applications easily justifies the slightly reduced efficiency in geometry compression.

The MDC coding of geometric data is also highly desirable for distributed and networked storage systems. For robustness against server down times and also for high throughput, massive geometric datasets can reside on different disk drives or even on distant sites either in entirety or in parts. If the datasets are MDC coded, then a rendering application can simultaneously retrieve data from different storage devices. If any subset of the multiple requests are satisfied, then the rendering process can proceed with the received MDC coded geometric data, the more the packets received, the better the quality of the reconstructed model, whereas a multi-resolution description code will impose a particular order of receiving different layers of data due to the prefix condition. The latter approach is clearly at disadvantage because the arrival order of data packets cannot be controlled by the receiver in distributed and networked data storage systems.

This paper is structured as follows. In the next section we formulate the problem. In Section 3 we discuss the role of connectivity (topology) and its coding in the context of multiple description compression. Section 4 deals with the partition of a 3D mesh into sub-meshes and preparing them for multiple description coding. Section 5 introduces a new surface-based quadratic predictor for adaptive predictive coding of the geometry of individual sub-meshes. Section 6 examines the problem of context-based adaptive arithmetic coding of prediction residuals. Section 7 presents experimental results in comparison with other geometry compression methods. Finally we conclude the paper in Section 8.

2 Problem Formulation

In MDC compression of geometry we partition the input dataset into two or more independent subsets, and code each of them separately into a co-description. The objective of MDC geometry compression is to achieve a good approximation of the underlying geometric model with each individual co-description, while striving for a much refined approximation by joint descriptions that are results of merging two or more co-descriptions.

We focus on MDC compression of 3D triangle meshes in this paper, even though much of the following development can be generalized to other forms of geometric data. MDC geometry compression poses a very hard combinatorial optimization problem even in the case of two co-descriptions. Let us examine a bipartition of a 3D polygonal mesh M into two sub-meshes M_1 and M_2. Each sub-mesh $M_i, i = 1, 2$, with a proper connectivity structure portraits an approximation of the input mesh M. Given a distortion measure for geometric approximation, let $D(M_i)$ be the distortion between M and M_i. Suppose that the i^{th} co-description has a probability P_i to be received successfully which is independent of the reception of the other co-description. Then given M_1 and M_2 the expected distortion of the underlying MDC geometry compression is

$$\begin{aligned} \bar{D}(M_1, M_2) &= P_1 P_2 D(M) + P_1(1-P_2)D(M_1) \\ &\quad +(1-P_1)P_2 D(M_2) \\ &\quad +(1-P_1)(1-P_2)D(\phi) \end{aligned} \tag{1}$$

where $D(M) = 0$ since the union of M_1 and M_2 will reconstruct the original 3D mesh M precisely; $D(\phi)$ is also a constant independent of the bipartition assuming that a fixed or no approximation of M is reproduced if none of the two co-descriptions is received. Therefore, we can formulate the optimal two-description geometry compression problem as to minimize

$$\bar{D}(M_1, M_2) = P_1(1 - P_2)D(M_1) + (1 - P_1)P_2 D(M_2) \quad (2)$$

over all possible bipartitions of M. This is a three-dimensional clustering problem (or vector quantization problem in the terminology of data compression) under a complicated distortion measure (e.g., the volume between M and M_i), which is well known to be NP-complete [8]. Note that the optimal bipartition of a 3D mesh is not convex because the distortion function requires a good global fit between the co-description M_i and the original mesh M.

Given the intractability of optimal solutions to MDC geometry compression, we necessarily resort to heuristic algorithms to solve the problem. One heuristics is to produce two sub-meshes M_1 and M_2 by a down-sampling of the original mesh M as uniformly as possible into an interleaved mosaic (see Section 4). The allocation of vertices of M into M_1 and M_2 is given by

$$\frac{m_1}{m_2} = \frac{P_1(1 - P_2)}{(1 - P_1)P_2}, \quad m_1 + m_2 = m \quad (3)$$

where m_1, m_2, and m are the numbers of vertices in the meshes M_1, M_2, and M respectively. The motivation is to make the quality of a co-description proportional to the probability of successful transmission of the corresponding channel. If more than two co-descriptions are desired, then one can always apply a two-description compression algorithm recursively to provide a tree-structured solution.

However, the decoder of MDC geometry compression needs more than the received sub-meshes to reconstruct the mesh. In order to refine the mesh by merging multiple co-descriptions, the decoder needs the connectivity information that associates vertices in different 3D sub-meshes. The fact that the topology can vary with respect to the same set of vertices differentiates MDC compression of 3D mesh from MDC of other media contents, such as videos and images [9] whose sample connectivity is fixed. The connectivity is a vital side information for the decoder to form joint descriptions. Special care is needed on how to split the vertex set and how to code the connectivity between the vertices in different sub-meshes.

3 Connectivity Side Information

Most representations of 3D meshes consist of two types of data: connectivity information and geometry information. A key issue in MDC geometry compression is whether it is advantageous, and if so, how to split connectivity information into several co-descriptions.

The connectivity information can be expressed using only edges; every triangle in the mesh can be determined by finding loops of three edges among three vertices. If we label the mesh vertices by numbers $1, 2, \ldots, n$, the connectivity information can be described by an adjacency matrix A: $A_{i,j} = 1$ if i and j are connected by an edge, and $A_{i,j} = 0$ if i and j are not connected. The adjacency matrix is symmetric.

For simplicity consider only two co-descriptions, which split the vertices of the mesh into two disjoint sets. Denote the vertices in the first co-description by $1, 2, \ldots m$ and the vertices in the second co-description $m + 1, m + 2, \ldots, n$. Then the original adjacency matrix A can be split into four sub-matrices as follows:

$$A = \left(\begin{array}{c|c} A_{1,1} & B \\ \hline B & A_{2,2} \end{array}\right).$$

The first (resp. second) description should include the connectivity information $A_{1,1}$ (resp. $A_{2,2}$). From the structure of matrix A it is clear that some part of original connectivity information (matrix B) will be shared by the two descriptions. In order to reconstruct the original 3D mesh, the decoder has to know the shared connectivity matrix B to merge the two descriptions into one. The matrix B represents the necessary merge side information in MDC coding of 3D meshes. This side information is the price to pay in the form of extra code length (added redundancy) to improve the error resilience of geometric compression.

A challenging optimization problem is how to form vertex subsets and how to use the connectivity information of $A_{1,1}$ and $A_{2,2}$ to code B (clearly, $A_{1,1}$ and $A_{2,2}$ have to be available to make use of B) so that the code length of merge side information B is minimized. This problem is however beyond the scope of this paper.

Fortunately, in practice the size of connectivity information is rather small compared to the size of geometry information. Furthermore, many efficient methods for encoding the connectivity of a mesh have been developed [22, 10, 23, 20, 14, 3, 11, 15, 18]. Typically the compressed size of the connectivity information is less than 2 bits per triangle [22, 20, 3]. In contrast the data rate for the compressed vertices is about 20–30 bits/vertex (assuming each vertex coordinate is quantized to a 16-bit integer) [12]. Therefore the merge side information B, even if coded suboptimally, does not have a big impact on the overall compression ratio because of its small size.

A simple MDC compression scheme of an input 3D mesh M is to only split the geometry data (vertex set) into two subsets, one for each sub-mesh, and code and transmit the connectivity of the entire mesh M as the common side information for all co-descriptions. Since the connectivity side information is indispensable for reconstruction of any co-description and the joint description, it has to be received by the decoder intact. The global side information can be sent as a separate description by itself via a high quality channel so that its reception is guaranteed. This design is suboptimal in terms of minimizing the overhead of

the side information when only one co-description is received, because the connectivity information of the entire mesh is not split and is shared by co-descriptions.

However, in many practical scenarios, the connectivity (topology) of a 3D dataset remains constant for the life time of the application, whereas the geometry data change constantly. In these situations, the amount of side information on connectivity, being already very small compared to the geometry data, will be negligible if amortized over time. For instances, for on-line interactive computer games, 3D visualization in virtual presence, and object-based video coding, it is well justified in terms of geometry compression to distribute the global connectivity information of a large model to all clients ahead of time. Only the interactive geometry data will be streamed via the Internet in multiple-description code to improve the quality of network service. Note that in these applications streaming 3D geometry data is necessary. The method of transmitting the 3D transformation matrices and rendering the scenes at the client sites does not work, because many motions will be non-rigid-body, non-linear for the visualization of sophisticated interactions between the users and the model, such as warping and morphing.

In networked interactive 3D animation applications, it is also beneficial, in terms of quality of network service, to send global connectivity information to all clients. During the time of network congestions, a client will receive different co-descriptions of compressed geometry data at random as an animation session proceeds. In this case it is possible for each client to utilize the correlation between different sub-meshes M_i to interpolate/estimate the missing vertices in the current animation frame, as long as the global connectivity information is available to all the clients. In other words, the side information on the global connectivity is a form of redundancy that can be effectively utilized to improve the error resilience of networked streaming of 3D animation contents.

4 Mesh Partition for Multiple Description Coding

In this section, we discuss how we partition the 3D triangle mesh into sub-meshes, to be coded into co-descriptions in our MDC compression scheme. As justified in Section 3, we only split the geometry data (vertex set) into subsets, one for each sub-mesh, and code and transmit the connectivity of the entire mesh as the common side information for all co-descriptions. This global connectivity side information is later used during decompression to interpolate/estimate the missing vertices if not all co-descriptions are available, as well as to combine different co-descriptions to improve the quality of the reconstructed model if more than one co-description is received.

Our task of splitting the vertex set into subsets for the co-descriptions is to decide which vertices belong to which co-description. Naively, in the case of two co-descriptions, one could compute a bounding box of the 3D model, split it into two halves, and let each co-description contain all vertices in one half. However, if only one such co-description is received, the quality of the reconstructed model would certainly be very undesirable. Intuitively, for better results, vertices included in each of the co-descriptions should be spaced evenly in the mesh, so that the missing vertices can be interpolated/estimated to the positions that are close to the original ones. In addition, the description of each vertex subset and the description of the global connectivity must be tightly coupled (and efficiently encoded as well to achieve a good compression ratio), in order to make use of the connectivity information to estimate the missing vertices and to combine different co-descriptions.

For densely sampled 3D objects, one way to divide the vertices into subsets whose vertices are spaced evenly is to construct a *vertex spanning tree*: take the graph whose nodes and edges are respectively the vertices and edges of the 3D mesh, and construct a spanning tree T of this graph. To divide the vertices into p subsets of roughly the same size, we can fix some degree-1 node of T as the root, defining for each node its level in the tree (where the root has level 0), and assign each vertex at level ℓ to the i-th subset, where $i = \ell \bmod p$. In this way, if a vertex v is missing from one co-description, at least the near neighbors are included and can be used to estimate v. Moreover, some efficient triangle mesh compression algorithms, such as the *topological surgery* approach of Taubin and Rossignac [22], already use a vertex spanning tree to compress the connectivity information. Therefore, the connectivity information can be efficiently encoded and tightly coupled with the encoding of the vertex spanning tree. This will serve our purpose of MDC compression, with only a very small additional overhead in compression ratios.

Let us briefly review the topological surgery approach [22] for the case of a simple mesh (a mesh that is homeomorphic to a sphere); the technical details for other cases are discussed in [22]. First, a vertex spanning tree T as described above is constructed. If we replace each edge in T with two directed edges, one going toward the root and the other going away from the root, then these directed edges form a loop L around the tree. We use the edges in T to cut the triangle mesh open and flatten it, in a way similar to peeling an orange so that the skin remains connected. The result is a triangulated, simply connected polygon P [22]. Each edge in T appears twice on the polygon boundary, and this boundary is exactly the loop L (ignoring its orientation). Now we construct the dual graph G of the triangulated simple polygon P: each node of G corresponds to a triangle in the triangulation (which is a triangle in the original triangle mesh), and each edge of G connects two nodes that are two triangles sharing a common edge in the triangulation. This dual graph G is a tree, called *triangle spanning tree*. In this way, the two trees T and G are tightly coupled, and can be used together to encode the complete connectivity information of the mesh [22]. The branching nodes and leaf nodes of the tree T (resp.

G) are interconnected by *vertex runs* (resp. *triangle runs*), namely, linear chains consisting of degree-2 nodes in the tree. We encode T and G based on encoding the information about the runs. In particular, marching along a triangle run, we can encode the *marching pattern* to distinguish, whether the current triangle along with its ancestor in the triangle spanning tree G forms a triangle fan or a triangle path. To increase the compression ratio, some heuristic method is used for creating T and G that tries to minimize the numbers of runs in these trees. For the geometry information, the vertex positions are first quantized and then stored compressed by the order of traversing the vertex tree T in a fixed order (e.g., depth-first traversal order), where the quantized vertex positions are encoded by some prediction method, followed by entropy encoding the predictive errors [22].

In our MDC compression algorithm, we use the topological surgery approach [22] to encode the connectivity information. To compress the geometry information in cases where code length is very important, we develop our own *surface-based predictor*, and optimize the encoding of the predictive errors by *context-based arithmetic coding*. We also deal with the missing vertices, in both encoding and decoding of co-descriptions. These techniques will be discussed in Sections 5 and 6.

5 Surface-Based Predictive Coding of Geometry

In accompany with connectivity coding with the topological surgery approach [22], we carry out predictive coding of the coordinates of the vertices. The prediction can be done fast by the *parallelogram rule* [23], or - if extra processing power is available - better yet by our own *surface-based predictor*. Either predictor is based on traversing edge-adjacent triangles, which is the natural outcome of traversing along the triangle runs in the triangle spanning tree.

In this paper we are interested in lossless compression of the vertices after they are quantized to a given precision. The 3D triangle mesh data can be significantly compressed because of the statistical redundancy in the form of spatial coherence between the neighboring vertices. This coherence is largely due to the smoothness of the underlying surfaces that the 3D triangle mesh approximates. A common technique to remove the statistical redundancy of 3D meshes is adaptive predictive coding. The vertices are traversed and coded along a path on the mesh. The spatial location of the current vertex is predicted by its neighbors. The existing prediction methods used in compression of 3D meshes are planar in nature [6, 22, 23, 12], assuming a flatness (small curvature) in local geometry. For instance, the original topological surgery approach [22] uses a linear predictor based on the k ancestors in the vertex spanning tree T of the vertex being predicted, namely,

$$P(\lambda, v_{n-1}, \ldots, v_{n-k}) = \sum_{i=1}^{k} \lambda_i v_{n-i}, \qquad (4)$$

where $v_{n-1}, \ldots, v_{n-k}$ are the k ancestors in T, and $(\lambda_1, \lambda_2, \ldots, \lambda_{n-k})$ is a vector of (fixed) coefficients. Since then, better predictors have been developed. Currently, most state-of-the-art predictors are based on the *parallelogram rule* introduced in [23]: Letting $\mathbf{v}_1, \mathbf{v}_2$ and $\mathbf{v}_3$ be the three vertices of the triangle that is opposite to the vertex $\mathbf{v}$ to be predicted, with $\mathbf{v}_1$ and $\mathbf{v}_2$ connected to $\mathbf{v}$ (see Fig. 1), $\mathbf{v}$ is predicted by flipping $\mathbf{v}_3$ through the midpoint of edge $(\mathbf{v}_1, \mathbf{v}_2)$ so that the edges $(\mathbf{v}_1, \mathbf{v}_2)$ and $(\mathbf{v}_3, \mathbf{v})$ are bisectors of each other (and thus $(\mathbf{v}_1, \mathbf{v}_3, \mathbf{v}_2, \mathbf{v})$ is a parallelogram). Observe that the prediction assumes that the two adjacent triangles $\triangle \mathbf{v}_1\mathbf{v}_2\mathbf{v}_3$ and $\triangle \mathbf{v}_1\mathbf{v}_2\mathbf{v}$ of the triangle mesh are co-planar. To relax such co-planarity assumption, we can adjust the rule by additionally rotating $\mathbf{v}$ around edge $(\mathbf{v}_1, \mathbf{v}_2)$ by an angle between two triangles encountered on the triangle path before. Predictors extending the parallelogram rule include the work given in [17, 5, 12, 13].

On a second reflection, if we want to make the code length even shorter, higher order predictors should work better than linear type of predictors since the assumption of flatness no longer holds in the case of MDC geometry compression. The sampling frequency of the sub-meshes is significantly lower than the original mesh. To exploit the spatial coherence of a coarse mesh in MDC geometry compression, the predictor needs to capture the global trend of the underlying surface. To this end we fit the six previously coded vertices that are topologically closest to the current vertex into a quadratic surface

$$z' = a_1 x'^2 + a_2 y'^2 + a_3 x' y' + a_4 x' + a_5 y' + a_6. \qquad (5)$$

Given the six closest encoded vertices, first their interpolating plane is computed and then the coordinates are transformed such that the interpolated plane becomes $x'y'$ plane of the new coordinate system. The coefficients a_i, $1 \leq i \leq 6$, are determined from the chosen six neighboring vertices in transformed coordinates (x'_i, y'_i, z'_i), $1 \leq i \leq 6$. The transformation tries to minimize variance in the z' coordinate for better fit as well as to remove degenerate cases, where the neighboring vertices would not define a surface (one z value for every (x, y) pair) in the original coordinate system. Under the assumption that the next vertex is close to the resulting quadratic surface, we want to select a point $(\hat{x}', \hat{y}', \hat{z}')$ on this surface as the prediction, and then transform the point back to original coordinate system - $(\hat{x}, \hat{y}, \hat{z})$.

Using the quadratic surface fitting the neighborhood data as one constraint, two more constraints are needed to uniquely determine the prediction point $(\hat{x}', \hat{y}', \hat{z}')$ (which has three unknowns $\hat{x}', \hat{y}'$, and $\hat{z}'$). These additional constraints should be selected according to some domain knowledge about the 3D triangle mesh. A statistically valid property of typical triangle meshes is that they present a

uniform piecewise linear approximation of a curved surface in a locality. In other words, the triangulation consists of roughly equilateral and/or isosceles triangles. This observation suggests a number of ways of designing the predictor.

Referring to Fig. 1 again, let $\mathbf{v}_t = (x_t, y_t, z_t)$, $1 \leq t \leq 3$, be the three vertices of the triangle that is opposite to the vertex $\mathbf{v}$ to be predicted. Then we can set up the two additional constraints by assuming that $\mathbf{v}$ has equal distance to $\mathbf{v}_1$ and $\mathbf{v}_2$, and that $\mathbf{v}_1$ has equal distance to $\mathbf{v}$ and $\mathbf{v}_3$. Namely,

$$\begin{aligned}(\hat{x}-x_1)^2+(\hat{y}-y_1)^2+(\hat{z}-z_1)^2 &= \\ (\hat{x}-x_2)^2+(\hat{y}-y_2)^2+(\hat{z}-z_2)^2 &\end{aligned}$$

$$\begin{aligned}(\hat{x}-x_1)^2+(\hat{y}-y_1)^2+(\hat{z}-z_1)^2 &= \\ (x_1-x_3)^2+(y_1-y_3)^2+(z_1-z_3)^2 &\end{aligned} \quad (6)$$

Solving the system of three equations in (5) and (6) for $(\hat{x}, \hat{y}, \hat{z})$, we obtain a prediction of the location of the next vertex. Note that we can use different constraints to solve for $(\hat{x}, \hat{y}, \hat{z})$ as well. Instead of the assumption that $\mathbf{v}_1$ has equal distance to $\mathbf{v}$ and $\mathbf{v}_3$, we may let $\mathbf{v}_2$ be of equal distance to $\mathbf{v}$ and $\mathbf{v}_3$, or let $\mathbf{v}_2$ be of equal distance to $\mathbf{v}$ and $\mathbf{v}_1$. The encoder can try all these possibilities and select the one to minimize the prediction error. This requires, however, side information to inform the decoder of the choice. Observe that by the nature of the predictor we do not assume that the two adjacent triangles of the mesh, $\triangle \mathbf{v}_1\mathbf{v}_2\mathbf{v}_3$ and $\triangle \mathbf{v}_1\mathbf{v}_2\mathbf{v}$, are co-planar.

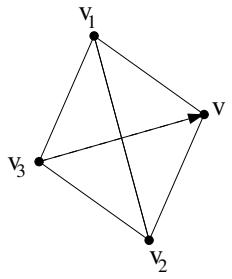

Figure 1. Illustration of the parallelogram rule and our surface-based prediction. In the parallelogram rule, we obtain $\mathbf{v}$ by flipping $\mathbf{v}_3$. In the surface-based prediction, we use the neighboring triangle of $\mathbf{v}$ to establish equal-distance constraints.

When we encode an individual co-description, we avoid using any missing vertices in the prediction to prevent cumulative errors. When we decode, we first recover the available vertex positions (possibly from more than one co-description), and then estimate the positions of the missing vertices from local neighbors. This can be done by using the vertex spanning tree (interpolating from both ancestors and descendants), or by using the parallelogram rule or our surface-based predictor. The estimation can be viewed as a prediction without correcting the predictive error (since we do not have such information), and thus any prediction scheme can be used; the better the prediction, the better the quality of the reconstructed model.

6 Context-Based Arithmetic Coding of Prediction Residuals

The last step of an MDC geometry compression system is entropy coding of prediction residuals for each of the co-descriptions. Let $\{\mathbf{v}_i\}$ be the sequence of vertices in a co-description produced by a traversal of topological surgery, and $\{\hat{\mathbf{v}}_i\}$ be the causal prediction sequence of $\{\mathbf{v}_i\}$. Namely, $\hat{\mathbf{v}}_i = (\hat{v}_{x,i}, \hat{v}_{y,i}, \hat{v}_{z,i})$ denotes the predicted vertex $\mathbf{v}_i = (v_{x,i}, v_{y,i}, v_{z,i})$ that is computed by the surface-based prediction scheme of the previous section. The task of predictive coding of geometry is to compress the corresponding sequence of prediction error vectors $\{\mathbf{e}_i\}$, where $\mathbf{e}_i = \mathbf{v}_i - \hat{\mathbf{v}}_i$.

We code $\mathbf{e}_i$ as a sequence of random vectors rather than three sequences $\{e_{i,x}\}$, $\{e_{i,y}\}$, $\{e_{i,z}\}$, of random variables because there exists strong correlation between the prediction errors in x, y, and z dimensions. Even with our improved surface-based high-order predictor the prediction still cannot remove all statistical redundancy in a 3D surface dataset, in particular if the 3D data are acquired from the real 3D objects. Consequently, the sequence of prediction residuals is not i.i.d. (independently and identically distributed), and it can be viewed to be Markovian. Therefore, we employ context-based arithmetic coding to compress the residual sequence. The minimum code length of the sequence is given by

$$-\log_2 \prod_{i=1}^{n} P(\mathbf{e}_i|\mathbf{e}^{i-1})$$

where $\mathbf{e}^{i-1}$ is the prefix of $\mathbf{e}_i$. Adaptive arithmetic coding could achieve this bound if $P(\mathbf{e}_i|\mathbf{e}^{i-1})$ were known. For real geometric data the conditional probability $P(\mathbf{e}_i|\mathbf{e}^{i-1})$ is unfortunately unknown, and can only be estimated from the sample data. The compression performance is determined by the quality of the probability estimate.

In some aspects estimating $P(\mathbf{e}_i|\mathbf{e}^{i-1})$ for geometry compression is more difficult than for compression of other media contents such as in image/video compression. Firstly, the high numerical precision of geometry data creates a very large alphabet (large dynamic range of $\mathbf{e}_i$) for the coder, up to 32 bits/sample for geometry vs. 8 bits/sample for video/image. Secondly, the data items are vectors instead of scalars. These factors aggravate the well-known problem of *context dilution* in adaptive entropy coding, meaning that the sample data are insufficient for the coder to reach a robust estimate of $P(\mathbf{e}_i|\mathbf{e}^{i-1})$ given the large number of parameters to be estimated.

In order to overcome the difficulty of context dilution, instead of estimating $P(\mathbf{e}_i|\mathbf{e}^{i-1})$ in the huge number of Markov states generated by the subsequence $\mathbf{e}^{i-1}$, we

condition the coding of $\mathbf{e}_i$ only on the prediction errors of vertices that are adjacent to $\mathbf{v}_i$ and are already coded. This context modeling technique is based on the following Markov behavior of prediction errors in space. The surface-based predictor performs very well in smooth regions but breaks down at places of discontinuities such as creases and bumps.

Consequently, small prediction errors are clustered in space, and so are the large errors. Furthermore, the three components in the error vector $\mathbf{e}_i = (e_{x,i}, e_{y,i}, e_{z,i})$ are also highly correlated. Thus, we should also condition the coding of $e_{y,i}$ on $e_{x,i}$, and the coding of $e_{z,i}$ on $e_{x,i}$ and $e_{y,i}$. As a result, we simplify the compression of error vector $\mathbf{e}_i$ to sequential adaptive arithmetic coding of $e_{x,i}$, $e_{y,i}$ and $e_{z,i}$. Finally the predictive coding of the vertex sequence $\mathbf{v}_1\mathbf{v}_2\ldots$ is done by adaptive arithmetic coding of the corresponding sequence of error terms $e_{x,1}, e_{y,1}, e_{z,1}, e_{x,2}, e_{y,2}, e_{z,2}, \ldots$.

Let e_t, $t = 1, 2, \ldots, K$, be the prediction error terms to condition the coding of the current error term e. We use a context quantizer Q to map the local error energy $\sigma = \sum_{1 \le t \le K} |e_t|$ into Θ conditioning states, $Q(\sigma) \in \{1, 2, \ldots, \Theta\}$. A dynamic programming algorithm of optimal context quantization proposed in [24] can be applied to design the context quantizer Q that minimizes the expected code length

$$\sum_{\theta=1}^{\Theta} P(Q(\sigma) = \theta) H(e|Q(\sigma) = \theta) \qquad (7)$$

where $H(e|Q(\sigma) = \theta)$ is the conditional entropy of the prediction errors in the m^{th} coding state.

To deal with the problem of large alphabet we adopt an escape coding mechanism [4]. This is based on the observation that the distribution of prediction errors is heavily centered around zero, although it has very long and sparsely populated tails. For each coding state $Q(\sigma) = \theta$, we find the interval $[-r_\theta, r_\theta]$ such that $\sum_{e=-r_\theta}^{r_\theta} P(e|Q(\sigma) = \theta) = 0.98$. The length of the interval $[-r_\theta, r_\theta]$ is fairly narrow, being typically around $[-64, 64]$ depending on θ (the larger the value of θ, the larger the variance of $P(e|Q(\sigma) = \theta)$. This greatly reduced alphabet size makes arithmetic coding practical and efficient (no waste of code space on zero-frequency symbols). In the θ^{th} coding state, if $e \in [-r_\theta, r_\theta]$ then it is coded by the proposed context-based arithmetic coding scheme; otherwise an escape symbol is sent to signal that e will be specified by a simple range coding scheme.

7 Experimental Results

We have implemented the proposed algorithm for multiple-description compression of 3D meshes, and tested it on a number of datasets. Table 1 lists the results of two-description compression in comparison with the single-description algorithm of topological surgery [22], and the multi-resolution compression method of [7], which is based on k-d tree and is one of the few techniques that focus on compressing the geometry information (vertex coordinates) while providing a multi-resolution capability.

In order to evaluate our MDC algorithm for both symmetric channels ($P_1 = P_2$) and asymmetric channels ($P_1 \neq P_2$), we ran it for both even and uneven partitions of the input 3D mesh. Recall from Section 2 that P_i, $i = 1, 2$, is the probability for the i^{th} co-description to be received successfully. The allocation of vertices of input mesh M to the two sub-meshes M_1 and M_2 is governed by Equation (3). The compression results of both even (1:1) and uneven (2:1) mesh partitions are given in Table 1.

For multiple-description geometry compression, if we sum up the sizes of three compressed files: connectivity, co-description 1 and co-description 2, the total code length is only slightly larger than the total code length of the multi-resolution method. However, the multi-resolution geometry code stream, without proper protection of packet erasure code, is highly susceptible to channel noise due to the prefix condition as discussed in Section 1. One has to add extra bits of packet erasure code to the multi-resolution code stream in order to match the high error resilience performance of multiple-description geometry compression in packet erasure channels.

Furthermore, recall from Section 3 that in many networked animation and visualization applications, the connectivity data will be shared by a large number of frames, and hence amortized over time to a negligible amount of side information. In this case, the performance of our MDC geometry compression will be determined by the size of compressed geometry data only, favoring the proposed MDC approach.

Finally, we show in Figures 2–5 some reconstructed 3D models decoded from either and both of their two co-descriptions as listed in Table 1. Comparing the image quality, we can clearly see that when only one co-description is received, no matter which one, our approach can still reconstruct an approximate model that is very close to the original one. This is advantageous over the multi-resolution compression scheme, in which the result of a packet loss is unpredictable, in the worst case no reconstruction is possible, due to the prefix condition.

8 Conclusions

We have presented a multiple-description coding framework for error resilient compression of 3D meshes. A new surface-based quadratic prediction scheme and optimized context-based arithmetic coding are integrated into this framework. The resulting MDC geometry compression method has a competitive compression performance compared with existing single description and multi-resolution geometry compression methods, while offering much improved error resilience in the presence of channel errors, as well as progressive transmission capability.

Dataset	Bunny	Horse	Bone	Dinosaur	Rabbit	Venus	Triceratops
Number of vertices	34834	48485	137062	56194	67038	134345	2832
Number of triangles	69451	96966	274120	112384	134074	268686	5660
Single Description							
Connectivity	14191	15047	39964	18464	20186	39424	1312
Geometry-Linear	76717	92626	251161	104766	121461	247261	7865
Geometry-Parallelogram	54690	70507	178484	83540	91625	171125	7431
Multi-resolution Connectivity	17380	24219	72736	31369	34370	67591	2278
Multi-resolution Geometry	61327	71491	179691	70694	94057	184882	6378
Multiple-Description 1:1							
Connectivity	14191	15047	39964	18464	20186	39424	1312
Geometry: Co-description 1	38693	47304	117235	55915	61128	118064	4322
Geometry: Co-description 2	38510	47212	117031	56071	61319	118176	4355
Geometry: Total	77203	94516	234266	111986	122447	236240	8677
Multiple-Description 2:1							
Connectivity	14191	15047	39964	18464	20186	39424	1312
Geometry: Co-description 1	52547	64851	174205	74370	86012	167350	5612
Geometry: Co-description 2	29343	35879	92372	41791	45521	89223	3179
Geometry: Total	81890	100730	266577	116161	131533	256573	8791
4 descriptions							
Connectivity	14191	15047	39964	18464	20186	39424	1312
Geometry: Co-description 1	23724	29150	71284	33293	36417	73318	2539
Geometry: Co-description 2	23739	29093	71230	33302	36402	73475	2561
Geometry: Co-description 3	23832	29152	71146	33507	36830	73285	2521
Geometry: Co-description 4	23703	29204	71046	33526	36788	73390	2515
Geometry: Total	94998	116599	284706	133628	146437	293468	10136

Table 1. File sizes in bytes of seven 3D triangle meshes compressed by: (a) the single-description compression algorithm of topological surgery [22] using the linear predictor (based on 4 ancestors in the vertex spanning tree) and the parallelogram rule; (b) the k-d tree-based multi-resolution compression algorithm [7]; (c) the proposed MDC compression algorithm in both co-descriptions and for both even (1:1) and uneven (2:1) partition of the input meshes. (d) Partition to 4 descriptions. Each vertex coordinate is first quantized into a 12-bit integer before compression. Therefore, before compression the dataset size is 36 bits/vertex for geometry and 12 bytes/triangle for connectivity.

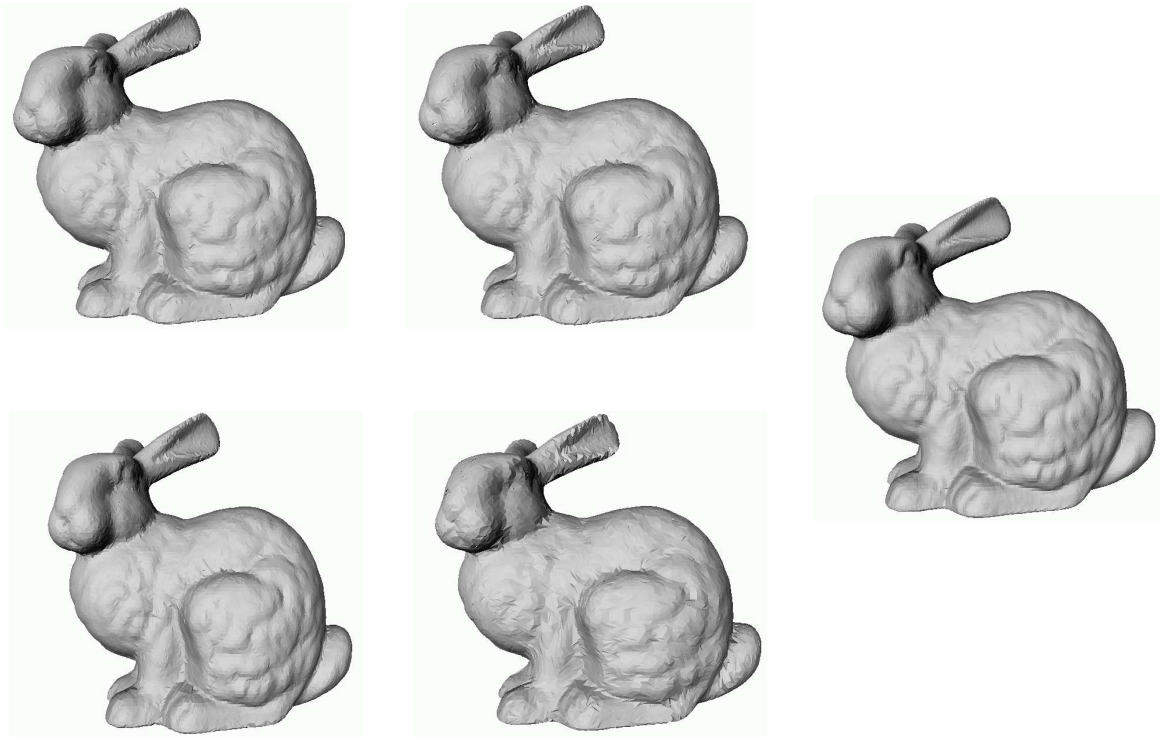

Figure 2. Reconstruction results of the Bunny model. Left column: co-description 1; middle column: co-description 2; right column: both co-descriptions. Top row: even partition; bottom row: uneven partition.

References

[1] G. Al-Regib, Y. Altunbasak, and J. Rossignac. An unequal error protection method for progressively compressed 3-d meshes. In *Proceedings of IEEE INFOCOM*, 2002.

[2] P. Alliez and M. Desbrun. Progressive encoding for lossless transmission of 3d meshes. In *Proceedings of ACM SIGGRAPH*, pages 198–205, 2001.

[3] P. Alliez and M. Desbrun. Valence-driven connectivity encoding for 3D meshes. *Proc. Eurographics 2001*, pages 480–489, 2001.

[4] T. Bill, J. Cleary, and I. Witten. *Text compression*. Prentice-Hall, 1990.

[5] D. Cohen-Or, R. Cohen, and R. Irony. Multi-way geometry encoding. TR-2002.

[6] Michael F. Deering. Geometry compression. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 13–20. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

[7] P.-M. Gandoin and O. Devillers. Progressive lossless compression of arbitrary simplicial complexes. *ACM Trans. Graphics*, 212(3):372–379, 2002. Special Issue for SIGGRAPH '02.

[8] M. Garey, D. S. Johnson, and H. S. Witsenhausen. The complexity of the generalized lloyd-max problem. *IEEE Trans. on Inform. Theory*, 28:255–266, 1982.

[9] Jerry Gibson, Toby Berger, Tom Lookabaugh, Rich Baker, and David Lindbergh. *Digital Compression for Multimedia: Principles & Standards*. Morgan Kaufmann, 1998.

[10] S. Gumhold and W. Straser. Real time compression of triangle mesh connectivity. In *SIGGRAPH 98 Conference Proceedings*, pages 133–140, 1998.

[11] M. Isenburg. Compressing polygon mesh connectivity with degree duality prediction. In *Proc. Graphics Interface*, pages 161–170, 2002.

[12] M. Isenburg and P. Alliez. Compressing polygon mesh geometry with parallelogram prediction. In *Proc. Visualization*, pages 141–146, 2002.

[13] M. Isenburg and S. Gumhold. Out-of-core compression for gigantic polygon meshes. *ACM Trans. Graphics*, 22(3):935–942, 2003. Special Issue for SIGGRAPH '03.

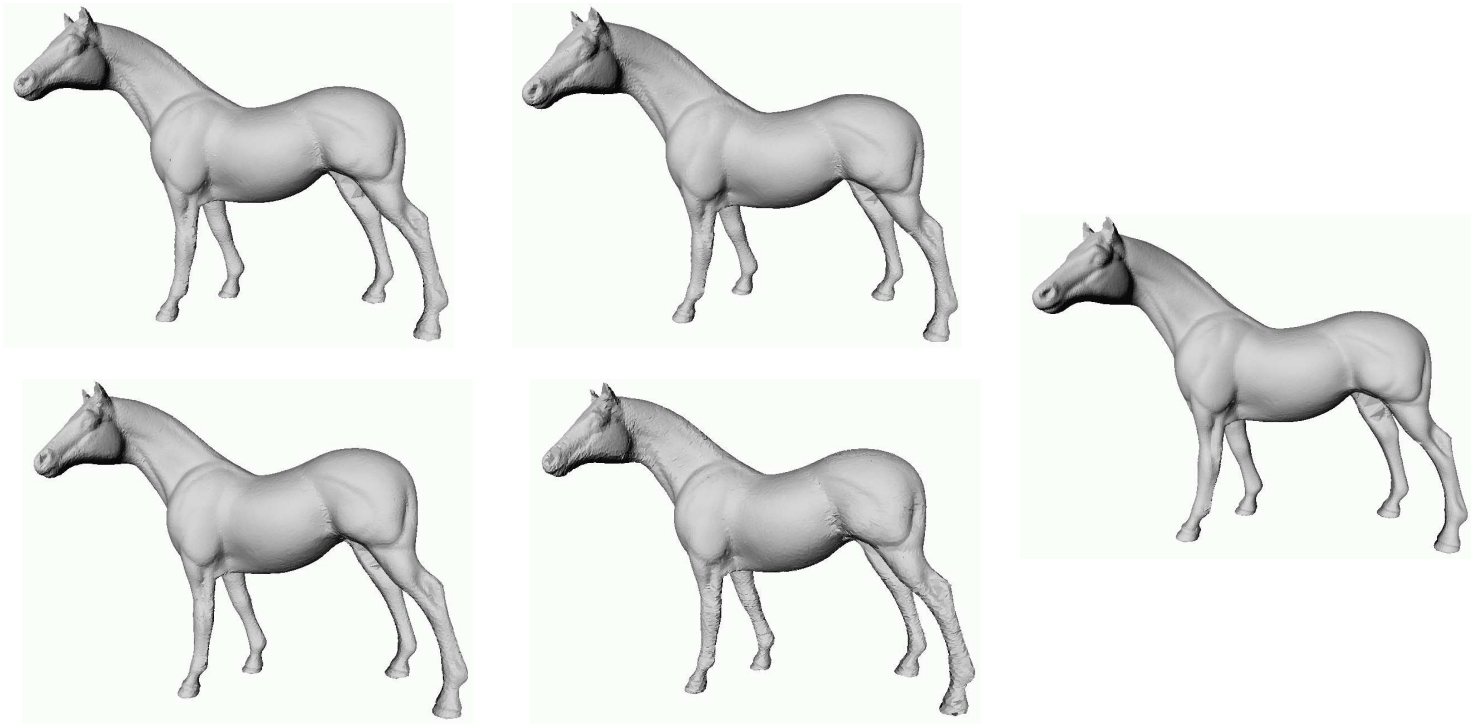

Figure 3. Reconstruction results of the Horse model. Left column: co-description 1; middle column: co-description 2; right column: both co-descriptions. Top row: even partition; bottom row: uneven partition.

[14] M. Isenburg and J. Snoeyink. Face fixer: Compressing polygon meshes with properties. In *SIGGRAPH 00 Conference Proceedings*, pages 263–270, 2000.

[15] Z. Karni and C. Gotsman. Spectral compression of mesh geometry. In *SIGGRAPH 00 Conference Proceedings*, pages 279–286, 2000.

[16] A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive geometry compression. In *Proceedings of ACM SIGGRAPH*, pages 271–278, 2000.

[17] B. Kronrod and C. Gotsman. Optimized compression for triangle mesh geometry using prediction trees. In *Proc. Sympos. on 3D Data Processing, Visualization and Transmission*, pages 602–608, 2002.

[18] H. Lee, P. Alliez, and M. Desbrun. Angle-analyzer: A triangle-quad mesh codec. In *Computer Graphics Forum*, 2002.

[19] R. Pajarola and J. Rossignac. Compressed progressive meshes. *IEEE Trans. on Visualization and Computer Graphics*, 6(1):79–93, 2000.

[20] J. Rossignac. Edgebreaker: Connectivity compression for triangle meshes. *IEEE Trans. Visualization Computer Graphics*, 5(1):47–61, 1999.

[21] G. Taubin, A. Guéziec, W.P. Horn, and F. Lazars. Progressive forest split compression. In *Proceedings of ACM SIGGRAPH*, pages 123–132, 1998.

[22] G. Taubin and J. Rossignac. Geometric compression through topological surgery. *ACM Trans. Graphics*, 17(2):84–115, 1998.

[23] C. Touma and C. Gotsman. Triangle mesh compression. In *Proc. Graphics Interface*, pages 26–34, 1998.

[24] X. Wu. Lossless compression of continuous-tone images via context selection, quantization, and modeling. *IEEE Trans. on Image Processing*, 6:656–664, 1997.

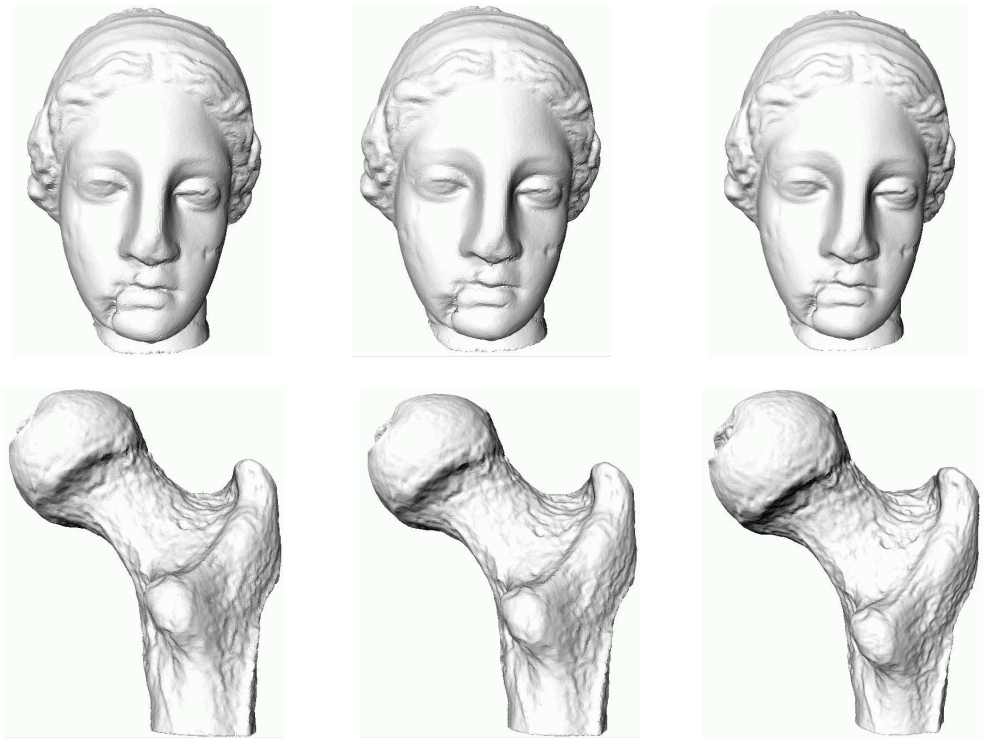

Figure 4. Reconstruction results of even partition. Left column: co-description 1; middle column: co-description 2; right column: both co-descriptions. Top row: the Venus model; bottom row: the Bone model.

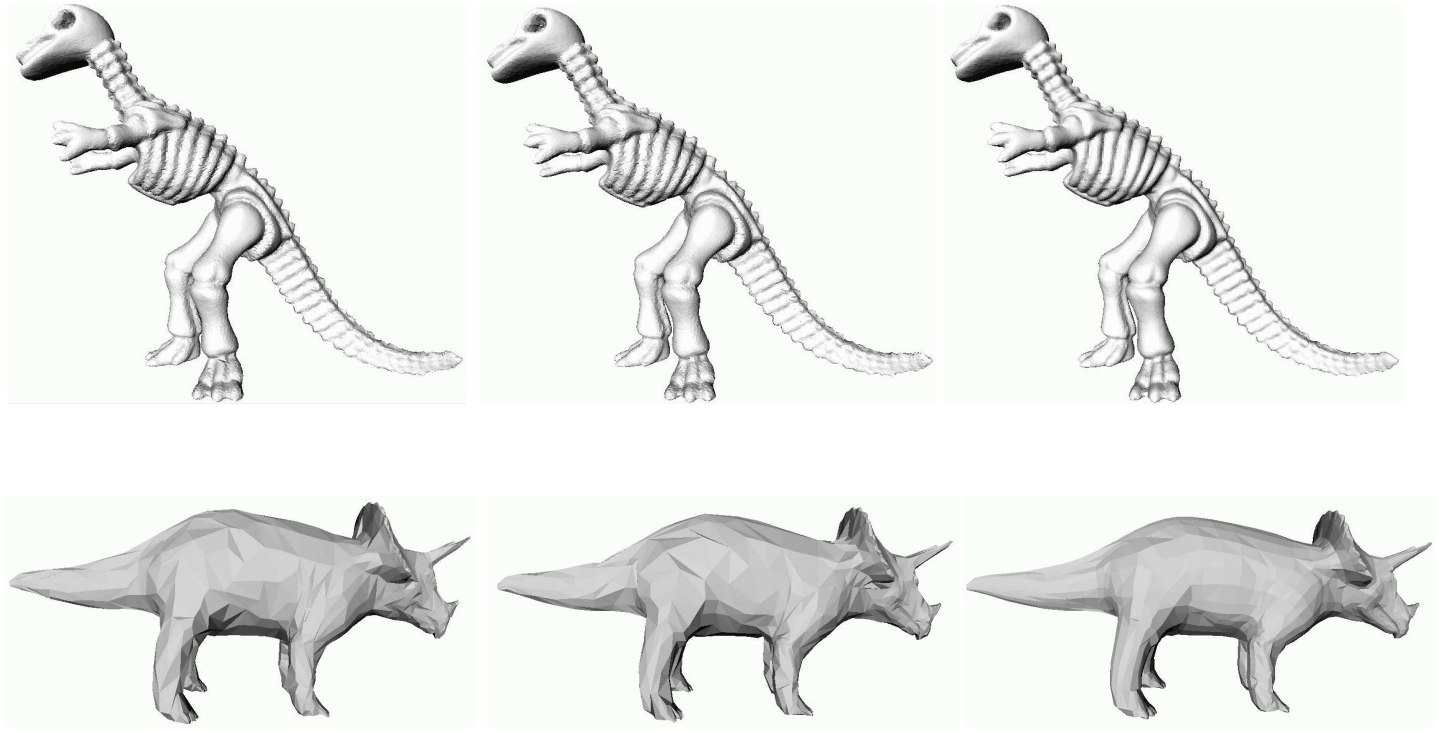

Figure 5. Reconstruction results of even partition. Left column: co-description 1; middle column: co-description 2; right column: both co-descriptions. Top row: the Dinosaur model; bottom row: the Triceratops model.