Point Cloud Compression Based on Hierarchical Point Clustering

Yuxue Fan and Yan Huang* and Jingliang Peng*
School of Computer Science and Technology, Shandong University, China
* Corresponding authors (E-mails: yan.h@sdu.edu.cn, jpeng@sdu.edu.cn)

***Abstract*—In this work we propose an algorithm for compressing the geometry of a 3D point cloud (3D point-based model). The proposed algorithm is based on the hierarchical clustering of the points. Starting from the input model, it performs clustering to the points to generate a coarser approximation, or a coarser level of detail (LOD). Iterating this clustering process, a sequence of LODs are generated, forming an LOD hierarchy. Then, the LOD hierarchy is traversed top down in a width-first order. For each node encountered during the traversal, the corresponding geometric updates associated with its children are encoded, leading to a progressive encoding of the original model. Special efforts are made in the clustering to maintain high quality of the intermediate LODs. As a result, the proposed algorithm achieves both generic topology applicability and good rate-distortion performance at low bitrates, facilitating its applications for low-end bandwidth and/or platform configurations.**

I. Introduction

Enabled by the advances in computer graphics technology, high definition models are being increasingly produced and used in many fields including gaming, filming and scientific simulation. Along with the increasing data size and application scope of 3D models comes the challenging issue of 3D model compression for efficient utilization of the storage and the bandwidth resources. In particular, for network-based graphics applications, progressive transmission and reconstruction is frequently desired to enable a smooth user experience at the client side.

Point primitives have been more and more used for representing and rendering 3D models since their first use by Levoy and Whitted [7] for 3D surface display. In contrast to a 3D polygonal mesh that is composed of vertices, edges and facets, a 3D point-based model, often called 3D point cloud as well, is composed of point primitives. The point-based representation is especially suitable for densely sampled 3D surfaces, saving the need of storage for connectivity information.

In this work, we focus on compressing the geometry of a 3D point cloud. That is, we focus on compressing the points' positions and normals. While many published algorithms on 3D point cloud compression have yielded good coding performance, not sufficient attention has yet been paid to the low bitrate performance of point cloud coding, which however is key to applications scenarios with limited network bandwidth and/or client rendering capabilities. Further, not all those methods are suitable for surfaces of generic topology, limiting their generic applicability. As such, in this work we propose a scheme for progressive compression of point clouds, whose distinctive features include:

- **Generic applicability.** Since the proposed method is based on point clustering, it is not restricted to any specific type of surface topology (*e.g.*, manifold) but literally can process surfaces of arbitrary topology, be it manifold or non-manifold.
- **Outstanding low bitrate coding performance.** The proposed method yields outstanding rate-distortion performance, especially at low bitrates, mainly because:
  - the quality of the intermediate LODs are optimized by the optimized point clustering process, and
  - the geometry update encoding is made highly efficient with various techniques including cylindrical coordinate representation, effective prediction, Gaussian sphere based normal representation and prediction, and entropy coding.

The rest of this paper is organized as follows. The related work is surveyed in Section II, an overview of the proposed algorithm is given Section III while the details of the proposed algorithm follow in Section IV and V, the experimental results are provided and analyzed in Section VI and the conclusion is drawn in Section VII.

II. Related Work

Since the beginning of this century, point-based 3D model compression has attracted intensive research attention. Rusinkiewicz and Levoy [11] have developed a mutiresolution point-based large model rendering system. While not specifically being a compression work, it compactly represents a bounding-sphere hierarchy by a 48-bit quantization of the position, normal and color attributes at each node. Gumhold *et al.* [3] proposed a single-rate encoder that builds a prediction tree for the input model and uses it to guide the prediction and entropy coding process. Krüger *et al.* [5] encode an input model into multiple LODs by constructing Multiple Hexagonal Close Packing (HCP) grids of various resolutions and encoding the sequence of filled cells for each HCP grid. Their encoder is however not a progressive one since the coding bits of the coarser LODs are not embedded in those of the finer ones.

Many algorithms on progressive point-based model compression have been proposed which enable adaptiveness to runtime bandwidth and client rendering capabilities. Some of these algorithms (*e.g.*, [2], [8], [16]) work best for samples from manifold surfaces while the others (*e.g.*, [14], [6], [13], [1], [12], [4], [15]) compress models of generic topological types.

Fleishman *et al.* [2] propose a multilevel point-based representation and reduce the coefficients from 3D to 1D for efficient compression. Ochotta and Saupe [8], [16] partition the model surface into relatively flat patches, each of which is converted to a height field and encoded using 2D wavelet coding techniques.

Wu *et al.* [14] simplify the original model into multiple LODs with a sequence of virtual edge collapses each contracting two points to one. The reverse of this iterative simplification process is encoded. Kalaiah and Varshney [6] conduct cluster-based hierarchical Principal Component Analysis (PCA) which leads to an efficient statistical geometry representation. However, no rate-distortion (R-D) data are given in these two works ([14], [6]), probably because their research focus is more on efficient rendering than on compression. Waschbüsch *et al.* [13] use point pair contractions to iteratively reduce the original model to multiple LODs and encode the reverse of this simplification process. However, the number of LODs that are generated and encoded appears to be limited.

Some algorithms [1], [12], [4], [15] are based on the iterative octree-partitioning of the object space. Associated with each octree cell subdivision, they need to specify which child cells are nonempty and the nonempty cells are further iteratively subdivided. Botsch *et al.* [1] use a byte code to indicate which child cells are non-empty. Inspired by Peng and Kuo's work on 3D mesh compression [10], Schnabel and Klein [12] encode for each cell subdivision the number of nonempty child cells and the index of the nonempty child combination. Huang *et al.* [4], [15] propose a local neighborhood based method for nonempty child prediction, and employ Gaussian sphere based normal representation and prediction, leading to good geometry coding performance. In addition to the geometry, all the three works [12], [4], [15] encode color information as well.

III. Overview

Three processes are involved in the proposed point cloud compression scheme: the LOD hierarchy construction process constructs a hierarchy of LODs for an input model; the LOD hierarchy encoding process traverses the LOD hierarchy from the root down to the leaves in a width-first order and encodes the model refinement associated with each node, leading to a progressive encoding of the input model; the LOD hierarchy decoding process decodes the received bit stream and restores the model progressively from low to high LODs. Since the LOD hierarchy encoding and decoding processes are reverse to each other, we will focus on only the LOD hierarchy construction and the encoding processes in the following algorithm description. It is noteworthy that we focus on the geometry coding only in this work, *i.e.*, we compress the positions and the normals of the points, but not other attributes like color.

LOD Hierarchy Construction: We reduce the resolution of the original model through clustering the original points and computing a representative for each cluster, all of which from an approximation to the original model. Iteratively conducting the clustering process, we obtain a hierarchy of LODs for the input model. For the point clustering, we adapt the Generalized Lloyd Algorithm (GLA) in order to maximize the approximation quality of the intermediate LODs. In particular, we propose a distance metric to drive the clustering process which takes both the position and the normal attributes into account for best preserved quality of the approximating LODs.

We construct an LOD hierarchy where each node corresponds to a point primitive and all the point primitives on the same level form an LOD of the model. Starting from the input model, each coarser LOD is constructed from a finer one through the clustering process and each point in the former represents a cluster of points in the latter, forming a parent-children relationship between those points in the hierarchy.

It is worthwhile to point out that the octree partitioning based point cloud encoders [1], [12], [4], [15] are also based on LOD hierarchy construction, but the LODs they generated are often blocky at low point counts. In contrast, the GLA-based approach produces significantly better model quality at similar point counts. It is equally worth mentioning that, GLA-based clustering is also used by Peng *et al.* [9] in the context of progressive 3D mesh compression. They perform the GLA-based clustering and encoding in one pass, from the coarsest LOD (*i.e.*, the root) to the finest. This top-down approach performs aggressive clustering at coarser LODs and therefore may easily smooths out sharp features in coarser LODs. As such, in this work, we perform bottom-up LOD construction and reduce the resolution of the input model gradually, with an effort to preserve sharp features better in coarser LODs. In addition, the GLA process is adapted in various aspects (*e.g.*, cardinality control, distance metric and representative computation) to work on the point primitives.

LOD Hierarchy Encoding: The encoder traverses the LOD hierarchy from the root down to the leaves in a width-first order. At each node during the traversal, the encoder encodes the position and the normal information of its child nodes, corresponding to a local refinement on the current LOD. Differential coding is conducted for both the position and the normal coding. Essentially, each child node's attributes are predicted from its parent's, and the differences are encoded.

For the position encoding, a local coordinate frame is set up to align with the position and orientation of the parent node; then the positional difference between each child and the parent is expressed in cylindrical coordinates in this local frame. Adaptive local quantization and effective prediction are made to the child nodes' cylindrical coordinates for coding efficiency. The normal residual coding is made with the help of the hierarchical Gaussian sphere subdivision and the local neighborhood linearization on the Gaussian sphere.

It is noteworthy that techniques from reference [9] are adapted for our position encoding and the method from reference [4], [15] is used for our normal encoding.

Notation: In this paper we adopt the surfel set representation for point-based models. A surfet set is defined as $S = \{s_i | 1 \leq i \leq n\}$, where n is the total number of points in the model and $s_i = \{\mathbf{p}_i, r_i, \mathbf{n}_i, \mathbf{c}_i\}$ denotes the i-th surfel

that is centered on $\mathbf{p}_i$ with a radius of r_i, a normal of $\mathbf{n}_i$ and a color of $\mathbf{c}_i$. As [4], [15], we do not encode the radius information since the radii can be easily reconstructed from the surfels' positions and normals. Specifically, we only encode the position and the normal information of the input surfels in this work. As such, we may sometimes omit the radius and/or the color terms from the surfel's denotation in the following text.

IV. LOD Hierarchy Construction

The LOD hierarchy is constructed through an iterative point clustering process. The bottom level of the hierarchy represents the original input model, each leaf node of which corresponds to a point in the original model. At each iteration, we reduce the number of point primitives by dividing them into clusters and computing a representative point for each cluster, all of which form a coarser LOD at one level up in the hierarchy. The parent-child relationship is specified between the nodes of a representative and each point in the associated cluster. The root level contains one node corresponding to the coarsest LOD containing only one point.

A. Point Clustering

We adapt the well-known GLA for the clustering purpose The adaption mainly happens in three aspects: cluster cardinality control, definition of distance metric, and representative computation.

Cardinality Control: The standard GLA makes the clustering based on the spatial distribution of data samples without explicit control on the cardinality of each cluster, which may lead to unbalanced LOD hierarchy and correspondingly increased entropy for data compression. As such, we prefer to evenly distribute the primitives among clusters while maximizing the approximation quality of the intermediate LODs at the same time. For a given LOD $S = \{s_i | 1 \leq i \leq n\}$, the optimized GLA algorithm to generate the coarser LOD S' with $k = \lceil n/a \rceil$ surfels is described as follows.

1) *Initialization:* Pick randomly an initial set of representatives, $Q = \{q_j | 1 \leq j \leq k\} \subset S$
2) *Iterative clustering:* For $l = 1, 2, \cdots$, we perform the following.
   a) Divide S into subsets G_j, $1 \leq j \leq k$ using the nearest neighbor rule and cardinality control. Specifically,
      i) Set $G_j = \emptyset$, $1 \leq j \leq k$, reset array A, and mark all $s_i \in S$ as unoccupied.
      ii) For each surfel $s_i \in S$, we compute its distance to q_j, $j = 1, 2, \ldots, k$, find the smallest distance d_i corresponding to q_{j_i}, and insert (d_i, i, j_i) into A whose elements are sorted in the ascending order of d_i's.
      iii) Starting from the beginning of A, for each $(d, i, j) \in A$, if s_i is not occupied and $|G_j| < a$, set $G_j = G_j \cup \{s_i\}$ and mark s_i as occupied;
      iv) For any $s_m \in S$ that is neither occupied nor clustered, put it in G_{j_m}, *i.e.*, $G_{j_m} = G_{j_m} \cup \{s_m\}$.
   b) Compute the new representative, q_j, from all the surfels in G_j, $1 \leq j \leq k$, update the representative set Q and compute the distortion

$$E_l = \frac{1}{n} \Sigma_{j=1}^{k} \Sigma_{s \in G_j} d(s, q_j).$$

3) *Stopping criterion*: The clustering iteration stops if $(E_{l-1} - E_l)/E_l < \delta$ or $l = L$, where δ and L are design parameters.
4) *Final result*: Q contains the final set of representatives, forming the coarser LOD, *i.e.*, $S' = Q$.

Distance Metric: We propose a formula taking into account both the position and the normal attributes; namely, given two surfels, $s_1 = \{\mathbf{p}_1, r_1, \mathbf{n}_1\}$ and $s_2 = \{\mathbf{p}_2, r_2, \mathbf{n}_2\}$, their distance is computed as

$$d(s_1, s_2) = r_1^2 \cdot r_2^2 (w_p \cdot |\mathbf{p}_1 - \mathbf{p}_2| + w_n \cdot |\mathbf{n}_1 - \mathbf{n}_2|). \quad (1)$$

We empirically use $w_p = w_n = 1.0$ in our experiments.

Representative Computation: For a surfel cluster, $G = \{g_t = \{\mathbf{p}_t, r_t, \mathbf{n}_t\} | t = 1, 2, \ldots, T\}$, we compute its representative surfel, $q = \{\mathbf{p}, r, \mathbf{n}\}$, to capture the geometric characteristics of the surfels in G as precisely as possible. Specifically, we compute $\mathbf{p}$, $\mathbf{n}$ as area-weighted average of $\mathbf{p}_t$ and $\mathbf{n}_t$, $t = 1, 2, \ldots, T$, respectively, and compute r as $r = \alpha \cdot (\Sigma_{t=1}^{|G|} r_t^2)^{1/2}$. Here α is a design parameter.

V. LOD Hierarchy Encoding

The LOD hierarchy is traversed level by level, from the root down to the leaves. A queue is used to dynamically maintain the nodes on the current tree front. During the traversal, we visit each node on the tree front, encode the geometric refinement associated with its children, and replace it with its children in the tree front.

For each node encountered during the traversal, we encode the position and the normal updates specified by the child nodes, as described in the following subsections. We denote the parent surfel as $q = \{\mathbf{p}, r, \mathbf{n}\}$ and the T child surfels as $v_i = \{\mathbf{p}_i, r_i, \mathbf{n}_i\}, i \in [1, T]$.

A. Position Encoder

The position encoder encodes the difference between each $\mathbf{p}_i, i \in [1, T]$ and $\mathbf{p}$. Similar to Peng *et al.* [9], we represent the children's spatial offsets in a local frame centered on $\mathbf{p}$ using cylindrical coordinates, conduct adaptive local quantization to the coordinates and make effective predictions to reduce the data entropy. However, we design different approaches to some of the above steps due to the specific characteristics of point primitives, as will be emphasized in the following description.

We firstly define the local frame, $F = (\mathbf{O}, \mathbf{x}, \mathbf{y}, \mathbf{z})$. We set $\mathbf{O} = \mathbf{p}$. Since we encode the normal information as well, we directly use the unit normal vector, $\mathbf{n}$, at q as the $\mathbf{z}$ basis vector, in contrast to Peng *et al.* [9] who compute a best fitting plane in the local neighborhood to obtain the $\mathbf{z}$ vector. We use the

same method as Peng *et al.* [9] to define the local $\mathbf{x}$ and $\mathbf{y}$ basis vectors; namely, we project the global frame's $\mathbf{x}$ basis vector onto the tangential plane at $\mathbf{p}$ to get the local $\mathbf{x}$ basis vector, and compute the local $\mathbf{y}$ basis vector as $\mathbf{z} \times \mathbf{x}$.

Thereafter, we convert the Euclidean coordinates of the children's positions to cylindrical coordinates in this local frame to obtain $(\rho_i, \theta_i, h_i), i \in [1, T]$. For the θ quantization, we evenly divide the range of $[0, 2\pi]$ to Q bins, as Peng *et al.* [9] do; for the ρ and h quantizations, however, we perform differently. Peng *et al.* [9] adaptively determine the quantization parameters for ρ and h based on the geometrical extents of the local neighborhood. For polygonal meshes, the local neighborhood is readily available from the local connectivity information. However, there is no explicit connectivity between surfels in a point-based model. We do not choose to explicitly construct and utilize the neighborhood relationship either for computation- and memory-efficiency considerations. As such, we may alternatively determine the range of ρ and h coordinates adaptively based on the radius, r, of surfel p. However, our point cloud encoder does not encode the radius of each surfel. Instead, we compute and encode the average surfel radius for each LOD with negligible bit cost, which provides the quantization range for ρ and h coordinates on each level of the LOD hierarchy. Denoting the quantization parameters for ρ, θ and h as G_ρ, G_θ and G_h, respectively, they are computed as $G_\rho = G_h = \alpha \cdot r/Q, G_\theta = 2\pi/Q$ where α and Q are design parameters specifying the scale of the bounding volume and the number of bins, respectively, for the quantization. We set α and Q to 6 and 64, respectively, in our experiments.

Before encoding, we sort all the K child surfels in the lexicographic order of their quantized θ, ρ and h coordinates. The quantized ρ and h coordinates are directly arithmetic-encoded. For the encoding of the quantized θ coordinates, we as Peng *et al.* [9] make effective predictions to reduce the data entropy. Assuming that the child surfels distribute evenly around the parent, we expect their θ values to distribute evenly in $[0, 2\pi)$. As such, we encode the quantized polar angle, θ_1, for the first child directly, make predictions for the following children's quantized polar angles, $\theta_i, 2 \le i \le K$, and encode the residuals. Specifically, the polar angles of h_1 and h_{i-1} are $\alpha_1 = \theta_1 G_\theta$ and $\alpha_{i-1} = \theta_{i-1} G_\theta$, respectively; we predict θ_i by $\theta'_i = \min[\theta_{i-1} + \frac{2\pi + \alpha_1 - \alpha_{i-1}}{(K-i+2)G_\theta}, \frac{2\pi}{G_\theta}]$; the residual $\theta_i - \theta'_i$ is then arithmetic-coded.

B. Normal Encoder

We employ the normal encoding method proposed by Huang *et al.* as part of their point cloud encoder [4], [15]. The normal data quantization is based on a multi-resolution subdivision of the Gaussian sphere surface, leading to a normal quantization table which is used by both the encoder and the decoder.

Lower(higher) resolutions of the normal quantization are used for coarser (finer) LODs during the encoding for better rate-distortion performance. If the next level in the LOD hierarchy uses the same normal resolution, the child surfels simply inherit normal from the parent without any encoding; otherwise, a finer normal resolution is used for the next LOD and the child surfels' normals need to be encoded. The difference between the quantized normals of each child surfel and the parent surfel is encoded through a local normal indexing scheme. That is, the quantized normals in a local neighborhood centered on the parent surfel's normal on the Gaussian sphere are organized into a 1D list, with smaller indices given to normals closer to the center. One bit is arithmetic-encoded to indicate whether the child surfel's normal is found in this 1D list. If it is, its local index in this list is arithmetic-encoded; otherwise, its index in the global normal quantization table is arithmetic-encoded. Since a child surfel's normal is typically close to that of the parent, this local indexing scheme increases the frequencies of small local normal indices, leading to significantly reduced entropy of the normal data.

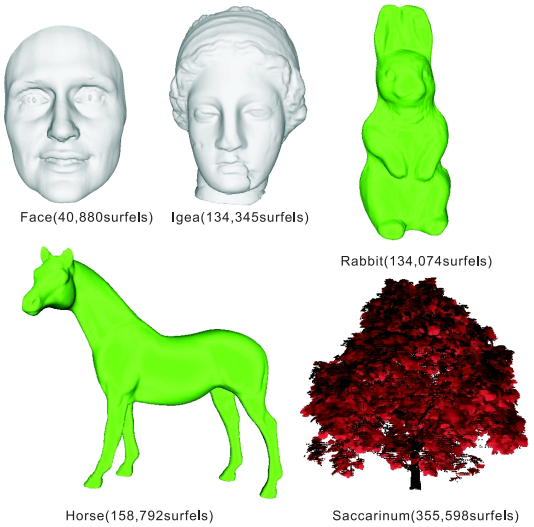

Fig. 1. Test models used in our experiments.

VI. Experiments

In our experiments, we use five test models, which are Face, Igea, Rabbit, Horse and Saccarinum, as shown in Fig. 1. The name and number of surfels of each model are marked in this figure. Note that Face, Igea, Rabbit and Horse correspond to manifold surfaces with or without boundary, while Saccarinum corresponds to a surface with highly complicated topology.

We compare with the octree-based point cloud encoder proposed by Huang *et al.* [15] which is the state-of-the-art generic progressive point cloud encoder that can compress models of arbitrary topology. For each encoder, we decode the test models at a sequence of bitrates, and compare the quality of the reconstructed LODs at the same bitrates between these two point cloud encoders. The comparison is made both quantitatively and visually.

We use the peak signal-to-noise ratio (PSNR) to measure the quality of an intermediate LOD. Specifically, we compute the PSNR for the position and the normal, respectively, according

to the following equations:

$$PSNR_p = 20 \times \log_{10} \frac{D}{E_p} \quad (2)$$

$$PSNR_n = 20 \times \log_{10} \frac{\pi}{E_n} \quad (3)$$

In Equations 2 and 3, D is the diagonal length of the original model's bounding box; E_n is the average angle between the normal of each point in the original model and that of its representative in the intermediate LOD; E_p is the average distance from each point in the original model to the tangential plane at its representative in the intermediate LOD.

A. Quantitative Comparison

We decode each test model at a sequence of bitrates using the octree partitioning based point cloud coder by Huang *et al.* [15] and our point cloud coder, respectively. For each decoded intermediate model, we measure the position and the normal PSNR values, which are reported in Table I. The bitrates are reported in the unit of bits per point (bpp) with respect to the total number of points in each original test model. From Table I, we observe that, for the four manifold models (Face, Horse, Igea and Rabbit), our coder yields higher position coding PSNR values than and comparable normal coding PSNR values to Huang *et al.*'s coder [15] at lower bitrates; our coder yields lower PSNR values when bitrates increase beyond a certain point. For the Saccarinum model with highly complicated topology, however, our coder always yields smaller PSNR values for both the position and the normal coding.

Based on the data reported in Table I, we conclude that, compared with Huang *et al.*'s coder [15], ours is superior at low bitrates for relatively smooth and manifold models but not not as good for models with high geometrical and/or topological complexity.

TABLE I
POSITION AND NORMAL CODING PSNR VALUES WITH HUANG *et al.*'S CODER [15] AND OUR CODER ON THE TEST MODELS.

Bitrate(bpp)		0.7	2.0	4.0	5.0	7.0
$PSNR_p$	our coder	51.73	53.74	58.79	59.28	60.63
	Huang *et al.* [15]	47.99	53.34	59.15	59.91	61.33
$PSNR_n$	our coder	22.33	24.08	30.42	30.45	30.45
	Huang *et al.* [15]	21.74	28.08	28.37	28.72	30.81

(a) Face

Bitrate(bpp)		0.2	2.0	3.0	4.0	5.0
$PSNR_p$	our coder	50.71	59.87	64.21	64.42	64.67
	Huang *et al.* [15]	47.41	59.25	60.32	61.78	63.90
$PSNR_n$	our coder	17.23	25.90	26.01	26.67	27.26
	Huang *et al.* [15]	17.37	25.68	27.00	29.47	31.81

(b) Horse

Bitrate(bpp)		0.2	0.5	3.0	4.0	6.0
$PSNR_p$	our coder	48.80	50.20	61.89	62.15	62.76
	Huang *et al.* [15]	46.51	48.30	59.10	59.79	61.93
$PSNR_n$	our coder	20.39	22.66	26.88	27.23	28.60
	Huang *et al.* [15]	21.20	23.92	28.51	29.96	32.52

(c) Igea

Bitrate(bpp)		0.2	0.3	2.0	3.0	4.0
$PSNR_p$	our coder	49.99	50.42	59.07	62.76	63.01
	Huang *et al.* [15]	47.14	48.19	59.11	60.43	61.99
$PSNR_n$	our coder	21.03	21.66	28.62	28.75	29.16
	Huang *et al.* [15]	21.21	22.08	28.62	29.85	32.17

(d) Rabbit

Bitrate(bpp)		0.5	1.0	4.0	6.0	8.0
$PSNR_p$	our coder	44.25	47.55	50.38	52.51	54.03
	Huang *et al.* [15]	48.77	51.42	59.53	60.44	61.83
$PSNR_n$	our coder	15.89	16.02	18.01	18.79	18.79
	Huang *et al.* [15]	16.42	18.09	20.41	24.49	47.78

(e) Saccarinum

B. Visual Comparison

Selected intermediate LODs at 2bpp and 4bpp for the test models are shown in Fig. 2. The coding method and corresponding bitrate is marked below each model in the figure. Visually comparing the LODs obtained with Huang *et al.*'s coder [15] and our coder, we find that our coder leads to better model quality for the manifold models (Face, Igea, Rabbit and Horse) but worse model quality for the complicate non-manifold model (Saccarinum). This again confirms the superiority of our coder in low bitrate manifold model coding.

VII. CONCLUSION AND FUTURE WORK

In this work, we have proposed a point cloud geometry encoder that is composed of two stages: LOD hierarchy construction and LOD hierarchy encoding. The first stage constructs a hierarchy of LODs using the adapted GLA algorithm, leading to optimized quality of intermediate LODs; the second stage makes progressive encoding of the LOD hierarchy using effective representation, prediction and entropy coding for the positional and the normal information points. Neither stage requires that the surface has to be manifold. As a result, the proposed point cloud encoder can process models of arbitrary topology, and yields better reconstructed model quality at low bitrates, making it potentially suitable for applications with low-end bandwidth and/or platform configurations.

In the future, we plan to design algorithms to efficiently encode the color information as well; Further, we will investigate how to optimally allocate the coding bits over the model surface such that the overall model quality is maximized for a given total bit budget.

ACKNOWLEDGMENT

This work is supported by the Scientific Research Foundation for the Excellent Middle-Aged and Youth Scientists of Shandong Province of China (Grant No. BS2011DX017), the National Natural Science Foundation of China (Grants No. 61070103 and No. U1035004), the Program for New Century Excellent Talents in University (NCET) in China, and Shandong Provincial Natural Science Foundation, China (Grant No. ZR2011FZ004).

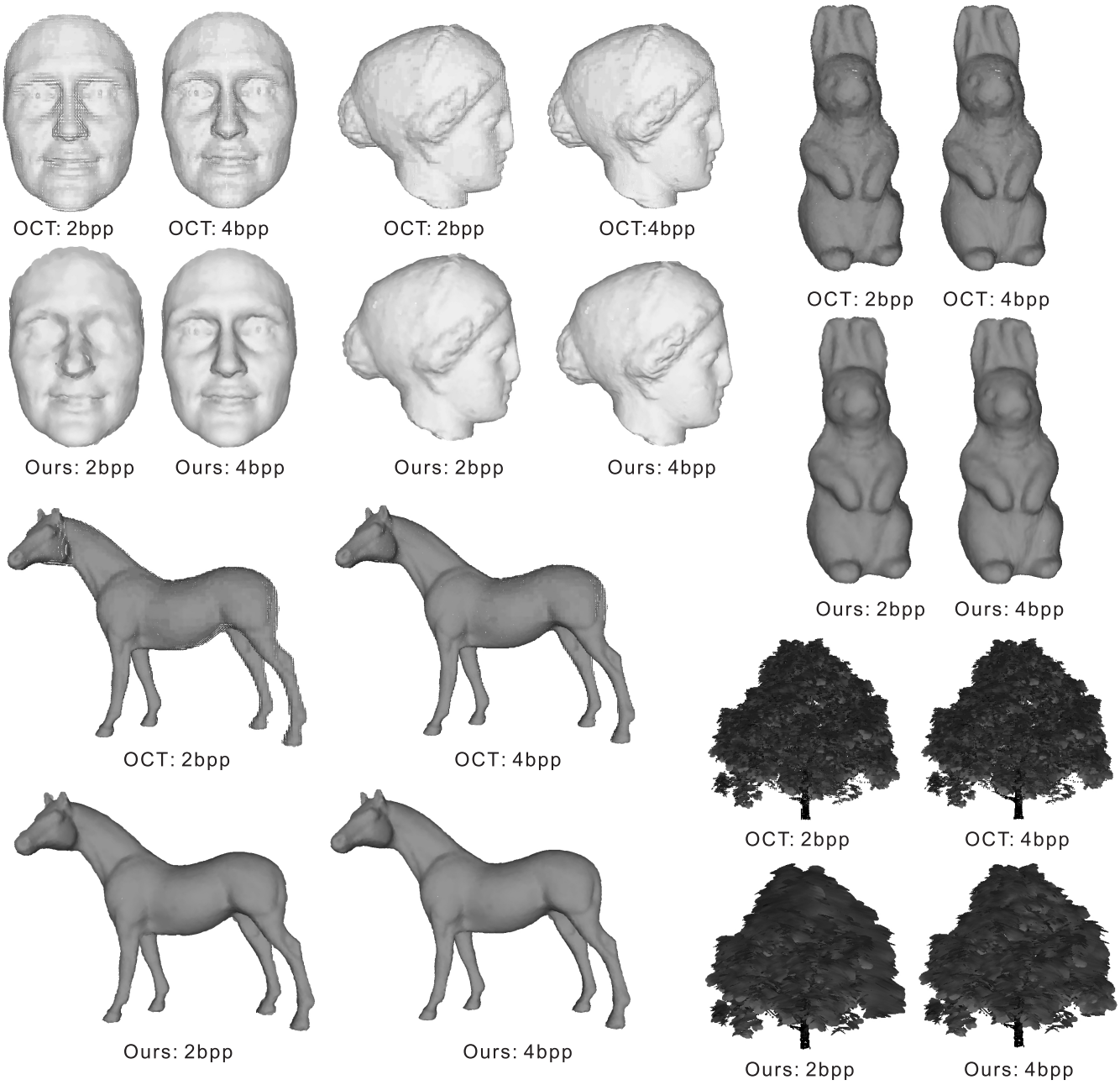

Fig. 2. Visual comparison between Huang *et al.* [15] (OCT) and our method (Ours) at low bitrates (2bpp, 4bpp) on the test models.

REFERENCES

[1] BOTSCH M., WIRATANAYA A., KOBBELT L.: Efficient high quality rendering of point sampled geometry. In *EGRW '02: Proceedings of the 13th Eurographics workshop on Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2002), Eurographics Association, pp. 53–64.

[2] FLEISHMAN S., COHEN-OR D., ALEXA M., SILVA C. T.: Progressive point set surfaces. *ACM Trans. Graph. 22*, 4 (2003), 997–1011.

[3] GUMHOLD S., KARNI Z., ISENBURG M., SEIDEL H.-P.: Predictive point-cloud compression. In *Siggraph Sketches* (2005).

[4] HUANG Y., PENG J., KUO C.-C. J., GOPI M.: Octree-based progressive geometry coding of point clouds. In *Eurographics Symposium on Point-Based Graphics* (2006), pp. 103–110.

[5] KRÜGER J., SCHNEIDER J., WESTERMANN R.: Duodecim - a structure for point scan compression and rendering. In *Eurographics Symposium on Point-Based Graphics* (2005), pp. 99–107.

[6] KALAIAH A., VARSHNEY A.: Statistical geometry representation for efficient transmission and rendering. *ACM Transactions on Graphics 24*, 2 (2005), 348–373.

[7] LEVOY M., WHITTED T.: *The use of points as a display primitive*. Technical report 85-022, Department of Computer Science, University of North Carolina, 1985.

[8] OCHOTTA T., SAUPE D.: Compression of point-based 3d models by shape-adaptive wavelet coding of multi-height fields. In *Eurographics Symposium on Point-Based Graphics* (2004), pp. 103–112.

[9] PENG J., HUANG Y., KUO C.-C. J., ECKSTEIN I., GOPI M.: Feature oriented progressive lossless mesh coding. In *Computer Graphics Forum* (2010), vol. 29, Wiley Online Library, pp. 2029–2038.

[10] PENG J., KUO C.-C. J.: Geometry-guided progressive lossless 3D mesh coding with octree (OT) decomposition. In *ACM SIGGRAPH* (2005), pp. 609–616.

[11] RUSINKIEWICZ S., LEVOY M.: Qsplat: A multiresolution point rendering system for large meshes. In *ACM SIGGRAPH* (2000), pp. 343–352.

[12] SCHNABEL R., KLEIN R.: Octree-based point cloud compression. In *Eurographics Symposium on Point-Based Graphics* (2006), pp. 111–120.

[13] WASCHBÜSCH M., GROSS M., EBERHARD F., LAMBORAY E., WÜRMLIN S.: Progressive compression of point-sampled models. In *Eurographics Symposium on Point-Based Graphics* (2004).

[14] WU J., ZHANG Z., KOBBELT L.: Progressive splatting. In *Eurographics Symposium on Point-Based Graphics* (2005), pp. 25–32.

[15] HUANG Y., PENG J., KUO C.-C. J., GOPI M.: A generic scheme for progressive point cloud coding. In *Visualization and Computer Graphics, IEEE Transactions on 14*, 2 (2008), 440–453.

[16] OCHOTTA T., SAUPE D.: Image-based surface compression. In *Computer graphics forum 27*, 6 (2008), 1647–1663.