Eurographics Symposium on Point-Based Graphics (2006)
M. Botsch, B. Chen (Editors)

Octree-Based Progressive Geometry Coding of Point Clouds

Yan Huang[†1] Jingliang Peng[‡2] C.-C. Jay Kuo[§2] M. Gopi[¶1]

[1]University of California, Irvine
[2]University of Southern California

Abstract

We propose a generic point cloud encoder that compresses geometry data including positions and normals of point samples corresponding to 3D objects with arbitrary topology. In this work, the coding process is led by an iterative octree cell subdivision of the object space. At each level of subdivision, positions of point samples are approximated by the geometry centers of all tree-front cells while normals are approximated by their statistical average within each of the tree-front cells. With this framework, we employ attribute-dependent encoding techniques to exploit different characteristics of various attributes. As a result, significant improvement in the rate-distortion (R-D) performance has been obtained with respect to the prior art. Furthermore, the proposed point cloud encoder can be potentially used for lossless geometry coding of 3D point clouds, given sufficient levels of octree expansion and normal space partitioning.

Categories and Subject Descriptors (according to ACM CCS): E.4 [Coding and Information Theory]: Data Compaction and Compression;

1. Introduction

3D models find applications in emerging fields such as gaming, animation and scientific visualization nowadays. With the increasing capability of 3D data acquisition devices and computing machines, it is relatively easy to produce digitized 3D models with millions of points. The increase in both availability and complexity of 3D digital models makes it critical to efficiently compress the data so that they can be stored, transmitted, processed and rendered efficiently.

Traditionally, polygonal meshes (and triangular meshes in particular) have been widely used to represent 3D objects. In a polygonal mesh, both its geometry and topology data have to be specified. In recent years, point-based 3D model representation gains more popularity with several advantages. For example, the triangulation overhead is saved, processing and rendering are facilitated without the connectivity constraint, and objects of complex topology can be more easily represented. They make the point based representation an ideal choice in many applications that use high quality 3D models consisting of millions of points. With such a huge amount of data, efficient compression becomes extremely important.

In this work, we propose an efficient progressive coder for point clouds, which processes point samples of 3D objects with arbitrary topology and encodes geometry attributes in a unified framework. We present novel contributions in position and normal coding. They include utilization of local data correlation in position prediction and novel quantization and data reorganization in normal coding. Both of them lead to greatly reduced entropy values and, as a result, significant improvement over the prior art in coding efficiency

1.1. Related Work

Mesh Compression: The problem of 3D mesh compression has been extensively studied for more than a decade. In general, there have been two lines of research: single-rate and progressive mesh compression. Compared with single-rate mesh coders, progressive ones allow a mesh to be transmitted and reconstructed in multiple levels of

† e-mail: yanh@ics.uci.edu
‡ e-mail: jingliap@usc.edu
§ e-mail: cckuo@sipi.usc.edu
¶ e-mail: gopi@ics.uci.edu

detail (LODs), which is suitable for streaming in networked applications. Most single-rate mesh coders [TR98, BPZ99b, TG98, AD01b, GS98, Ros99] and early progressive mesh coders [Hop96, COLR99, AD01a, LK98, BPZ99a] are connectivity-driven. That is, connectivity coding is given a higher priority, which drives the whole coding process and the geometry coding. Since geometry information is often visually important and it demands more bits to represent, it is natural to develop geometry-driven progressive mesh coders [GD02, PK04, PK05, KG00a, KSS00, KG00b]. In general, connectivity-driven coders can only treat manifolds while some geometry-driven coders [GD02, PK04, PK05] can handle meshes with arbitrary topology. For a comprehensive survey of 3D mesh coding techniques, readers are referred to [PKK05].

Point-based Model Compression: Similar to techniques for mesh coding, most coders for point-based models can be classified into single-rate coders [GKIS05] and progressive coders [BWK02, FCOAS03, OS04, WGE*04, WZK05, KV05]. The work in [KSW05] is unique in the sense that, although it encodes an input model into multiple LODs, the bitstream of a coarser LOD is not embedded in that of a finer one. Thus, we do not classify it as a progressive coder. Furthermore, some point-based model coders only deal with samples from manifold objects [FCOAS03, OS04] while others can handle samples from arbitrary 3D objects [BWK02, WGE*04, GKIS05, WZK05, KSW05, KV05].

A prediction tree was built up for each input model in [GKIS05] to facilitate prediction and entropy coding, which is however not suitable for progressive coding. A multilevel point-based representation was adopted in [FCOAS03], where the coefficient dimension is reduced from 3D to 1D for higher coding efficiency. Techniques of 3D model partitioning and height field conversion were introduced in [OS04] so that the 2D wavelet technique can be used to encode the 3D data. The coders proposed in [FCOAS03] and [OS04] can only deal with points from manifold surfaces.

To the best of our knowledge, among all previous coders for point-based models, only coders in [BWK02, WZK05, KSW05, KV05, WGE*04] can handle samples of arbitrary objects. Only position data were encoded in [BWK02] based on iterative octree-based space partitioning. An extended edge collapse operator merged two end-points of a virtual edge into one point in [WZK05]. Multiple Hexagonal Close Packing (HCP) grids with decreasing resolutions were constructed in [KSW05], where sequences of filled cells were extracted and encoded for each HCP grid. Cluster-based hierarchical Principal Component Analysis (PCA) was used in [KV05] to derive an efficient statistical geometry representation. Since research in [BWK02, WZK05, KSW05, KV05] focused on efficient rendering, no R-D data of point cloud compression were reported therein. Iterative point pair contraction was conducted for LOD construction and the reverse process was encoded in [WGE*04], where all point attributes were encoded under a common framework. Although this technique is applicable to samples from non-manifold objects in principle, no such results were presented. Besides, there is a limit on the number of LODs in [WGE*04]. If this constraint is not met, coding efficiency can be significantly degraded.

1.2. Main Contributions

In this work, we propose a novel technique for progressive coding of geometry attributes, including positions and normals, of point samples from 3D objects with arbitrary topology. Our major contributions include the following.

- **Full-range Progressive Coder:** At the decoder side, a model is progressively reconstructed from a single point to the complete complexity of the original model.
- **Generic Coder:** It can compress point data for objects with arbitrary topology.
- **Effective Geometry Prediction:** A novel neighborhood-based prediction technique is used to exploit the data correlation in a local neighborhood, leading to greatly reduced entropy.
- **Efficient Normal Coding:** A novel quantization scheme and a local data reorganization scheme are proposed to exploit the correlation of local normal data effectively.

The rest of this paper is organized as follows. An overview of the proposed algorithm is given in Section 2. The proposed position and normal coders are detailed in Sections 3 and 4, respectively. Experimental results are presented in Section 5, and concluding remarks are given in Section 6.

2. Overview of Proposed Algorithm

Constructing LODs of the Model: The proposed encoder recursively and uniformly subdivides the smallest axis-aligned bounding box of a given model into eight children in the octree data structure. Only the nonempty child cells will be subdivided further. The part of the model within each cell is represented by its cell's attributes – the position of each cell is represented by the geometric center of the cell, and the normal of each cell is set to the average of the normals of contained points. A leaf node of the octree consisting of one point has the same normal as the original point in the model, but its position is the grid center of the leaf node. Hence, even a one-point cell can be further subdivided so that the center of a smaller cell approximates the actual geometry as close as possible. The attributes of nonempty cells in each level in the octree structure yield an LOD of the original 3D model. Each point in an LOD is called a representative.

Coding of LODs: The main contribution of the paper lies in the coding of LODs of the model represented by the octree data structure. In association with each octree cell subdivision, we encode the geometry attributes of each nonempty

child cell. The position of each cell is implicit as the subdivision of a cell is uniform and the center of the cell can be computed from the position of the parent cell. Nevertheless, the sequence of non-empty child cells has to be encoded efficiently, as detailed in Section 3. The second attribute, *i.e.*, the normal, is first quantized based on uniform subdivision of the unit sphere. For each cell subdivision, all child normals are predicted by the normal of their parent. Then, their residuals are encoded. On the unit sphere, quantized normals around the predicted normal are locally sorted and indexed, resulting in a reduced entropy of normal residual indices. The coding of normals is described in Section 4.

3. Position Coder

Our main contribution in position coding is a technique to lower the entropy of codes representing nonempty children by a neighborhood-based predictor. For each octree cell subdivision, the point representing the parent cell is replaced by points representing the nonempty child cells. The decoder needs to know which child cells are nonempty so that a representative can be placed at the geometry center of each nonempty child cell, leading to a finer approximation to the original point cloud model.

Occupancy Code: In our position coder, a 1-bit flag is used to signify whether a child cell is empty or not, with '1' indicating a nonempty child cell and '0' an empty child cell. If we traverse all child cells according to a fixed order, and collect the flag bits of all child cells, we will obtain an 8-bit code, which we call the *occupancy code* that has to be compressed. For the ease of illustration, we consider a 2-D example and show the quadtree subdivision and its occupancy code in Figure 1. If we traverse child cells according to the fixed order, we will obtain two occupancy codes, 1010 and 0101, for two cell subdivisions in Figure 1(a), respectively.

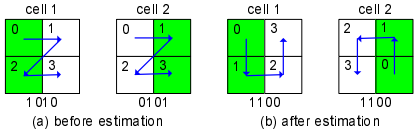


Figure 1: *Examples of occupancy code formation (a) before and (b) after estimation of each child cell's relative probability of being nonempty, where nonempty and empty child cells are colored green and white, respectively. The traversal orders are denoted by the blue arrows.*

Although both cell subdivisions lead to the same number of nonempty child cells, their occupancy codes are quite different. If we can estimate, with high accuracy, the relative probability of each child cell's being nonempty, and traverse child cells with higher estimated probability earlier, we can "push" '1'-bits toward one end of the occupancy code as shown in Figure 1. As a result, the entropy of occupancy codes will be reduced so as to improve coding efficiency.

It is worthwhile to point out that the technique of octree cell subdivision was also used in [PK04] and [PK05] for mesh compression. In [PK05], the index of each nonempty-child-cell tuple, instead of the occupancy code, is encoded. Despite its high coding efficiency, the process of pseudo-probability estimation and tuple sorting is complex, and may not be applicable to real-time decoding on low-end computing devices. A bit-reordering technique was used in [PK04] to reduce the entropy by bit reordering to compress triangular meshes. As compared to the reordering technique in [PK04], the occupancy code reordering method proposed above differs extensively in local neighborhood identification and probability assignment as detailed below.

Occupancy Code Reordering: The probability of a child to be empty or non-empty, which determines the occupancy code, is computed using the neighborhood information of the parent cell. Based on this probability, the occupancy code is reordered and encoded. Thus, this process of occupancy code entropy reduction has three steps: neighborhood identification, probability assignment, and bit reordering according to the relative probability values.

Neighborhood Identification: For 3D meshes, an edge between two vertices indicates the neighbor relationship, which was utilized in [PK04] for bit reordering. Since we do not have edges in a point-based 3D model, we call two representatives c_1 and c_2 in the current LOD (and the corresponding octree cells, C_1 and C_2) neighbors if and only if the following conditions are satisfied.

- The difference of the level numbers of C_1 and C_2 is less than a predetermined threshold α.
- The distance between c_1 and c_2 is less than $\delta \times \min(diag(C_1), diag(C_2))$, where δ is a constant and $diag(C_i)$ is the diagonal length of cell C_i.

The first condition requires at most α continuous octree levels, instead of the whole octree, to be maintained during the process of compression. This allows a memory-efficient implementation of the encoder and the decoder. The second condition guarantees that only nearby representatives (cells) could be neighbors, and the range of local neighborhood is controlled by parameter δ. Interestingly, a similar condition was used in [GKS00] for neighborhood identification of points for surface reconstruction. In our experiments, we choose $\alpha = 3$ and $\delta = 1.5$. Note that there are data structure and computational geometry algorithms [BKOS98] to determine immediate neighbors of a cell in both complete and incomplete octrees. However, these algorithms are not directly applicable to our case since we would like to control the extent of the neighborhood using spacial relationships.

To determine the neighbors of a cell after subdivision, we first construct a list of candidate neighbors of the target cell by inheriting the neighborhood relationship from its parent and including all children of the parent's siblings that have been compressed till now. We prune cells in this list that do not satisfy the two distance criteria as stated above.

Probability Assignment: In general, points are roughly distributed over the surface in a uniform manner, and points within each cell tend to be close to the local tangent plane given by the average normal of all contained points. Based on the observations, for an octree cell, given its local neighborhood and the average normal at the geometry center (to be detailed in Section 4), we can estimate the probability of child cells' being nonempty with the following steps.

- We denote the geometry center of the octree cell by o, the average normal at o by n, and the approximating plane passing through o with normal n by p.
- Calculate the absolute distance d_i, $i = 1,2,\ldots,K$, from each neighbor representative nb_i to p.
- For either side of plane p, sum up the distances of all neighbor representatives on that side to p, resulting in two distance sums, denoted by S_1 and S_2, respectively. Without loss of generality, we assume $S_1 > S_2$.
- Higher relative probability values are assigned to child cells on the side of p corresponding to S_1 than those on the other side.
- For child cells on the same side of p, the closer a child cell is to plane p, the higher its probability value will be.

Note that, for the sake of computational efficiency, we use a plane instead of a higher order surface patch to approximate the local surface.

Being different from the probability assignment in [PK04] which orders child cells purely based on their distances to a local surface approximation, our algorithm first prioritizes all child cells on one side of plane p over those on the other. In general, the plane-side-based priority assignment has led to additional coding gain in experiments. The reason is that p provides a reasonable approximation to the local tangent plane, and points in a local neighborhood tend to lie on one side of the tangent plane, which is often the case except for saddle and other complex surfaces.

Bit Re-ordering: It is not the exact probability values but their relative magnitudes that matter in the proposed algorithm. They guide the child cell traversal and the order of corresponding bits in the occupancy code. For the quadtree cell subdivision example in Figure 2, since the neighbor representatives on the lower side of plane p have a larger distance sum, child cells C_2 and C_3 are assigned higher relative probability values than C_0 and C_1. Furthermore, since C_3 is closer to p than C_2, C_3 is assigned a higher relative probability value than C_2. Similarly, C_0 is assigned a higher relative probability value than C_1. Let P_i be the relative probability value of C_i with $i \in \{0,1,2,3\}$, we have $P_3 > P_2 > P_0 > P_1$, which determines the order of child cell traversal as illustrated by red arrows in the figure. Accordingly, the resultant occupancy code is 1100, with '1's pushed to the left side. Note that this probability estimation algorithm takes into account the local geometry of point-based 3D models implicitly, and it works well for different local curvatures including regions of maxima and minima.

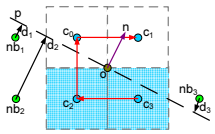
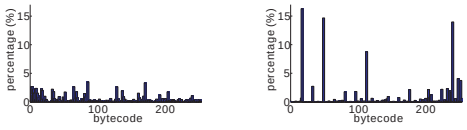


Figure 2: *Determination of the child cell traversal order based on the estimated relative probability values, where nonempty child cells are filled and the order of child cell traversal is shown by red arrows.*

Effects of Bit Reordering: With the proposed bit reordering technique, the entropy of occupancy codes can be greatly reduced, especially at coarse octree levels. This is because cell sizes are comparable to distances between nearby representatives at coarse octree levels. However, cell sizes are small when compared to distances between nearby representatives at finer octree levels and, as a result, probability estimation is less accurate.

To demonstrate the effectiveness of the relative probability estimation and the bit re-ordering techniques, we show the distributions of occupancy codes before and after bit reordering, respectively, in Figure 3(a) and Figure 3(b) based on the accumulative statistics for the octopus model with eight-level octree subdivision. We see from Figure 3 that, after bit re-ordering, occupancy codes are concentrated on several values with high peaks, leading to a reduced entropy value. The entropies of occupancy codes are 6.95 and 4.58 before and after the bit re-ordering, respectively. The same procedure of estimating child cells' relative probability of being nonempty is carried out by both the encoder and the decoder based on the information from a local neighborhood of the parent representative.



(a) (b)

Figure 3: *Distribution of occupancy codes (a) before and (b) after bit re-ordering.*

4. Normal Coder

Our main contribution in normal coding is to rearrange the normal data based on their differences using a novel local normal indexing scheme so that the entropy of normal data is greatly reduced.

Overall Structure: In any intermediate LOD, the normal of a representative is the normalized average of normals of all points contained in the corresponding octree cell. In essence, the normalization process reduces the dimension of

normal data from 3D to 2D, with each representative normal corresponding to a point on the unit sphere. For each cell subdivision, all non-empty child cells are predicted to have the same normal as their parent, and the prediction residuals are coded using a local normal indexing scheme that organizes similar normals around the predicted one on the unit sphere into a 1D list. In the following, for the ease of description, we use the cell normal and the representative normal interchangeably.

Normal quantization: Before compression, normals need to be quantized. For this purpose, the whole normal space (*i.e.* the unit Gauss sphere) is iteratively subdivided to a pre-determined resolution [THLR98, BWK02], and a normal table is constructed accordingly. Afterwards, each input unit normal can be approximated by one quantized normal that can be uniquely identified by an index into the normal table. As to subdivision and indexing of the normal space, we use the approach proposed in [BWK02]. Initially, we inscribe a regular octahedron into the unit sphere, and the eight triangular facets of the octahedron form the first level of partition. Every subsequent level of partition is obtained by 1-to-4 subdivision of all triangles in the previous level and projecting the newly inserted vertices onto the unit-sphere. All normal vectors in the direction of any point inside a triangle on the unit sphere will be approximated by the unit normal of that triangle. The normal table consists of entries that have vertices and normals of triangles in the subdivision. The iterative process of normal space partitioning is illustrated in Figure 4, where we use $(i:j)$ to index the j-th partition at the i-th level. In the i-th level, there are 8×4^i partitions in total and $3+2i$ bits are needed to identify each of them.

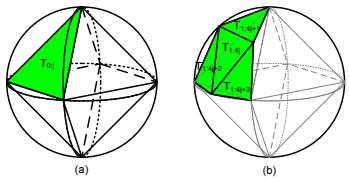


Figure 4: *Normal quantization: (a) an octahedron is inscribed into a unit sphere, and its eight facets, $T_{0:j}$ $(j=0,1,\ldots,7)$ form the first level of partition; and (b) triangle $T_{0:j}$ $(j\in\{0,1,\ldots,7\})$ is subdivided into four sub-triangles, $T_{1:4j}\sim T_{1:4j+3}$, with index $(1:4j)$ assigned to the central sub-triangle whose normal is equal to that of $T_{0:j}$.*

As described in Section 3, the position resolution is progressively increased with the octree subdivision level increases. In terms of the R-D performance, it is not meaningful to encode the normal information in high resolution at coarser octree levels since the positional resolution is still low. Thus, we build up multiple normal tables, one for each level of normal space partitioning, and associate an appropriate resolution-level normal table with each octree level. This is different from [BWK02] where, for the purpose of efficient rendering, only one global normal table is built up according to one pre-determined resolution of normal quantization. Furthermore, we do not increase the resolution of normal quantization for each octree level but only for every other octree level.

The effect of normal precision on visual quality decreases when the density of representatives over a surface increases. Based on this observation, a maximum of 13 bits (corresponding to 6 levels of sphere partitioning) are used for the normal quantization once LODs go beyond a certain level of octree subdivision. For example, cell normals at the 0th and 1st octree levels are quantized by 3 bits; cell normals at the 2nd and 3rd octree levels are quantized by 5 bits, and so on. The maximum normal quantization resolution of 13 bits is used for all levels equal or greater than the 10th.

Since we only increase the normal resolution at odd octree levels, when an even-level octree cell is subdivided, its normal is simply inherited by all nonempty child cells without coding. When an odd-level octree cell is subdivided, we need to encode the normal of each nonempty child cell. In most cases, the normal of a nonempty child cell falls within a small spatial range around that of its parent, and the normal difference between the parent and the child has a smaller magnitude than that of the child cell's absolute normal. Therefore, for each nonempty child cell, we predict its normal to be the same as that of the parent and encode the residual instead for coding efficiency.

Local normal indexing: To represent the normal residual, we propose a local normal indexing scheme. For each triangular facet $T_{i:4j}$ at the i-th level of normal space partitioning, we re-index the same-level facets in a local neighborhood of $T_{i:4j}$ on the unit sphere based on their differences in normal from $T_{i:4j}$ and maintain an array, $A_{i,4j}$, of pointers to the facets in the local neighborhood as shown in Figure 5. Technically speaking, we can further expand the local neighborhood. However, three neighbor rings have already produced good performance according to our experiments. Note that not each $T_{i:4j}$ has a neighborhood of size 52; triangles at low quantization resolutions may have fewer neighbors. The same normal space partitioning and local normal indexing is pre-computed by the encoder and the decoder before the actual encoding/decoding process.

In the following, we denote the quantized normal corresponding to $T_{i:j}$ as $\mathbf{n}_{i:j}$. For the normal of the root cell, 3 bits are used to directly encode its global normal index. When a cell in an odd octree level with quantized normal $\mathbf{n}_{i:j}$ is subdivided, we first predict all its nonempty child cells to have the same normal, $\mathbf{n}_{i:4j}$ (=$\mathbf{n}_{i:j}$). Then, for each nonempty child cell, we search the above-constructed local neighborhood for a match of normal. A 1-bit flag is arithmetic encoded to indicate whether the local search is successful. If it is, the local index of the matching normal is arithmetic encoded; otherwise, a global search is conducted and the global normal index is arithmetic encoded.

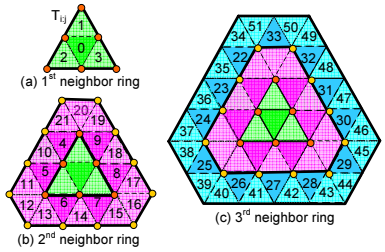


Figure 5: *Local normal indexing: (a) $T_{i:4j} \sim T_{i:4j+3}$ are assigned the smallest four indices since they have the smallest difference in normal from $T_{i:4j}$. $T_{i:4j+1} \sim T_{i:4j+3}$ form the 1^{st} neighbor ring of $T_{i:4j}$; (b) the 1^{st} neighbor ring of $T_{i:4j}$ is expanded and the 2^{nd} neighbor ring (in purple and pink) is formed by the triangular facets around the 1^{st} neighbor ring. Note that the purple facets are assigned smaller indices than the pink ones since their normals are closer to that of $T_{i:4j}$ than those of the pink facets; (c) the local neighborhood is expanded to the 3^{rd} neighbor ring similarly.*

In essence, the proposed local normal indexing scheme increases the occurrence frequencies of local normal indices (from 0 to 51 in our implementation), resulting in a reduced entropy of the normal data. Figure 6 demonstrates the effectiveness of the local normal indexing scheme. By comparing Figure 6(a) and Figure 6(b), we see a much more concentrated distribution around a small number of local normal indices in Figure 6(b).

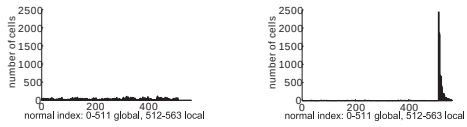


Figure 6: *Distribution of normal indices: (a) the distribution of global normal indices at 9-bit quantization for the Igea model; and (b) the corresponding distribution of normal indices after local normal indexing. Note that the local normal indices are offset by 512 for the purpose of plotting.*

5. Experimental Results

Five point-based models are tested in our experiments. They are the igea, the dragon and the octopus models by the courtesy of Pointshop 3D, the model of acer_saccarinum from Xfrog public plants (http://web.inf.tu-dresden.de/ST2/cg/downloads/publicplants/), and the model of happy buddha (vripped reconstruction) from the Stanford 3D scanning repository (http://graphics.stanford.edu/data/3Dscanrep/). Note that the models of acer_saccarinum and happy buddha have been re-sampled and transferred to the Surfel format by us.

The coding cost is measured in terms of bits per point (bpp) with respect to the total number of points in the original model. As for the distortion metric, the MLS surface was used in [PGMK02, FCOAS03] to compare the difference between two point-based models. However, we do not use the MLS-surface-based distortion metric since it is not directly suitable for measuring the normal distortion and, as argued in [WGE*04], it is unable to measure the sampling of a model, which is crucial for high-quality point-based rendering. Instead, we use the peak-signal-to-noise ratio (PSNR) to measure the position and the normal distortions, respectively, as done in [WGE*04] for fair comparison. The position PSNR is calculated using Euclidean distances between corresponding points in the original and reconstructed models, with the peak signal given by the diagonal length of the original model's smallest axis-aligned bounding box. The normal PSNR is calculated using angles between original and reconstructed normals with a peak signal of 180 degrees.

We compare the R-D performance of the proposed point cloud coder with those proposed in [BWK02, WGE*04] which are the most related prior art in the sense that they can progressively encode point samples of 3D objects with arbitrary topology. The other coders, as stated in Section 1, are single-rate coders, may not apply to samples from non-manifold objects, and/or provide no R-D data on point cloud compression. The coder in [BWK02] only compresses position data while that in [WGE*04] compresses point attributes including position, normal and color.

To compare the R-D performance between our position coder and that in [BWK02], we plot the R-D curves in Figure 7 for the octopus model, where the R-D data for [BWK02] were obtained with our own implementation. We see from Figure 7 that our position coder costs less at any fixed PSNR value. In particular, for any PSNR below 65, the bitrate of our position coder is roughly 33∼50% less than that of [BWK02].

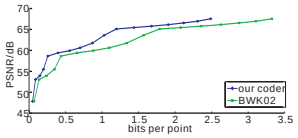
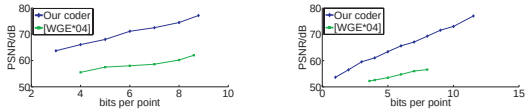


Figure 7: *R-D curves with the proposed position coder and that in [BWK02] for the octopus model.*

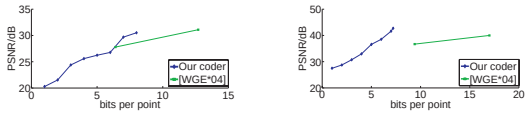
To compare the R-D performance between our progressive coder and that in [WGE*04], we plot the R-D curves for position coding in Figure 8(a) and (b) and the R-D curves for normal coding in Figure 9(a) and (b), respectively, for the dragon and the igea models. In these figures, the horizontal axes and vertical axes give the position/normal coding bitrates and the corresponding PSNR values, respectively. Note that the R-D data of position coding for [WGE*04] are taken from the progressive coding

curves in Figure 10 in [WGE*04] and the R-D data of normal coding for [WGE*04] are taken from the progressive coding results in Table 2 of [WGE*04]. Due to the lack of data, we can not make a full-range comparison especially in the case of normal coding as shown in Figure 9. The curves in Figures 8 and 9 clearly demonstrate the advantage of our coder over [WGE*04] in both position and normal coding. As shown in Figure 8, the PSNR improvement for position coding is around 10dB for the igea model, and around 15dB for the dragon model, at all bitrates. As shown in Figure 9, at certain high PSNR values, the normal coding bitrate with our coder can be about 50% of that in [WGE*04].



(a) (b)

Figure 8: *R-D curves with the proposed position coder and that in [WGE*04] for (a) dragon and (b) igea.*



(a) (b)

Figure 9: *R-D curves with the proposed normal coder and that in [WGE*04] for (a) dragon and (b) igea.*

A comprehensive list of R-D data is provided in Table 1, which provides the R-D data for the position and normal coding with the proposed point cloud coder for the five test models. In this table, the bitrates are reported in bpp, and the distortions in PSNR. In Table 1, the 'N/A' notation signifies unavailable data. This is due to the fact that we only expand the octree up to 11 levels in experiments and the coding gains of the octopus and the happy buddha models are higher than those of others. We see from this table that the R-D performance of normal coding is worse for the acer_saccarinum model than others. This is due to the high randomness in the normals of acer_saccarinum, which is more challeging to encode effectively.

Visual examples of intermediate LODs are shown in Figure 10 for dragon, igea and acer_saccarinum. The same dragon and igea models are used in [WGE*04]. The first four rows show the models reconstructed at total bitrates of 2bpp, 4bpp, 8bpp and 16bpp, respectively, and the last row shows the uncompressed original models. As shown in this figure, a reasonable profile already appears at 2bpp. We can achieve very decent model quality at 8bpp. Finally, the reconstructed models are almost indistinguishable from the original models at 16bpp.

6. Conclusion

A generic point cloud coder was proposed to encode geometry attributes, including position and normal, of points sampled from 3D objects with arbitrary topology in this work. With novel and effective schemes of quantization, prediction and data rearrangement, the proposed point cloud coder results in a significant R-D gain over the prior art. Another advantage of the proposed point cloud coder is that it does not re-sample the input geometry. Thus, it can be potentially used for lossless geometry coding, if the levels of octree expansion and normal space partitioning are sufficiently large.

As an extension of the current work, we will work on the coding of other point attributes such as radius and color. Furthermore, we will conduct experiments on gigantic point clouds with hundreds of millions of points and explore efficient decoding and rendering on GPUs.

7. Acknowledgment

We would like to thank M. Waschbüsch for patiently answering our questions about his work [WGE*04].

References

[AD01a] ALLIEZ P., DESBRUN M.: Progressive encoding for lossless transmission of triangle meshes. In *ACM SIGGRAPH* (2001), pp. 198–205.

[AD01b] ALLIEZ P., DESBRUN M.: Valence-driven connectivity encoding for 3D meshes. In *EUROGRAPHICS* (2001), pp. 480–489.

[BKOS98] BERG M. D., KREVELD M. V., OVERMARS M., SCHWARZKOPF O.: *Computational Geometry: Algorithms and Applications*. Springer, 1998.

[BPZ99a] BAJAJ C., PASCUCCI V., ZHUANG G.: Progressive compression and transmission of arbitrary triangular meshes. In *IEEE Visualization* (1999), pp. 307–316.

[BPZ99b] BAJAJ C. L., PASCUCCI V., ZHUANG G.: Single resolution compression of arbitrary triangular meshes with properties. *Computational Geometry: Theory and Applications 14* (1999), 167–186.

[BWK02] 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.

[COLR99] COHEN-OR D., LEVIN D., REMEZ O.: Progressive compression of arbitrary triangular meshes. In *IEEE Visualization* (1999), pp. 67–72.

[FCOAS03] FLEISHMAN S., COHEN-OR D., ALEXA M., SILVA C. T.: Progressive point set surfaces. *ACM Trans. Graph. 22*, 4 (2003), 997–1011.

Table 1: *R-D data for position and normal coding using the proposed point cloud coder. Bitrates (R) and distortions (D) are reported in bits per point (bpp) and PSNR respectively. For each model, its point number is listed in the parenthesis following the model name. An 'N/A' is used to mark each unavailable data item.*

Total bitrate	Data type	Position					Normal				
		igea (134k)	dragon (436k)	acer_saccarinum (889k)	octopus (466k)	happy buddha (1,088k)	igea (134k)	dragon (436k)	acer_saccarinum (889k)	octopus (466k)	happy buddha (1,088k)
1.0	R	0.41	0.36	0.46	0.19	0.27	0.59	0.64	0.54	0.81	0.73
	D	49.25	53.30	53.02	55.29	53.91	27.08	19.99	12.95	19.22	18.99
2.0	R	0.85	0.59	0.69	0.74	0.44	1.15	1.41	1.31	1.26	1.56
	D	53.38	54.38	53.46	60.89	56.19	27.65	20.78	13.23	22.98	21.37
4.0	R	1.47	1.03	1.16	1.69	1.86	2.53	2.97	2.84	2.31	2.14
	D	54.71	57.88	54.52	65.83	63.69	30.01	24.28	13.70	23.61	24.52
8.0	R	3.34	3.84	2.10	2.89	3.34	4.66	4.16	5.90	5.11	4.66
	D	60.02	65.81	57.99	69.15	66.87	36.12	25.64	14.86	29.10	26.11
12.0	R	6.47	5.33	5.44	6.01	4.68	5.53	6.67	6.56	5.99	7.32
	D	66.25	68.89	65.06	76.58	70.47	37.55	29.07	15.07	33.55	28.68
16.0	R	8.79	8.32	6.77	N/A	N/A	7.21	7.68	9.23	N/A	N/A
	D	71.33	75.36	66.79	N/A	N/A	42.73	30.44	16.44	N/A	N/A

[GD02] GANDOIN P. M., DEVILLERS O.: Progressive lossless compression of arbitrary simplicial complexes. *ACM Trans. Graphics 21*, 3 (2002), 372–379.

[GKIS05] GUMHOLD S., KARNI Z., ISENBURG M., SEIDEL H.-P.: Predictive point-cloud compression. In *Siggraph Sketches* (2005).

[GKS00] GOPI M., KRISHNAN S., SILVA C.: Surface reconstruction using lower dimensional localized delaunay triangulation. *Eurographics 19*, 3 (2000), 467–478.

[GS98] GUMHOLD S., STRASSER W.: Real time compression of triangle mesh connectivity. In *ACM SIGGRAPH* (1998), pp. 133–140.

[Hop96] HOPPE H.: Progressive meshes. In *ACM SIGGRAPH* (1996), pp. 99–108.

[KG00a] KARNI Z., GOTSMAN C.: Spectral compression of mesh geometry. In *ACM SIGGRAPH* (2000), pp. 279–286.

[KG00b] KHODAKOVSKY A., GUSKOV I.: Normal mesh compression. *Preprint* (2000).

[KSS00] KHODAKOVSKY A., SCHRÖDER P., SWELDENS W.: Progressive geometry compression. In *ACM SIGGRAPH* (2000), pp. 271–278.

[KSW05] 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.

[KV05] KALAIAH A., VARSHNEY A.: Statistical geometry representation for efficient transmission and rendering. *ACM Transactions on Graphics 24*, 2 (2005), 348–373.

[LK98] LI J., KUO C.-C. J.: Progressive coding of 3-D graphic models. *Proceeding of the IEEE 86*, 6 (Jun 1998), 1052–1063.

[OS04] 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.

[PGMK02] PAULY M., GROSS M., M., KOBBELT L.: Efficient simplification of point-sampled surfaces. In *Proc. VIS* (2002), pp. 163–170.

[PK04] PENG J., KUO C.-C. J.: Progressive geometry encoder using octree-based space partitioning. In *Proc. of the 2004 IEEE International Conference on Multimedia and Expo, ICME 2004* (2004), pp. 1–4.

[PK05] PENG J., KUO C.-C. J.: Geometry-guided progressive lossless 3D mesh coding with octree (OT) decomposition. In *ACM SIGGRAPH* (2005), pp. 609–616.

[PKK05] PENG J., KIM C.-S., KUO C.-C. J.: Technologies for 3D mesh compression: A survey. *Journal of Visual Communication and Image Representation 16*, 6 (2005), 688–733.

[Ros99] ROSSIGNAC J.: Edgebreaker: Connectivity compression for triangle meshes. *IEEE Trans. Visualization and Computer Graphics 5*, 1 (1999), 47–61.

[TG98] TOUMA C., GOTSMAN C.: Triangle mesh compression. In *Proceedings of Graphics Interface* (1998), pp. 26–34.

[THLR98] TAUBIN G., HORN W., LAZARUS F., ROSSIGNAC J.: Geometry coding and VRML. *Proceeding of the IEEE 96*, 6 (Jun 1998), 1228–1243.

[TR98] TAUBIN G., ROSSIGNAC J.: Geometric compression through topological surgery. *ACM Trans. Graphics 17*, 2 (1998), 84–115.

[WGE*04] WASCHBÜSCH M., GROSS M., EBERHARD F., LAMBORAY E., WÜRMLIN S.: Progressive compression of point-sampled models. In *Eurographics Symposium on Point-Based Graphics* (2004).

[WZK05] WU J., ZHANG Z., KOBBELT L.: Progressive splatting. In *Eurographics Symposium on Point-Based Graphics* (2005), pp. 25–32.