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

Octree-based Point-Cloud Compression

Ruwen Schnabel and Reinhard Klein†

Institut für Informatik II, Universität Bonn, Germany

Abstract

In this paper we present a progressive compression method for point sampled models that is specifically apt at dealing with densely sampled surface geometry. The compression is lossless and therefore is also suitable for storing the unfiltered, raw scan data. Our method is based on an octree decomposition of space. The point-cloud is encoded in terms of occupied octree-cells. To compress the octree we employ novel prediction techniques that were specifically designed for point sampled geometry and are based on local surface approximations to achieve high compression rates that outperform previous progressive coders for point-sampled geometry. Moreover we demonstrate that additional point attributes, such as color, which are of great importance for point-sampled geometry, can be well integrated and efficiently encoded in this framework.

Categories and Subject Descriptors (according to ACM CCS): E.4 [Data]: Coding and information theory - Data compaction and compression; I.3.5 [Computer Graphics]: Computational geometry and object modeling - Curve, surface, solid and object representations

1. Introduction

The increasing amount of geometric information acquired with 3D scanners gives rise to an equally growing demand for data representations that allow for efficient and compact storage as well as transfer of this data. Geometry compression has therefore been an important field in computer graphics for a long time and recently point-sampled geometry received an increasing amount of attention in particular. In general, the output of a 3D scanning session is a dense and relatively regular point sampling of the surface. Some acquisition devices equip the point samples with additional attributes, of which color usually is one of the most important. In this paper we present a progressive compression method for point sampled models that is specifically apt at dealing with densely sampled surface geometry as produced by 3D scanning devices. The compression method presented here is lossless and therefore is also suitable for archiving the unfiltered, raw scan data. The progressiveness of the algorithm renders it suitable for streaming applications, e.g. on the internet.

Our method represents the point-cloud as an octree decomposition of space which we compress using novel prediction techniques that were specifically designed for point sampled geometry to achieve high compression rates, outperforming previous progressive lossless point-cloud encoders and achieving comparable performance during progressive decompression as existing lossy methods. We also demonstrate that additional point attributes, such as color, which are of great importance for point-sampled geometry, can be well integrated and efficiently encoded in this framework.

2. Previous work

Several compression schemes for point-sampled geometry have been proposed. The progressive point set surfaces by [FCOAS03] are based on the MLS surface definition of [ABCO*03]. The input point-set $\mathcal{P}$ is reduced, the resulting base point set is triangulated and then compressed using a mesh compression algorithm. The base point-set is refined by insertion of additional samples in the local neighborhoods of the base points. These new samples are projected on the MLS surface induced by $\mathcal{P}$ and only the difference needs to be encoded, yielding a lossy compression. Unfortunately the method has a tendency to smooth out sharp features as a consequence of the MLS surface approximation. The algorithm of [OS04] makes use of the MLS surface as well. They use a set of planar height fields to resample the surface and encode

† e-mail:{schnabel,rk}@cs.uni-bonn.de

the data using image based techniques. The partitioning of the point-cloud into planar regions is obtained by a method similar to the one given in [PG01]. The method could be called semi-progressive as it uses progressive coding only within the patches. Since this method is based on the MLS surface as well, it also suffers from the smoothing inherent to this approximation, but also may introduce artifacts at the patch boundaries. The MLS surface is also not well suited for compression of noisy scanning point-clouds, as it may break down in regions of too much noise or high irregularity. [WGE*04] create a hierarchy on the point-cloud by joining pairs of similar samples. The similarity measure can be chosen to include, besides spatial proximity, additional attributes, such as color or normals. Pairs are collapsed to their average and only the offset to one of the points has to be encoded for compression and a local coordinate system is used for offset representation, resulting in a progressive coder for geometry and arbitrary point attributes. Although not technically a compression algorithm, one of the earliest methods concerned with efficient representations of point-clouds is the QSplat rendering system of [RL00]. QSplat is based on a hierarchical bounding sphere data structure where each node is quantized to a size of 48 bits, including attribute data such as color and normal information. High quality renderings are obtained despite this strong quantization.

Since many important concepts of geometry compression were introduced in the context of mesh compression and we also build upon some of those ideas we briefly discuss the contributions most relevant to our work, a more thorough survey can be found in [AG03]. For meshes, not only geometry information, but also the mesh connectivity has to be encoded. Mesh compression methods can be loosely separated into two categories: connectivity or geometry centered. Instances of connectivity centered algorithms are the algorithms of [GS98] and [Ros99]. These coders are not progressive and the ideas therein have been transferred to point-clouds with the single rate encoders proposed by [GKIS05] and [MMG06]. A spanning tree is constructed on the points and traversed. Only offsets to consecutive points are encoded and several prediction rules for the geometry are applied [TG98]. In general the compression rates of single rate encoders are higher compared to those of progressive coders, since no intermediary information needs to be transmitted. The geometry centered algorithm of [KG00] is a lossy codec based on a spectral analysis of the mesh. While high compression rates are obtained, unfortunately computing the coefficients is computationally very expensive. A different approach was introduced in [DG00] and [GD02]. While the algorithm was originally intended for mesh compression it is nonetheless directly applicable to lossless point cloud compression. It is based on a binary hierarchical decomposition of space. For each cell subdivision, the number of points in each cell are encoded and only non-empty cells are further subdivided until the desired precision is reached. [PK05] use an octree and, similar to [BWK02], do not encode the number of points in each cell but instead store for each child the fact wether it is occupied or empty. However the approach of [BWK02] is not truly a point-cloud compression algorithm but encodes an implicit surface, i.e. sampling rate and precision are identical. The method of [PK05] on the other hand can not be directly applied to point-sampled geometry as it makes extensive use of connectivity information for prediction of non-empty child cells. [PK05] and [BWK02] both show that an octree decomposition is capable of achieving very good compression rates. This is the motivation for our lossless progressive compression method that also builds on an octree decomposition. However, we suggest novel prediction schemes that are specifically designed for point-sampled surfaces and result in compression rates that are similar to, or even better than, lossless single rate encoders, such as [GKIS05]. Moreover we show how additional attributes can be integrated in the framework.

3. Octree compression

In this section we briefly review the general concept of octree-based geometry compression as introduced by [BWK02] [PK05]. Given the bounding cube of the point-cloud $\mathcal{P}$ that is to be compressed, an octree $\mathcal{O}$ is constructed with a maximum number of levels L and the points in $\mathcal{P}$ are sorted into the cells of the octree. The points in $\mathcal{P}$ are replaced by the cell centers of the octree's leaves, i.e. the points in $\mathcal{P}$ are quantized. The number of levels determines the precision of the coordinate quantization. An octree of depth L gives a precision of L bits per coordinate direction. In accordance with the notion found in the literature, our compression scheme is lossless in the sense that these quantized coordinates are preserved during the compression. As noted in the seminal work of [Dee95], usually a quantization into 16 bits per direction inside the bounding cube suffices to achieve a virtually lossless compression.

In order to compress the quantized point set $\mathcal{P}$ only the octree has to be encoded since $\mathcal{P}$ can then be reconstructed as the cell centers of the leaves of $\mathcal{O}$. The coding of the octree proceeds in a top-down and breadth-first fashion. The root cell can always be assumed to be non-empty. Then, for each cell it has only to be encoded which child cells are non-empty . The decoder can then faithfully recover the octree by following the same traversal rules as the encoder while always constructing those child cells that have been specified as occupied by the encoder. Note that no further information has to be encoded for empty cells.

Therefore the compression of the octree *only* depends on the way the non-empty child cells are encoded. If the occupied child cell configurations can be well predicted, high compression rates can be achieved.

In [BWK02] the existent child cells are specified in a single byte per cell subdivision, i.e. each bit specifies the occupancy of a child cell. This way the octree is encoded as

a sequence of bytes that can be compressed with arithmetic coding [MNW98], exploiting the fact that certain child cell configurations appear more frequently than others. However no prediction is employed and hence the compression is only minimal.

[PK05] on the other hand propose to encode a child cell subdivision in two steps, instead of in a single byte: First the number of non-empty child cells is written. For a given number e of non-empty child cells the number of possible child cell configurations is given by $\binom{8}{e}$ and all these configurations are stored in a look-up table. The cell occupancy is then encoded in a second step as index into this look-up table. This approach is advantageous as it is more amenable to elaborate predictive coding, since the information of the first step can be used to achieve a more accurate prediction in the second step. However, [PK05] do not explicitly predict the number of non-empty child cells in the first step and for the second step use a prediction that heavily depends on explicitly encoded mesh connectivity. For point-clouds though, explicitly encoding connectivity information that, for instance, could be captured in a nearest-neighbors graph, introduces an unnecessary overhead.

Consequently, in our approach, we employ the same two-step encoding, but employ novel prediction techniques for both, the non-empty child count of a cell e and, taking e into account, also the child cell configuration. Our prediction is based solely on the point-sampled geometry and does not require any explicitly encoded connectivity information. This is the key to the high compression rates achieved by our method. The prediction of the number of non-empty child cells is presented in detail in section 4.2 and the prediction of the child cell configurations in section 4.3.

4. Predictive coding

Since the octree is traversed in breadth-first order the centers of the cells on the traversal front at all times provide a coarse approximation $\mathcal{Q}$ of the complete point-cloud $\mathcal{P}$. During traversal $\mathcal{Q}$ is refined progressively as additional child cells are visited (or created in case of the decoder). Indeed the decoder can stop processing at any time and return $\mathcal{Q}$ as a preliminary result of the decompression, e.g. in streaming applications. Moreover, as $\mathcal{Q}$ is an approximation of the original point-cloud and is accessible to both encoder and decoder, we use it to predict the child cell configurations of the cell subdivisions.

4.1. Surface approximation

The prediction is based on an MLS approximation of the surface induced by $\mathcal{Q}$. For a leaf cell $C \in \mathcal{O}$ that is to be subdivided, a planar approximation $F_{\mathcal{Q}}^{C}$ of the true underlying surface in C is obtained. To compute the plane $F_{\mathcal{Q}}^{C}$ we find the k nearest neighbors $N_k = \{q_1 \dots q_k\}$ in $\mathcal{Q}$ to the cell center $c(C)$ of C (note that N_k includes $c(C)$).

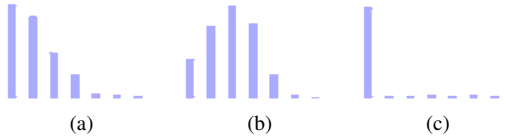


Figure 1: *The quality of the non-empty child count prediction $\hat{e}$. The charts show the absolute difference between $\hat{e}$ and the true number e of occupied cells ranging from 0 to 7, i.e. a difference of zero (leftmost column) is a correct prediction. (a) Shows the distribution on octree level 5, (b) on level 7, and (c) for the entire octree. Note that it is not important for the arithmetic encoder to have a peak at zero as long as there is a distinct peak.*

$F_{\mathcal{Q}}^{C}$ is defined as the plane that best approximates these points in the weighted least-squares sense. The points in N_k are weighted by a Gaussian fall-off around $c(C)$ with $\sigma = \frac{1}{2}\max_{i=1\dots k}\|c(C) - q_i\|$ [ABCO*03] [PGK02]. The intersection of this plane with the child cells and the distance of the child cell centers to the plane can then be used for prediction of the non-empty child cells. For clarity of presentation, we will drop the annotation of $F_{\mathcal{Q}}^{C}$ in unambiguous cases.

4.2. Number of non-empty cells

As stated above, the subdivision of C is encoded in two steps. First we output the number e of non-empty child cells. The prediction of e is based on an estimate of the sampling density ρ of the original point-cloud $\mathcal{P}$. Since ρ is in general unknown it is estimated in a preprocessing step by finding the k nearest neighbors N_k^i to every point $p_i \in \mathcal{P}$ and estimating a local sampling density by

$$\rho_i = \frac{k}{(\max_{j=1\dots k}\|p_i - q_j\|)^2\pi} .$$

Then ρ is given by

$$\rho = \frac{\sum_{i=1\dots|\mathcal{P}|}\rho_i}{|\mathcal{P}|} .$$

The number of occupied child cells of C is related to the area of the plane F inside C. The larger the area, the more points can be expected to be contained in C. Also, if the area of intersection of a certain number $\hat{e}$ of child cells of C already is large enough to account for almost all the expected points, the number of non-empty child cells will probably be $\hat{e}$. Hence, to predict e, we compute the areas A_i of intersection of F with the child cells of C, $T_1, \dots, T_8$. Given the estimate ρ of the sampling density and the total area of intersection in C, $A = \sum_{i=1}^{8} A_i$, the expected number S_C of points contained in C can be determined by $S_C = \max(1, \lfloor(\rho A) + .5\rfloor)$. To find the prediction $\hat{e}$ of e, the A_i are sorted in descending order and then $\hat{e}$ is given as the

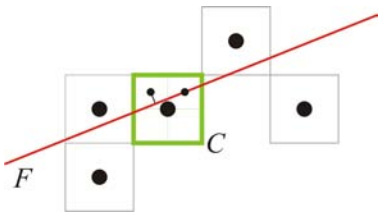


Figure 2: *A cell is to be subdivided. Those child cells that are closest to the surface approximation F_Q^C are more likely to be occupied.*

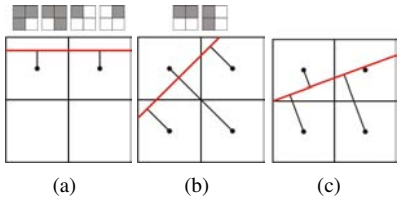


Figure 3: *Different alignments of the plane yield different expressiveness. In a), if there are two non-empty child cells this gives a very useful prediction. In the case of one or three occupied cells however, the expressiveness of the plane is less distinct as several configurations receive the same weight. Ambiguous cases are depicted on top. In b) three and in c) all four child cells have a distinct distance to the plane. b) has a high expressiveness for all but two non-empty child cells. c) is expressive for all configurations.*

smallest i such that $\lfloor(\rho\sum_{j=1\ldots i}A_j)+.5\rfloor \geq S_C$. This way child-cells that, although intersected by the plane, only contribute a marginal amount to the overall area of intersection are not counted as occupied, especially if only a corner of the cell is "cut-off" by the plane (for an example see in figure 2 the lower left child cell). We encode e using adaptive arithmetic coding under the contexts of the octree level of C, $\hat{e}$ and S_C. Using S_C as additional context is helpful as it provides a measure for the quality of the prediction, which improves for smaller values of S_C. As we found that a differentiation between larger values of S_C did not improve the compression rate, for the context of the arithmetic coder, we truncate S_C at a value of 16. The quality of the prediction is illustrated in figure 1. As can be seen, even though e is not always predicted correctly, the prediction usually is offset only by a fixed number. For the arithmetic coder though, this does not prohibit efficient compression as long as certain non-empty child counts appear significantly more frequently than others. This clearly is the case with our prediction. In figure 1 c) the performance of the prediction is shown for the processing of the entire octree: A child count of 1 dominates over the whole octree (since practically all cells on finer levels in the hierarchy have only one child) and our prediction is able to correctly detect these cases.

4.3. Child cell configuration

Given the number of non-empty child cells e there is only a limited number of possible non-empty child cell configurations, e.g. 70 if there are four non-empty child cells or 28 in the case of two. A configuration $\mathcal{T}$ is defined as the set of non-empty child cells: $\mathcal{T} = \{T_i | T_i \text{ is non-empty}\}$. In the second step of the encoding of a cell subdivision the respective configuration has to be specified. To this end, each of the possible configurations is assigned a weight $w(\mathcal{T}_i)$ by the prediction and the array of weighted possible configurations is sorted in ascending order. The prediction is designed to assign lower weight to more likely configurations. The configuration of the subdivision is then encoded as index into the sorted array. As more likely configurations receive lower weights, smaller indices become more frequent than larger ones and the entropy of the sequence of indices decreases.

The prediction is based on the observation that the cell centers of occupied child cells tend to be close to the approximated surface tangent F^C (see also figure 2). As a consequence we sum up the distances of every non-empty child cell's center to the plane F, so that more likely configurations receive a lower weight. However, we do not use the euclidian distance to the plane, but employ a weighted L_1 distance where coordinate directions are weighted with the cosine of the angle between the plane's normal and the direction of the respective axis:

$$d(F,p) = \sum_{i\in\{x,y,z\}} |n(F)_i| \, |p_i - prj(F,p)_i| \, ,$$

with $n(F)$ being the normal to F and $prj(F,p)$ the projection of p onto F. This reflects the fact that the more aligned the plane's normal is with one of the coordinate directions, the less likely it becomes that a cell is occupied if its cell center is not close to the plane along this direction. The weight of a configuration $\mathcal{T}$ is then given by:

$$w(\mathcal{T}) = \sum_{T\in\mathcal{T}} d(F, c(T))$$

4.3.1. Index compression

The index of the configuration in the sorted array is encoded using arithmetic coding under two contexts. The first context is the octree level of the cell C. The second context reflects the expressiveness $e(F)$ of the plane F with respect to the cell subdivision, i.e. the orientation of the plane to the splitting planes of the cell, as the orientation of the plane in space has a strong influence to the accuracy of the prediction, see also figure 3. We set:

$$e(F) = \frac{3}{2}\left(-\frac{1}{3} + \max_{i\in\{x,y,z\}} \frac{|n(F)_i|}{\sum_{j\in\{x,y,z\}} |n(F)_j|}\right)$$

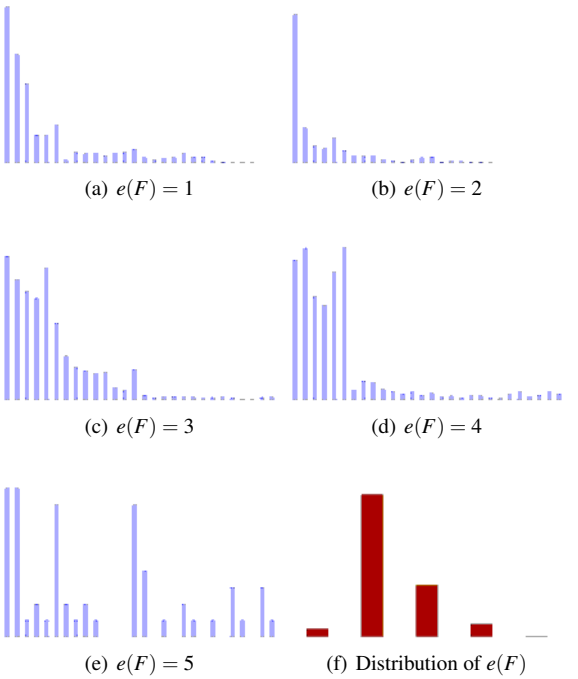


Figure 4: *The quality of the configuration prediction with respect to the expressiveness of the plane F. Depicted are the distributions of the encoded indices in the case of two non-empty child cells (there are 27 possibilities) for the five different expressiveness values. The chart (f) on the lower right gives the distribution of the five different expressiveness values. The data was collected using the Igea model but exhibits the typical behavior we found for a wide range of different point-clouds.*

This gives a value in the interval $[0,1]$ that reflects the angle of the plane to the coordinate directions. In order to use $e(F)$ as context for arithmetic coding it has to be quantized. In our experiments we found that a discrimination of five bins was sufficient and delivered the best results. In figure 4 the distribution of the encoded indices for the different expressiveness values in the case of two non-empty child cells are shown. In the case of two occupied cells, the more aligned the plane is with one, or several, of the axes, the less expressive is the prediction, as usually about four child cells are almost equally close to the plane and no further distinction exists between them. As expected the prediction performs better for less aligned planes, i.e. lower expressiveness values. Please note that the expressiveness values having a low entropy distribution appear more frequently than those with higher entropy, as show in figure 4 (f). Note also that for different numbers of non-empty child counts other expressiveness values may lead to better predictions, e.g. for four occupied cells a prediction with $e(F) = 5$ performs well.

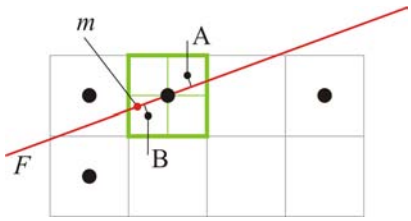


Figure 5: *A cell with only one non-empty child cell is to be subdivided. The centroid m of the points in N_k is depicted as dot on the approximating plane. Cells whose center's projection onto F are farther away from the centroid are more likely to be occupied. Thus, in the depicted case the child cell A is more likely to be non-empty than B even though A and B have equal distance to the plane.*

4.4. Single child-cell configurations

For cells that have only one occupied child cell a second observation can be exploited. Since real-life point clouds usually exhibit a certain regularity due to the regular sampling grid of the scanning device, not only the cells close to the surface can be predicted, but also the location of samples in the surface itself. This is especially relevant for finer levels in the octree hierarchy where only a single sample remains per octree cell and therefore not all child-cells that are close to the approximated plane can be occupied. Since, for lossless compression, the case of only one occupied child-cell dominates all other cases the quality of the prediction for single child-cell configurations has a great impact on the overall compression performance.

We use the centroid m of the weighted neighbors N_k for the sample location prediction within the surface. Quite surprisingly, those child cells whose cell centers' projection on F are farthest away from m are more likely to be non-empty. The reason for this is that the area farther away from the centroid can be seen as undersampled and therefore an introduction of a sample in this area becomes more likely since the surface sampling is expected to be regular and no undersampled regions should exist. An example is depicted in figure 5. Thus, for cells with only one non-empty child cell, the weights $w(\mathcal{T}_i)$ for the eight possible configurations are given as:

$$w(\mathcal{T}_i) = d(F, c(T_i)) - d(m, prj(F, c(T_i)))$$

The distance to m has to be subtracted because the likelihood of T_i being occupied increases for greater values, which is the inverse behavior as exhibited by the distance to the plane. Single child-cell configurations are then encoded in the same way as described above.

5. Traversal order

While a standard breadth-first traversal of the octree already yields good results, the performance of the compression

can still be improved by reordering the traversal. Processing those cells earlier that introduce the greatest error will not only lead to a faster increase of the signal to noise ratio during progressive decompression, but can also improve the overall compression rate. This is due to the fact that the prediction becomes more accurate for cells processed later on. [PK05] use a prioritized traversal order that allows to expand some cells even before all cells of the previous levels have been processed. In our experiments a traversal that retains the breadth-first order of the levels resulted in a better performance though, which we believe is due to the fact that, in our case, expanding cells at different levels introduces irregularities in the point-cloud $\mathcal{Q}$ that adversely affects the quality of the surface approximations $F^C_{\mathcal{Q}}$. Hence we only reorder the cells on every level to augment the traversal.

To get an estimate for the error introduced by a cell, we perform a smoothing step every time a level has been completed. The estimate is based on the assumption that points that are more strongly smoothed are still farther away from the true surface. All points $q_i \in \mathcal{Q}$ are replaced by a smoothed version $\hat{q}_i$ defined as their projection onto the plane $F^{q_i}_{\mathcal{Q}}$. The cells of the level are then traversed in descending order according to the magnitude of the movement made by their cell center q_i, i.e. the distance $\|q_i - \hat{q}_i\|$. As cells with a larger movement are traversed first, cells with a large quantization error are favored over those that already provide a fair approximation to the underlying surface. Therefore larger errors are alleviated earlier in the process and the signal to noise ratio improves more quickly. The estimation of the local surface approximations $F^C_{\mathcal{Q}}$ also benefits from the early removal of larger errors, resulting in an overall gain in the compression rate as the reliability of the non-empty child cell prediction is significantly increased. Using the smoothed versions of the points in $\mathcal{Q}$ for the prediction on the next level has a similar effect by reducing the impact of the coarse quantization on higher levels. Please note though that during the smoothing points are not allowed to be projected out of their corresponding cells and that after the final level of the octree has been processed no smoothing is performed in order retain the losslessness of our method.

6. Color

Color is an important point attribute if the point-cloud is to be used for rendering purposes. [WGE*04] specifically accounted for color similarities when pairing points to construct their point-cloud hierarchy. However, when doing so, some of the geometric similarity may have to be traded for greater similarity in color values, so that the compression rates of the geometry may suffer in a way that diminishes the gain in the compression of the color attributes. We find the point grouping introduced by the octree to provide sufficient similarity in color for many cases, even though spatial coherence does not in general guarantee similarity in point color, since many scanned objects possess wide, uniformly colored regions where color deviations are smooth and stem largely from shading. As our geometry compression is lossless, we also aim at a lossless compression of 24 bit RGB color.

We represent colors in an indexed array, i.e. each point is assigned an index into the array of all colors present in the object. This already saves some bits compared to the 24 bits RGB representation, if the array of colors can be encoded efficiently. As colors themselves are three dimensional attributes they can be compressed in an octree hierarchy as well. This gives an efficient representation for the array of colors.

6.1. Color octree

The compression of the octree of colors proceeds similarly to the one of the positions, albeit different prediction techniques are employed, since, obviously, the prediction rules described above do not apply in the case of colors. To encode a cell subdivision we use the same two-step procedure as above: first the number e of non-empty child cells is encoded, followed by the child cell configuration. For the prediction we again make use of the colors in the traversal front $\mathcal{Q}$ and the nearest neighbors to a cell center $N_k \subset \mathcal{Q}$, which are defined in the same sense as above.

6.1.1. Number of non-empty cells

This time the number of non-empty children e is not explicitly predicted, but simply adaptively arithmetic coded under two contexts. The first context is the current octree level, as the number of non-empty child cells tends to decrease for finer octree levels. The second context is the local color density, since we observe a greater number of non-empty child cells in regions of higher density. The color density $\rho(C)$ around the cell C is measured by the radius of the ball containing the N_k nearest neighbors relative to the width of the cell $l(C)$:

$$\rho(C) = \frac{max_{q \in N_k} \|c(C) - q\|}{l(C)}$$

We truncate $\rho(C)$ at a value of 8.

6.1.2. Child cell configuration

The prediction of a child cell configuration exploits the fact that colors tend to appear in clusters and therefore non-empty child cells are more likely closer to the centroid m of the nearest neighbors N_k. Thus, a child cell configuration $\mathcal{T} = \{T_i | T_i \text{ is non-empty}\}$ is weighted using the distance of the non-empty child cell centers to the centroid m of N_k (note that in the case of colors we do not weight the neighbors with a Gaussian):

$$w(\mathcal{T}) = \sum_{T \in \mathcal{T}} \|c(T) - m\|$$

This way the colors are compressed at rates between 4-5 bits per color.

6.2. Color indices

To compress the point set $\mathcal{P}$ together with point colors, the octree of the colors is compressed first. Then the cells in the octree of points have to be assigned a color index on every level. We want the color referenced by a cell to be close to the mean color of all points in the cell. Hence the color index $i(C)$ for the cell C is chosen such that the L_2-distance between the mean color of the points contained in the cell and the indexed color is minimal, i.e. among all possible colors the nearest neighbor to the mean color of the cell is chosen. Then the compression of $\mathcal{P}$ proceeds as described in section 3-5 with one additional step after a cell subdivision has been encoded, which specifies the color indices of the child cells.

Note that, in accordance with the progressive nature of our algorithm, and in order to improve the compression rate, the array of possible colors is slowly enlarged as the traversal reaches a new octree level. To this end the array of possible colors is chosen from the corresponding level in the octree of colors, i.e. if a cell on level l in the octree is subdivided, the new colors for the child cells are chosen from the colors present on level $l+1$ in the color octree (or the maximum level if $l+1$ is too large). If a cell with only one occupied child is subdivided and the color of the cell has already been encoded within the maximal array of possible colors, the new child inherits its parent's color and no further information has to be encoded.

6.2.1. Color prediction

To efficiently compress the indices a prediction is performed for every new child cell T. The prediction is based on a local surface approximation given by the plane $F_{\mathcal{Q}}^{T}$ defined similarly as in section 4.1, but using $c(T)$ instead of the center of the parent cell. Note that at the time of the prediction the cell centers of the new child cells are not yet contained in $\mathcal{Q}$. The color in T is predicted component-wise, i.e. for each component of the color a linear model is fitted to the respective color components of the surrounding points in N_k, i.e. $F_{\mathcal{Q}}^{T}$ serves as local planar parametrization of the surface in which a linear model can be fitted to the colors of the neighboring points. The three predictions of the color components result in a predicted color $\hat{o}(T)$ for the cell T.

6.2.2. Color index encoding

To encode the color index $i(T)$ of the cell, the array of possible colors is sorted by similarity to the predicted color $\hat{o}(T)$, i.e. all possible colors o_i are assigned a weight $w(o_i,\hat{o}(T)) = \|o_i - \hat{o}(T)\|$ and then sorted in ascending order. Now $i(T)$ is given as the location of the cell's color in the sorted array. Since the predicted color is likely to be similar to the true color of T, $i(T)$ tends to be close to zero. Again, this is exploited by arithmetic coding of $i(T)$.

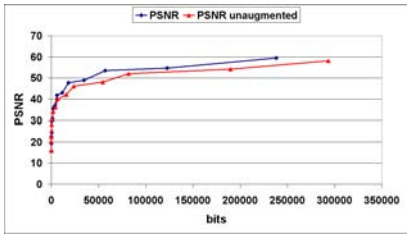


Figure 6: *Comparison of the compression with and without the augmented traversal order for the Venus model.*

7. Results

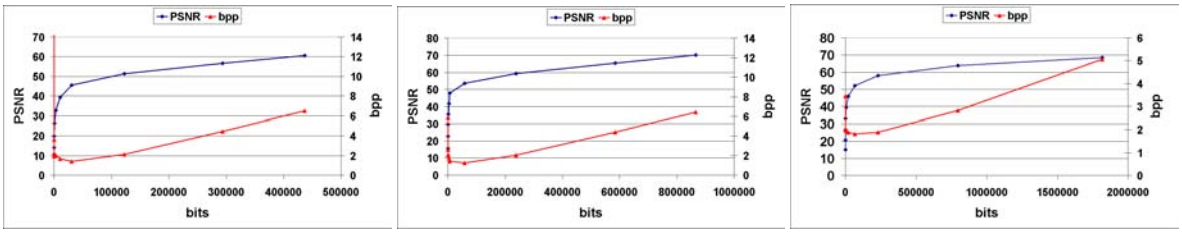
To evaluate our method and compare it with existing approaches, we have applied the algorithm to various point-clouds. As usual, compression performance is measured in bits per point (bpp) and the loss of quality by the *peak signal to noise ratio* (PSNR). When evaluating the error of the geometry compression we follow [WGE*04] and measure the symmetric root mean square distance $RMS(\mathcal{P},\mathcal{Q})$ between the original, unquantized point-cloud $\mathcal{P}$ and the decompressed points $\mathcal{Q}$. The PSNR is then given as $20\log_{10}\frac{d_B}{\max(RMS(\mathcal{P},\mathcal{Q}),RMS(\mathcal{Q},\mathcal{P}))}$ where d_B is the bounding box diagonal of $\mathcal{P}$. $RMS(\mathcal{P},\mathcal{Q})$ is computed by finding nearest neighbors for every point of $\mathcal{P}$ in $\mathcal{Q}$. Taking $RMS(\mathcal{P},\mathcal{Q})$ as error measure is preferable to measuring the RMS between the MLS surfaces as the distance between discrete point-pairs takes into account the difference in the sampling as well. The MLS error on the other hand only gives the distance between the surfaces which neglects the fact that the quality of the sampling plays an important role for point-clouds.

All our results were obtained with $k = 8$ which we found to yield a good compromise between smoothing of the quantization and achieving a precise normal estimation.

First of all we evaluate the effect of the augmented traversal order and the smoothing. In figure 6 the PSNRs are shown, once with the augmented traversal activated and the other time deactivated for the Venus model. Our proposed traversal order improves the compression rate by about 10% on average.

In figure 7 the performance of the progressive compression is shown for three models. Compared to the results of [WGE*04], in case of the Venus model we are able to achieve a PSNR of 59.39 with only 2.03 bpp and obtain a PSNR of 76.65 for 8.71 bpp, whereas their method requires almost 8 bpp to achieve a PSNR of about 57. Our method is able to increase the PSNR rapidly until a PSNR of about 50 is reached, after that the increase of the PSNR is slowed down, but still comparable to that of [WGE*04].

Table 2 compares the bitrates of our method to those of [GKIS05] and [WGE*04]. All models were quantized to



(a) Rabbit (b) Venus (c) Dragon

Figure 7: *Geometry compression: The PSNR versus bits for three models. bpp are measured with respect to the number of points of the decompressed model.*

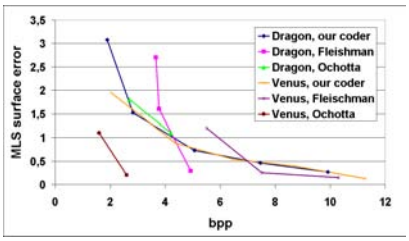


Figure 8: *Comparison of our method to the lossy schemes of [FCOAS03] and [OS04]. The MLS surface error is specified in terms of $10^{-4}d_B$ where d_B is the bounding box of the object. Note that our method becomes lossless for higher bitrates.*

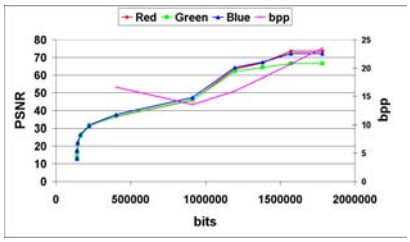


Figure 9: *The PSNR for the different RGB color components during the progressive decompression of the first 12 octree levels for the Santa model. The exact reconstruction is achieved after 16 levels.*

model	#colors	oct	bpp	Total
Santa	32376	4.34	11.5	23.44
MaleWB	58642	3.21	14.5	23.37
FemaleWB	39712	3.13	13.12	21.99

Table 1: *Compression of the colors for different models. We use RGB colors with 24 bits. oct gives the bits per color used by the color octree. bpp gives the bits per point used for the color information (including the octree) after 12 octree levels. Total are the bpp including the geometry.*

12 bits per coordinate direction. In case of the Santa model our method even outperforms the non-progressive scheme of [GKIS05]. We gain about 30% compared to [WGE*04].

To compare our method to the lossy coders of [FCOAS03] and [OS04] we also computed the mean MLS surface error as specified in [FCOAS03]. We are able to achieve superior results to the method of Fleishman for lower bitrates while our method requires more bits to achieve smaller errors. But conversely to Fleishman, in the limit our method is able to produce a lossless representation. For the dragon, the rates of our coder are comparable to those of [OS04], but for the Venus their method achieves smaller errors for lower bitrates. This is due to the fact that, in the case of the Venus, their method requires only very few planar patches to cover the entire surface. Note however that their coder is lossy and only semi-progressive.

When comparing our method to the one of [PK05], as done in table 3, one has to consider the fact that our method does not make use of any connectivity information in the models. The effects of this become apparent when compressing less regularly and less densely sampled mesh geometry, such as the horse and feline models. Here [PK05] can exploit the additional information to achieve higher geometry compression rates than our coder. Please note however that the cost of connectivity encoding has to be included in the results as the additional information is explicitly exploited in the geometry prediction. Compared to the combined cost our method requires slightly less bpp. For true point-sampled geometry, as in the rabbit model, our method performs equally well as their geometry compression, but even without the overhead of connectivity encoding.

We also give some results for the color compression. The differences in color are measured by finding corresponding point pairs in the decompressed and original model with a nearest neighbor search and computing the root mean square error for each color component. In figure 9 the PSNR for all three RGB components during decompression of the first 12 octree levels are depicted. Note that although the colors

model	n	[GKIS05]	[WGE*04]	Ours
Venus	134k	10.83	14.28	11.27
Santa	75k	12.23	18.28	11.94
MaleWB	148k	7.26	13.59	8.87

Table 2: *Comparison of the bitrates (bpp) of our coder to those of [GKIS05] and [WGE*04]. Note that [GKIS05] is a single rate method and can therefore be expected to deliver better compression. All models have been quantized to 12 bits per coordinate direction.*

model	n	[PK05] Geom.	[PK05] Total	Ours
Horse	19k	13.7	16.6	16.02
Feline	49k	13.2	16.7	14.88
Rabbit	67k	11.4	14.8	11.37

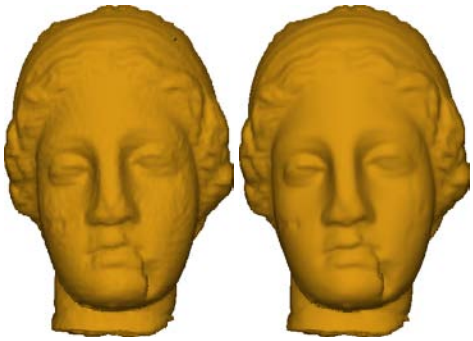
Table 3: *Comparison of the bitrates (bpp) of our coder to the one of [PK05]. All models were quantized to 12 bits per coordinate direction.*

have already full precision on the last levels, the error has not vanished because the corresponding point pairs are not entirely correct yet due to the point mismatches caused by the quantization at this stage of the progressive decoding. Also note that the bpp is unusually large for the first levels, which stems from the fact that our current implementation encodes the entire octree of colors at the beginning of the file. However this is not inherent to our method as the encoding of the color octree could be interleaved with the geometry information. Table 1 gives the bpp required for the color information for different models.

Finally, figures 12, 11 and 10 show various stages of the progressive decompression for different models. Renderings are obtained with Pointshop3D [ZPKG02] using two-sided normals. Note that compared to the tree based mesh coders of [PK05] and [DG00], in the case of point-clouds the coarse quantization on lower levels is far less visible. Our method further reduces the artifacts with the smoothing step.

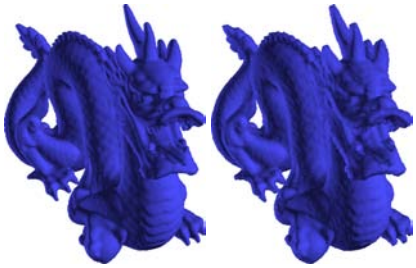
8. Conclusion and Future Work

In this paper we introduce a method for progressive lossless compression of point-sampled models that achieves compression rates that outperform those of previous lossless algorithms. During the progressive decompression, the performance of our approach is comparable to that of previous lossy methods. Although the main focus of our technique is geometry compression, we show that color attributes can be handled by our framework as well. Using local surface approximations yields accurate predictions for the cell subdivisions that are exploited by arithmetic coding. Adapting an augmented traversal order and incorporating a smoothing step improves not only the PSNR during progressive decompression but also the overall compression rate.



(a) 2.03 bpp (b) 6.45 bpp

Figure 10: *Two different stages of the decompression for the Venus model.*



(a) Original (b) 230309 bits

(c) 1816466 bits

Figure 11: *Different stages of the decompression of the dragon. The rightmost model is the original. The model in b) requires 1.89 bpp and in c) 5.06 bpp*

In future we plan to include additional point attributes in the algorithm. Especially normals should benefit from the prediction with the local planar surface approximations. Further it needs to be evaluated if using different quantizations of the color space can improve the compression rates without affecting the visual quality. Some of our prediction techniques could also be applied in the context of mesh compression.

Acknowledgements

The 'Venus', 'Rabbit', 'Santa', 'MaleWB' and 'FemaleWB' models appear courtesy of Cyberware, Inc (www.cyberware.com). The 'Dragon' model was taken from the Stanford 3D Scanning Repository (http://graphics.stanford.edu/data/3Dscanrep/). We use the arithmetic coder of the QccPack library that is provided and maintained by Jim Fowler.

References

[ABCO*03] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Computing and rendering point set surfaces. *IEEE Transactions on Visualization and Computer Graphics 9*, 1 (2003), 3–15.

[AG03] ALLIEZ P., GOTSMAN C.: Recent advances in compression of 3d meshes. In *Proceedings of the Symposium on Multiresolution in Geometric Modelling* (2003).

[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.

[Dee95] DEERING M.: Geometry compression. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1995), ACM Press, pp. 13–20.

[DG00] DEVILLERS O., GANDOIN P.-M.: Geometric compression for interactive transmission. In *VIS '00: Proceedings of the conference on Visualization '00* (Los Alamitos, CA, USA, 2000), IEEE Computer Society Press, pp. 319–326.

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

[GD02] GANDOIN P.-M., DEVILLERS O.: Progressive lossless compression of arbitrary simplicial complexes. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2002), ACM Press, pp. 372–379.

[GKIS05] GUMHOLD S., KARNI Z., ISENBURG M., SEIDEL H.-P.: Predictive point-cloud compression. In *ACM SIGGRAPH Conference Abstracts and Applications* (2005).

[GS98] GUMHOLD S., STRASSER W.: Real time compression of triangle mesh connectivity. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), ACM Press, pp. 133–140.

[KG00] KARNI Z., GOTSMAN C.: Spectral compression of mesh geometry. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 279–286.

[MMG06] MERRY B., MARAIS P., GAIN J.: Compression of dense and regular point clouds. In *Afrigaph '06: Proceedings of the 4th international conference on Computer graphics, virtual reality, visualisation and interaction in Africa* (New York, NY, USA, 2006), ACM Press, pp. 15–20.

[MNW98] MOFFAT A., NEAL R. M., WITTEN I. H.: Arithmetic coding revisited. *ACM Trans. Inf. Syst. 16*, 3 (1998), 256–294.

[OS04] OCHOTTA T., SAUPE D.: Compression of point-based 3d models by shape-adaptive wavelet coding of multi-height fields. In *Proceedings of the Eurographics Symposium on Point-Based Graphics* (June 2004), pp. 103–112.

[PG01] PAULY M., GROSS M.: Spectral processing of point-sampled geometry. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2001), ACM Press, pp. 379–386.

[PGK02] PAULY M., GROSS M., KOBBELT L. P.: Efficient simplification of point-sampled surfaces. In *VIS '02: Proceedings of the conference on Visualization '02* (Washington, DC, USA, 2002), IEEE Computer Society, pp. 163–170.

[PK05] PENG J., KUO C.-C. J.: Geometry-guided progressive lossless 3d mesh coding with octree (ot) decomposition. *ACM Trans. Graph. 24*, 3 (2005), 609–616.

[RL00] RUSINKIEWICZ S., LEVOY M.: Qsplat: a multiresolution point rendering system for large meshes. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 2000), ACM Press/Addison-Wesley Publishing Co., pp. 343–352.

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

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

[WGE*04] WASCHBÜSCH M., GROSS M., EBERHARD F., LAMBORAY E., WÜRMLIN S.: Progressive compression of point-sampled models. In *Proceedings of the Eurographics Symposium on Point-Based Graphics* (June 2004), pp. 95–102.

[ZPKG02] ZWICKER M., PAULY M., KNOLL O., GROSS M.: Pointshop 3d: an interactive system for point-based surface editing. *ACM Trans. Graph. 21*, 3 (2002), 322–329.

(a) 221746 bits (b) 917174 bits (c) 1782856 bits

Figure 12: *Different stages of the decompression of the santa models. The last stage is after level 12 in the octree and is visually indistinguishable from the original. Note that even for coarse approximations no obvious quantization grid is visible.*