Eurographics Symposium on Point-Based Graphics (2004)
M. Alexa, S. Rusinkiewicz, (Editors)

Compression of Point-Based 3D Models by Shape-Adaptive Wavelet Coding of Multi-Height Fields

Tilo Ochotta, Dietmar Saupe

Department of Computer and Information Science, University of Konstanz, 78457 Konstanz, Germany

Abstract

In order to efficiently archive and transmit large 3D models, lossy and lossless compression methods are needed. We propose a compression scheme for coordinate data of point-based 3D models of surfaces. A point-based model is processed for compression in a pipeline of three subsequent operations, partitioning, parameterization, and coding. First the point set is partitioned yielding a suitable number of point clusters. Each cluster corresponds to a surface patch, that can be parameterized as a height field and resampled on a regular grid. The domains of the height fields have irregular shapes that are encoded losslessly. The height fields themselves are encoded using a shape-adaptive wavelet coder, producing a progressive bitstream for each patch. A rate-distortion optimization provides for an optimal bit allocation for the individual patch codes. With this algorithm design compact codes are produced that are scalable with respect to rate, quality, and resolution. In our encodings of complex 3D models competitive rate-distortion performances were achieved with excellent reconstruction quality at under 3 bits per point (bpp).

Categories and Subject Descriptors (according to ACM CCS): I.4 [Coding and Information Theory]: Data compaction and compression

1. Introduction

The number of 3D models that are created by means of geometry acquisition, by modelling tools, and by means of simulation is increasing rapidly. In order to allow space-saving storage and time efficient transmission of 3D models, methods for their efficient representation are needed. Thus, geometry compression has developed to become an important field in computer graphics.

The most common type of 3D data models are surfaces. There are three main categories of representations for surfaces, namely meshes, volumetric data yielding isosurfaces, and point-based models. A surface encoder transforms an explicit surface representation of one of the above types into a compact bitstream which can be decoded at the receiver in order to generate a surface reconstruction. The reconstruction can be lossless or lossy.

1.1. Related work

Common 3D graphics systems are based on polygon meshes. For meshes there are three major components to transmit, namely vertex coordinates, mesh connectivity, and photometric values for each vertex or face. Methods are needed to achieve compression for all three types of data. Research in 3D compression has focused mainly on the geometry of the models. Several fundamentally different encoding methodologies have been developed.

In Taubin and Rossignac [43, 36] and Touma and Gotsman [44] the connectivity of the mesh is encoded separately. The coordinate data of the vertices is treated in a second pass. Vertex locations are predicted and the prediction residuals are entropy coded or vector quantized [4]. Connectivity coding has been greatly improved [1, 21] and out-of-core geometry coding for extremely large meshes was developed [18].

A different paradigm for mesh compression was developed by applying methods for parameterization of surfaces [10]. Surface parameterizations can be encoded and the receiver may reconstruct a surface approximation by resampling the parameterization at arbitrary points. In order to produce a mesh reconstruction, the scattered data points need to be triangulated. This is straightforward, when the parameter-

ization is above a base mesh [20, 22, 24]. When patches are encoded separately without enforcing boundary compatibility, a zippering method is required to connect the patches in a common mesh [39].

Another approach for representing 3D models is given by point sets. In 3D model acquisition, range scanners generate point clouds. In point-based rendering, surface elements (surfels, [34]) are used for each point to visualize the complete surface of the object. In general, there is no triangulation procedure at any step of the processing pipeline, starting with acquisition of point data up to the rendering of the corrected and simplified 3D model. Various algorithms were proposed which deal with point-based 3D models. For example, Zwicker et al. [47] introduced Pointshop3D, an interactive modeller. There are different efficient ways to render surfaces from point-based models, mostly based on splatting [2, 3, 15, 34, 37, 41, 46]. In [3], a pure software implementation rendered up to 14 million shaded and textured points or 4 million high quality splats per second on a PC platform. When dedicated hardware support becomes available for point-based rendering, such systems may well provide a rendering performance equal or even superior to traditional mesh-based renderers on high performance graphics cards.

Efficient representation of point-based 3D models has been discussed primarily in the rendering context, e.g., [37, 3], rather than for the purpose of storage and transmission. Point clouds can be represented hierarchically and compressed in the spatial domain using tree data structures such as kd-trees [7] and octrees [3, 38, 33]. Compression is achieved by pruning the tree at nodes where the corresponding spatial cell does not contain any of the surface points. Fleishman et al.[9] described a multi-resolution method that is based on an embedded sequence of point sets. Local coordinate frames and least squares approximation are used to define approximating, so called moving least squares (MLS) surfaces and refinement points.

In some papers it was recognized that one can achieve excellent coding results when well developed methods from image wavelet coding are applied [20, 22, 24, 32, 42]. Since images are given in pixels on a regular grid the surface must be regularly resampled in order to apply image coding algorithms to geometry compression. Remeshing algorithms can achieve this. For example, the MAPS algorithm [25] produces a semi-regular mesh of the original topology. Each triangle in the base mesh corresponds to a connected region of the given mesh which can be parameterized. In place of parameterization over a base mesh one can create 'geometry images' over one or several non-intersecting regions in a rectangular domain [13, 39], or over a sphere [17, 35], if topology allows.

The coordinate data at the regular or semi-regular mesh is three-dimensional by nature. However, the construction of the multi-resolution semi-regular mesh can be organized with the goal that only normal displacements are necessary to define the newly generated vertices of the next finer level, yielding so-called 'normal meshes' [14, 22]. In practice, however, the subdivision is algorithmically complex and not all detail coefficients can be expressed by only a scalar value.

A method that enforces a parameterization of the given surface on a regular grid such that only scalar values are required to locate the corresponding 3D points on the surface can be obtained by representing the surface as a set of approximately flat (possibly overlapping) patches such that each of them can be seen as a graph of a scalar function over a rectangular 2D domain located on a plane near the surface patch (figure 1a-c). In other words, the parameterized surface can be represented as an atlas of charts that are approximated by a collection of height fields properly placed in 3D space. In [30], such local parameterizations were used to implement signal processing tasks like filtering on surfaces. Because the height fields in [30] are overlapping, a blending technique was necessary when merging the processed charts for surface reconstruction.

In [39], the given surface mesh was partitioned into disjoint approximately flat regions which were represented as charts. The domains of the charts having irregular boundaries are laid out without overlap in a rectangle, which are resampled on a regular grid. These charts were designed to minimize distortion and, thus, are not height fields and coordinate values are three-dimensional. The purpose of the work was to develop multi-chart geometry images, such that surface reconstruction gives the best possible quality with a given memory constraint.

1.2. Overview of our work

In this paper, we contribute an encoding method for geometry of point-based surfaces. Our work has been inspired mainly by the two papers, "Spectral processing of point-sampled geometry" by Pauly and Gross [30], and "Multi-chart geometry images" of Sander et al. [39]. Here we combine, modify, and extend their methods for the purpose of geometry compression. We propose to use an atlas of height fields as in [30], which, however, are not overlapping and have irregular boundaries as in [39]. The advantages of such an atlas of height fields over normal meshes and over geometry images are simplicity and strictly scalar point coordinate representation. However, clearly there are three drawbacks to this approach, namely

- the charts have irregular boundaries that require a portion of the encoding rate,
- a complex surface may require a large number of height field charts, and
- the sampling of height fields at regular grid positions generally does not yield a uniform sampling of the surface.

In this paper, we propose to use a heuristic split-and-merge partition of the surface point set that is similar to that used in

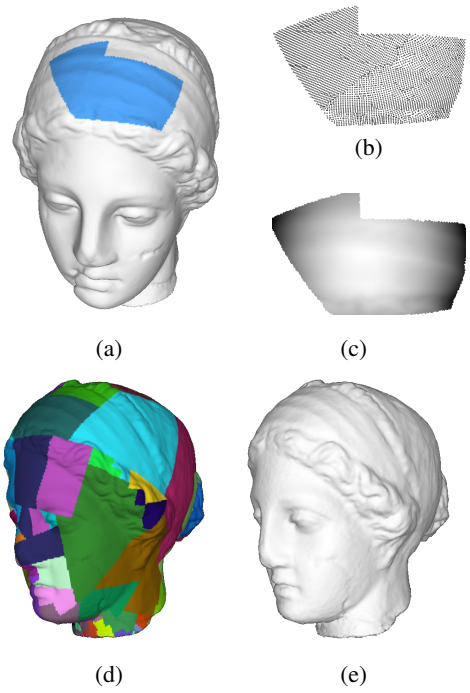

Figure 1: *(a) A patch is considered as cluster of points in the original model; the points are parameterized to a base plane (b) and resampled on a regular grid to obtain a 2D image (c); The model is represented as a set of height fields over the respective base planes (d); The height values are encoded and decoded by shape-adaptive wavelet coding yielding the reconstructed model (e).*

[30]. The irregular height field boundaries can be extracted and encoded using standard bitmap coding. The height field data over irregular domains is encoded by a special state-of-the-art shape-adaptive wavelet coder.

Wavelet-based shape-adaptive image coding has recently been proposed whereby wavelet coefficients corresponding to transparent image regions are permanently considered insignificant [27]. This approach can be implemented, e.g., for the SPIHT and EBCOT coders [29, 28], but we have chosen shape-adaptive tarp coding [11], which is easier to implement and which yields a better performance than shape-adaptive SPHIT and comparable performance to shape-adaptive EBCOT [11].

Figure 1 gives an example of the main steps in our approach. In (a) to (c), we consider a patch which is parameterized over a base plane and resampled to a scalar height field. The height values are encoded by shape-adaptive wavelet coding. The set of height fields for all patches in the model (d) yields the reconstructed point set (e).

The remaining part of this paper is organized as follows. In section 2 we begin by recalling the concept of moving least squares surfaces which are at the basis of our compression approach (subsection 2.1). The partitioning of the point set into clusters is given in subsection 2.2. Subsection 2.3 explains how a height field corresponding to a cluster of points is parameterized. The encoding of the constituent data for a multi-height field representation is given in subsection 2.4. Section 3 is devoted to the rate-distortion optimization of our coder. Results are discussed in section 4, and we conclude with remarks about future work in section 5.

2. The compression pipeline

2.1. Error measures

When compressing a point-set surface given by a point set P, an encoder produces a bitstream that is converted back into another point set $\hat{P}$ at the decoder. In order to evaluate the quality of the approximation we follow previous work [22, 24] in the spirit of METRO [5] and define a symmetric root-mean-square (rms) error and a peak-signal-to-noise ratio (PSNR) based on surfaces $S(P)$ and $S(\hat{P})$ that are given by P and $\hat{P}$.

Let P denote a finite point set in $\mathbb{R}^3$, corresponding to an original surface $S = S(P)$, that we choose to define as the moving least squares (MLS) surface for the point set P. Moving least squares surfaces were recently applied to point sets in computer graphics in [2, 9, 30, 31, 26]. Locally an MLS surface is obtained by approximating nearby points from P by a least-squares polynomial fit. $S(P)$ is a smooth surface depending on the input point set P which is defined without piecewise parameterization. Computationally, a point $p \in S(P)$ is found by applying a certain projection operator $\mathcal{P}$ to an arbitrary point $q \in \mathbb{R}^3$, $p = \mathcal{P}(q)$. To compute the projection, a nonlinear equation must be solved by applying an iteration procedure. A point $p \in S$ projects on itself. Thus, the complete MLS surface is given by the set of fixed points, $S = \{p \mid p = \mathcal{P}(p)\}$. For details, see [26, 2].

If $\hat{P}$ denotes another point set with a corresponding MLS surface $\hat{S} = S(\hat{P})$, then the (one-sided) L_2-distance of $\hat{S}$ as an approximation of S is given by

$$d(\hat{S},S) = \left(\frac{1}{\text{area}(\hat{S})} \int_{\hat{S}} (d(p,S))^2 dp \right)^{\frac{1}{2}}$$

where $d(p,S)$ denotes the distance of the point p to the surface S. The symmetric rms error is given by

$$E_{rms}(\hat{S},S) = \max\left(d(\hat{S},S), d(S,\hat{S})\right).$$

In practice $d(\hat{S},S)$ is computed by sampling the surface $\hat{S}$ and averaging squared distances. The distances $d(p,S)$ themselves are estimated by the applying the projection operator, $d(p,S) \approx ||p - \mathcal{P}(p)||$.

The PSNR is given by $20\log_{10}\frac{d_B}{E_{rms}(\hat{S},S)}$ where d_B denotes the length of the diagonal of the minimal bounding box of the given surface S.

2.2. Geometry partitioning

The given point set P is partitioned into a number of point clusters with the goal that for each cluster the portion of the surface $S(P)$ that can be attributed to the cluster (called a patch) can be parameterized as a height field and efficiently be encoded. There are many surface partitioning methods, see, e.g., [25, 19, 30, 40, 39]. In [30] the objective is as in this paper to arrive at height field parameterizations. Thus, our approach is similar.

We adopt a standard split-merge procedure. The splitting of a point cluster is done using principal component analysis (PCA), dividing the cluster using a plane passing through the center of mass of the points and orthogonal to the first principal component vector. Initially, all points in P belong to the same cluster. The splitting is done recursively until each cluster satisfies a normal cone condition. We assume that besides the points, also corresponding normal vectors are given. If that is not the case, then normal vectors can be estimated, for example by using covariance analysis [31], whereby the normal orientations are reconstructed using the technique described in [16]. The normal cone condition specifies that all normals belonging to the points in a cluster are in a cone with its apex at the origin and an apex angle that does not exceed a threshold which is a parameter of the method. Thus, if n_c denotes the normal vector along the axis of a normal cone then for each normal n in the cone we require $\langle n, n_c \rangle \geq c_0$ for some constant $c_0 < 1$. To compute a normal cone we use mini balls [12] as in [30]. We may assume that for sufficiently large c_0 and for surfaces sampled densely enough the normal cone condition guarantees that the corresponding surface patch can be seen as a height field over a base plane orthogonal to the normal cone axis. After splitting, we have a partitioning of P in a number of clusters P_i, $i = 1, 2, ...$

In the merge phase, neighboring clusters are iteratively merged. To begin we define a neighborhood graph where vertices denote clusters and edges indicate neighboring clusters. We propose to classify a point $p \in P_i$ as a cluster boundary point if the set of its k nearest neighbors in P contains at least one point from a different cluster $P_j, i \neq j$. This induces the set of neighboring clusters for each cluster and thus the required neighborhood graph. In our experiments, we found that $k = 5$ led to good results. During the merging we require in addition that the normal cone condition is satisfied for a slightly relaxed constant $c_1 < c_0$. Thus, all edges (i, j) where $P_i \cup P_j$ violates this criterion are removed from the neighborhood graph.

Let us assume that each cluster P_i incurs some costs $J(P_i)$. If cluster P_i is a neighbor of P_j, then after merging P_i with P_j the costs are $J(P_i \cup P_j)$. Therefore, we sort edges (i, j) in the neighborhood graph according to the difference in costs $J(P_i \cup P_j) - J(P_i) - J(P_j)$ and iteratively collapse edges with minimal edge costs followed by an update of the neighborhood graph.

The cost function for a given cluster, $J(P_i)$, should capture the encoding cost for the patch given by P_o. In this paper, we set the cost equal to the sum of the squared differences $||n - n_c||^2$, taken over all normal vectors n belonging to the points of cluster P_i and where n_c again denotes the normal vector defining the corresponding normal cone for P_i. The motivation behind this definition is the expectation that large normal variations lead to height fields that are rough and, thus, require a higher encoding rate.

Our partitioning scheme differs from others, e.g., [30], in that we terminate the recursive splitting as soon as a patch is found that passes the normal cone condition. Further subdivision would lead to more convoluted patches which require higher rates to encode.

2.3. Parameterization of a patch

The goal of our parameterization is to represent a surface patch, given by a cluster of points obtained in the partitioning, as a regularly sampled height field over some base plane. The points satisfy the normal cone condition and, thus, it is natural to define the base plane as the unique plane which passes through the center of mass of the point cluster and which is orthogonal to the axis of the cone in the normal cone condition. The center of mass also serves as the origin of a coordinate system for the base plane. The points of the cluster are orthogonally projected onto the plane, and a rectangle is fitted for the projected points [45]. This rectangular region serves as the domain for the sampling of the height field [30].

The next step is to extract the boundary of the patch in its parameter domain, i.e., the corresponding rectangle in the base plane, and to define the resolution of the regular grid for sampling the patch height field. For a given square regular grid on the rectangular domain, we define for each grid point, whether it belongs to the support of the height field or not according to the following procedure. We define the average parameter sampling density of the cluster, μ. More precisely, μ is the average distance of the projected points in the base plane to their three respective nearest neighbors. Now, given a grid point in the rectangular domain, we say, that it belongs to the support of the height field, if its distance to the nearest projection of one of the cluster points into the base plane is less than 1.5μ. This procedure defines the support of the height field as a binary mask. The resolution of the sampling grid on the rectangle is chosen such that the corresponding number of sampling points in the masked area is roughly equal to the number of points in the cluster. With higher, respectively lower resolutions we can obtain upsampled or downsampled parameterizations, if desired.

Finally, to complete the parameterization, we compute a height value for each grid point center in the masked area, such that the point above the base plane at the grid point coordinates is precisely on the MLS surface of the original

data points. These heights are computed iteratively with a Newton-like method using the MLS projection operator $\mathcal{P}$ (see Section 2.1). For a given point in 3-space, initialized as the grid point location on the base plane, we first project it on the MLS surface by $\mathcal{P}$ and compute the tangential plane of the surface at the projection. Then the intersection of this tangential plane with the line through the pixel center and normal of the base plane defines a new point which is used for the next MLS projection. This procedure is iterated until convergence.

2.4. Patch encoding with shape-adaptive wavelet coder

The foremost goal of the encoding is to enable the decoder to reconstruct the elevation data for each patch parameterization from the encoder output. Thus, for each patch, we encode

- the base plane rectangle and its grid sampling resolution,
- the binary mask defining the support of the height field on the base plane rectangle, and
- the scalar height field values.

The base plane rectangle can be encoded using eight quantized floating point numbers defining the coordinates of the origin and two vectors corresponding to the sides of the base plane rectangle.

There are many methods for bitmap encoding. For simplicity, we have used in our implementation and adaptive context-based arithmetic coder [23] for the binary masks of the height field supports.

Our height fields have irregularly shaped supports. We propose to use an encoding method that has been shown to be efficient for shape-adaptive image coding [27]. The height fields are padded with zeroes to fill the entire rectangular-array and wavelet transformed (using the 4-scale 9/7 biorthogonal filter). Coefficients belonging to regions that overlap with the masked areas are entropy coded in a significance pass and a refinement pass as usual in wavelet based image coding. In the shape-adaptive tarp coder, a special estimation for the probability of significance of a wavelet coefficient is applied. For details, see [11].

3. Optimal bit allocation

Our rate-distortion optimization provides optimal bit allocation between the patches as follows. The shape-adaptive tarp coder employs a progressive bitplane coder of wavelet coefficients. Thus, for each patch the output bitstream may be truncated yielding a certain reconstruction quality at the decoder. The rate-distortion optimization optimally allocates the bits in the target bitrate for the entire 3D model between the patches such that the overall reconstruction quality is maximal. We achieve this goal using standard discrete Lagrangian optimization [8]. We briefly describe the procedure. For patch P_i let $(R_i^{(j)}, D_i^{(j)})$, $j = 1,2,\ldots$ denote the

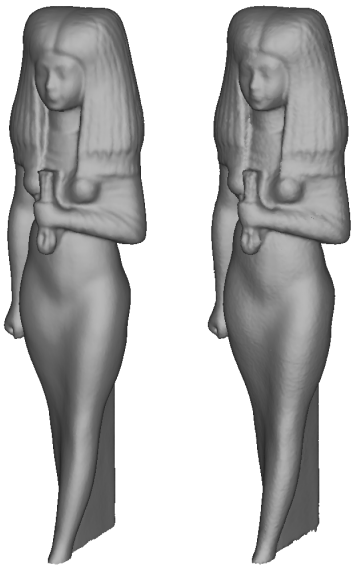

Figure 2: *Encoding of 'Isis'; left: original model, right: reconstructed model, 2.13 bpp, $E_{rms} = 0.87 \cdot 10^{-4} d_B$.*

(finitely many) rate-distortion points achievable by truncating the tarp coder output at different positions numbered by $j = 1,2,\ldots$. The rates $R_i^{(j)}$ include bits for the base plane parameters, for the boundary code, and the truncated wavelet coefficient bit stream. The distortion is the sum of the squared differences between the true and the decoded height values. The truncation index $j = j(i)$ for patch i is given by

$$j(i) = \arg \min_{j=1,2,\ldots} (R_i^{(j)} + \lambda \cdot D_i^{(j)}).$$

This yields a solution with total rate $\sum_i R_i^{(j(i))}$ with minimal total square distortion $\sum_i D_i^{(j(i))}$ [8]. In order to meet a prescribed total rate constraint we vary the Lagrange parameter λ, using the bisection method.

Using the optimization described above, we obtain a rate-distortion optimal coding for a given partition of the surface into patches P_i. Each patch P_i contributes an amount

$$J(P_i) = J_\lambda(P_i) = \min_j (R_i^{(j)} + \lambda \cdot D_i^{(j)})$$

to the total Lagrangian cost $J = J_\lambda = \sum_i J_\lambda(P_i)$.

4. Experimental results

We present experimental results for the models 'Isis', 'Igea', and 'Rabbit' which are available at the Cyberware courtesy [6], and 'Dragon' which was taken from the Stanford 3D Scanning Repository. Overall coding results for various bit rates are shown in figure 3. The bit rate is the total amount of bits divided by the number of input points ($|P|$). The error

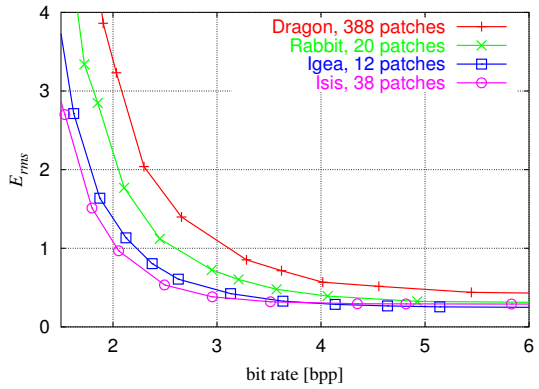

Figure 3: *Coding results for 'Dragon' (437645 points), 'Rabbit' (67039 points), 'Igea' (134345 points), and 'Isis' (187644 points); the error E_{rms} corresponds to the symmetric rms error and is expressed in units of 10^{-4} of the bounding box of the original model.*

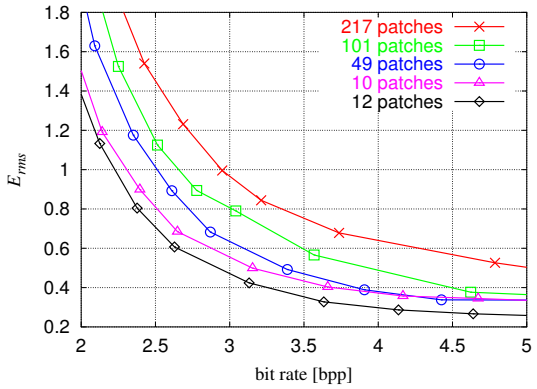

Figure 4: *Coding results for various numbers of patches for 'Igea'.*

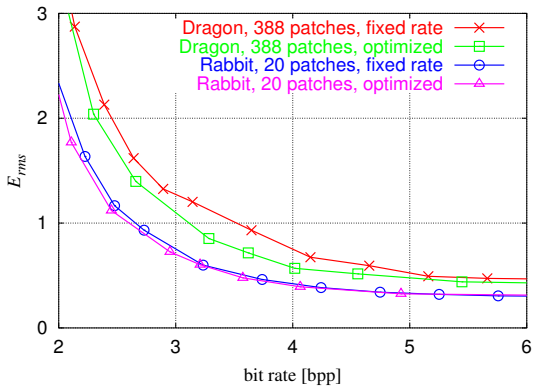

Figure 5: *Coding results for 'Dragon' and 'Rabbit' with and without rate-distortion optimization.*

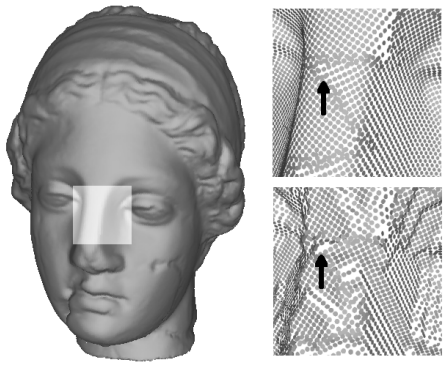

Figure 6: *Detail of reconstructions near patch boundaries at bit rates of 2.5 bpp (upper right) and 1.0 bpp (lower right). Patch discontinuities (marked) can be observed only at very low bit rates.*

is measured in the sense of E_{rms} (section 2.1). All errors in this paper are expressed in units of $10^{-4}d_B$, where d_B is the length of the diagonal of the bounding box of the original model. Detailed coding results are listed in Table 1.

Figure 4 compares the coding results for five different patch layouts of 'Igea'. Lowering the patch number down to 12 improves the coding performance due to decreasing costs for parameter planes and increasing efficiency of the wavelet coder. However, further reduction of the patch number leads to height field artifacts worsening distortion. We found that this behavior is typical. In general, there is an optimal number of patches for which the rate-distortion curve is lowest.

Figure 5 demonstrates that our rate-distortion method is important for achieving good performances when the number of patches is large, which is necessary for complex models like 'Dragon'.

Figure 2 shows the original and reconstructed 'Isis' model at a bit rate of 2.13 bpp. For the tested models we obtain visually pleasant decoded models already at bit rates around 2 to 3 bits per point.

Although we independently encode patches we do not observe boundary artifacts unless we severely reduce the bit rate (see figure 6).

In Figure 7 we present visualizations for the rms-error at different bit rates for 'Igea' and 'Dragon'. The leftmost figures in the first and third row show the original model and below the patch layout. The next figures in the first and third rows show decoded models at different bit rates. The figures below show the original model and visualize the distance from each point to the MLS surface of the reconstructed point set, $d(p,\hat{S}), p \in P$.

header costs [bits]	param. costs [bits]	mask costs [bits]	wavelet costs [bits]	total bit rate [bpp]	E_{rms} $[10^{-4} d_B]$	PSNR [dB]
Dragon, $\|P\| = 437645$, 388 patches, $\|\hat{P}\| = 441115$						
464	48112	222625	333816	1.38	12.61	57.98
464	48112	222625	618200	2.03	3.23	69.81
464	48112	222625	1165872	3.28	0.85	81.38
Isis, $\|P\| = 187644$, 38 patches, $\|\hat{P}\| = 187497$						
464	4712	24433	194848	1.20	6.27	64.06
464	4712	24433	355832	2.05	0.97	80.28
464	4712	24433	565816	3.17	0.36	88.96
Igea, $\|P\| = 134345$, 12 patches, $\|\hat{P}\| = 135107$						
464	1488	13272	135152	1.12	8.95	60.97
464	1488	13272	270248	2.13	1.13	78.91
464	1488	13272	472928	3.63	0.33	89.70
Rabbit, $\|P\| = 67039$, 20 patches, $\|\hat{P}\| = 67635$						
464	2480	10864	60600	1.11	14.38	56.85
464	2480	10864	127456	2.11	1.77	75.04
464	2480	10864	225728	3.57	0.48	86.42

Table 1: *Detailed coding result for 'Dragon', 'Isis', 'Igea', and 'Rabbit'. The error is computed as described in section 2.1 (d_B is the length of the diagonal of the bounding box).*

Model	coder in [9]		proposed coder	
	costs	error	costs	error
Igea (Venus)	5.0	1.2	1.6	1.1
	7.6	0.2	2.6	0.2
Isis	3.8	1.8	1.4	1.9
	6.5	0.3	4.2	0.3
Dragon	3.7	2.7	2.7	1.8
	4.0	1.0	4.2	1.1

Table 2: *Comparison of our coder to the coder in [9]; here the error is as in [9] and measured in terms of $10^{-4} d_B$; costs are given in bits per point.*

Model (patches)	Rabbit (20)	Igea (12)	Isis (38)	Dragon (388)
pre-processing	1.7	3.4	5.7	9.9
partitioning	4.6	9.7	21.2	38.7
parameterization	0.1	0.2	0.2	0.4
resampling	106.2	196.4	278.8	872.3
encoding	0.4	0.8	1.0	1.5
total	113.0	210.5	306.9	922.8

Table 3: *Encoding times in seconds. Decoding of the coordinate data takes about one to two seconds.*

We compared our scheme to the coder of Fleishman [9] which uses iterative refinements of a base point set in order to obtain a reconstructed model, table 2. In contrast to [9] we do not need to encode a base model which takes about 2 or 3 bpp. Our coder already achieves visually pleasant reconstructed models at these rates. For table 2 we implemented the error measure in [9] and applied it to our reconstructed models. This error is different to ours. The results in table 2 for the coder in [9] were read off the graphs in the figures of [9].

Our results for compression of point-based models are comparable to those using progressive geometry compression [20] and normal meshes [14]. However, note that in [20, 14], the models are not equal to those we have used, and that the error measure is based on meshes.

Run times of the encoder are listed in table 3 and range from about 2 to 15 minutes. For the experiments we used a 2.4 GHz CPU with 1 GB RAM. The times are dominated by the resampling.

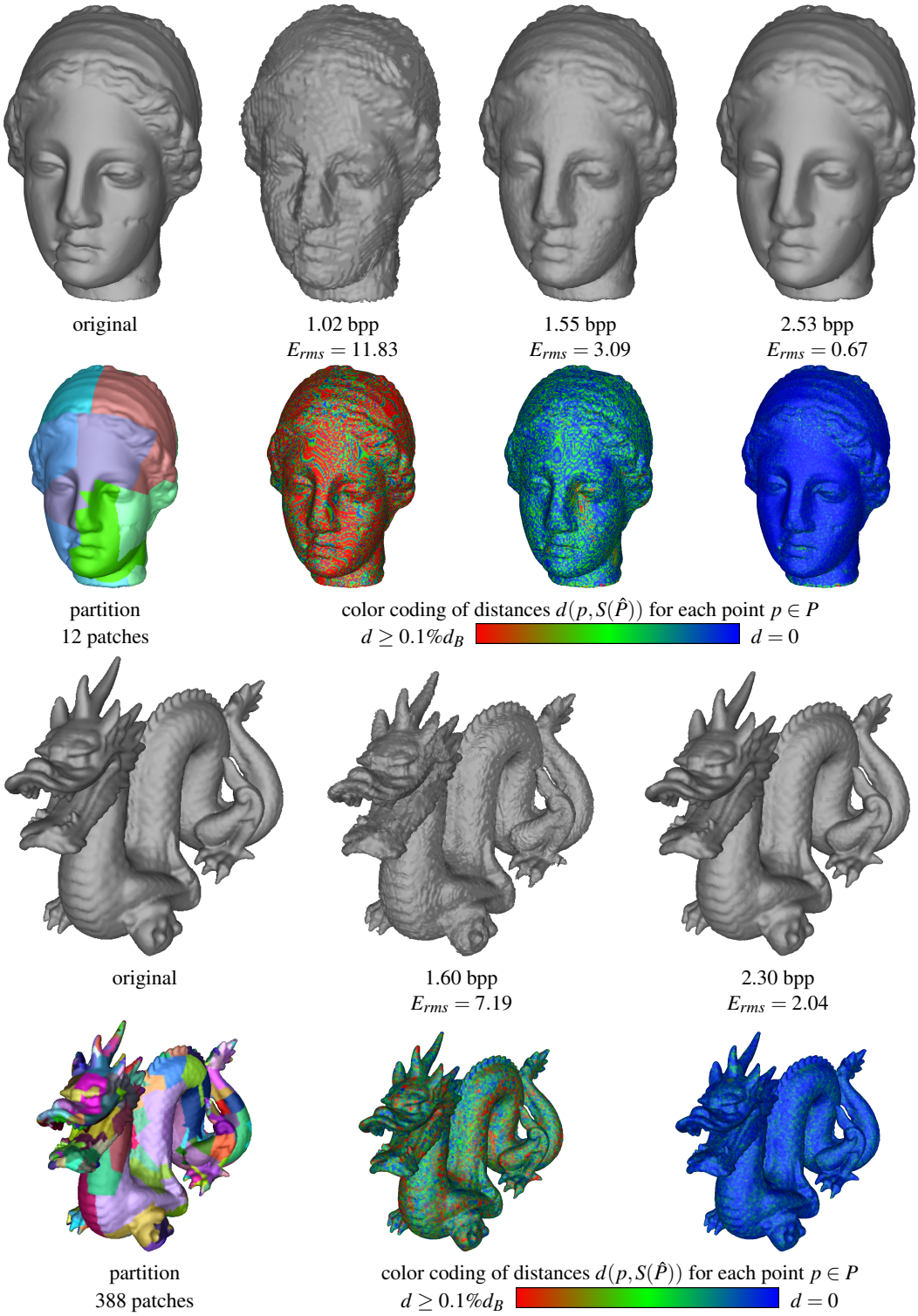

Figure 7: *Various decodings for 'Igea' and 'Dragon'. The first and third line show from left to right, the original and three respective decoded models; second and fourth rows show the used patch layout and colored (original) models and visualize the distance from each point to the MLS surface of the reconstructed point set above, $d(p, \hat{S}), p \in P$; red, green, and blue indicate large, medium and small errors respectively.*

5. Conclusions and future work

In this work we have presented a method for compressing point based 3D models that partitions a surface into patches, each of which is represented as a height field. Our coder is fast and efficient yielding better rate-distortion performance than the recently proposed coder in [9] and similar results to those of state-of-the-art mesh-compression. There is a number of improvements we will address in future work:

- the rate-distortion optimization can be extended to include the partitioning method and the selection of the resampling resolution,
- the encoder output can be reorganized to consist of a base layer and a progressive enhancement layer,
- the numerics for the resampling can be accelerated using appropriate techniques such as continuation methods,
- out-of-core techniques for extremely large models should be developed.

We also plan to give a detailed comparison of the performance of our coder with tree-based methods [3, 33, 7] and mesh-based coders [20, 22, 24].

Acknowledgments

We thank the workgroup of Pointshop3D for providing software and fruitful discussions. Special thanks go to Jim Fowler for maintaining the QccPack library containing wavelet analysis and coding software. We also thank the reviewers for their useful comments. This work was supported by a grant of the Kurt Lion foundation.

References

[1] P. Alliez, M. Desbrun, Valence-driven connectivity encoding for 3D meshes, Proceedings Eurographics 2001, Computer Graphics Forum 20(3), pp. 480–489, 2001.

[2] M. Alexa, J. Behr, D. Cohen-Or, S. Fleischman, D. Levin, C. Silva, Computing and rendering point set surfaces, IEEE TVCG 9(1), 2003.

[3] M. Botsch, A. Wiratanaya, L. Kobbelt, Efficient high quality rendering of point sampled geometry, Eurographics Workshop on Rendering, 2002.

[4] P. Chou, T. Meng, Vertex data compression through vector quantization. IEEE Transactions on Visualization and Computer Graphics, 8(4), pp. 373–382, 2002.

[5] P. Cignoni, C. Rocchini, R. Scopigno, Metro: measuring error on simplified surfaces, Computer Graphics Forum, 17(2), pp. 167–174, 1998.

[6] Cyberware sample models archive, http://www.cyberware.com/samples/

[7] O. Devillers, P. Gandoin, Geometric compression for interactive transmission, Proceedings IEEE Visualization 2000, pp. 271–326, 2000.

[8] H. Everett, Generalized Lagrange multiplier method for solving problems of optimum allocation of resources, Operations Research, vol. 11, pp. 399–417, 1963.

[9] S. Fleishman, D. Cohen-Or, M. Alexa, C. T. Silva, Progressive point set surfaces, ACM Transactions on Graphics, vol. 22(4), 2003.

[10] M. Floater, K. Hormann, Surface parameterization: a tutorial and survey, Cambridge Workshop on Multiresolution in Geometric Modelling, 2003.

[11] J. Fowler, Shape-adaptive tarp coding, Proc. IEEE Intern. Conf. Image Proc. (ICIP2003), vol. 1, 2003

[12] B. Gärnter, Fast and robust smallest enclosing balls, Proceedings of the 7th Annual European Symposium on Algorithms, pp. 325–338, 1999.

[13] X. Gu, S. Gortler, H. Hoppe, Geometry images, Proceedings SIGGRAPH 2002, ACM Computer Graphics, pp. 355–361, 2002.

[14] I. Guskov, K. Vidimce, W. Sweldens, P. Schröder, Normal meshes, Proceedings SIGGRAPH 2000, ACM Computer Graphics, pp. 95–102, 2000.

[15] J. Grossman, W. Dally, Point sample rendering, Proceedings Rendering Techniques 1998, pp. 181-192, 1998.

[16] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, W. Stuetzle, Surface reconstruction from unorganized points, Proc. SIGGRAPH 1992, ACM Computer Graphics, pp. 71–78, 1992.

[17] H. Hoppe, E. Praun, Shape compression using spherical geometry images, Multiresolution in Geometric Modeling, 2003.

[18] M. Isenburg, S. Gumhold, Out-of-core compression for gigantic polygon meshes, Proc. SIGGRAPH 2003, ACM Computer Graphics, pp. 935–942, 2003.

[19] Z. Karni, C. Gotsman, Spectral compression of mesh geometry, Proceedings SIGGRAPH 2000, ACM Computer Graphics, pp. 279–286, 2000.

[20] A. Khodakovsky, P. Schröder, W. Sweldens, Progressive geometry compression, Proceedings SIGGRAPH 2000, ACM Computer Graphics, pp. 271–278, 2000.

[21] A. Khodakovsky, P. Alliez, M. Desbrun, P. Schröder, Near-optimal connectivity encoding of 2-manifold polygon meshes, Graphical Models 64, 147-168 (2002).

[22] A. Khodakovsky, I. Guskov, Compression of normal meshes, Geometric Modeling for Scientific Visualization, Springer-Verlag, Heidelberg, Germany, 2002.

[23] G. G. Langdon, J. Rissanen, Compression of black-white images with arithmetic coding. IEEE Trans. Communications, 29(6), pp. 858-867, 1981.

[24] S. Lavu, H. Choi, R. Baraniuk, Geometry compression of normal meshes using rate-distortion algorithms, Eurographics Symposium on Geometry Processing 2003, Aachen, pp. 52–61, 2003.

[25] A. Lee, W. Sweldens, P. Schröder, L. Cowsar, D. Dobkin, MAPS: multiresolution adaptive parameterization of surfaces, Proceedingss SIGGRAPH 1998, ACM Computer Graphics, pp. 95–104, 1998.

[26] D. Levin, Mesh-independent surface interpolation, Geometric Modeling for Scientific Visualization, (Brunnett, Hamann and Müller, Eds.), Springer-Verlag, 2003.

[27] S. Li, W. Li, Shape-adaptive discrete wavelet transforms for arbitrary shaped visual object coding, IEEE Trans. Cir. Sys. Video Tech, 10(5), pp. 725–743, 2000.

[28] Z. Liu, J. Hua, Z. Xiong, K. Castleman, Lossy-to-lossless ROI coding of chromosome images using modified SPIHT and EBCOT, Proceedings 1st IEEE International Symposium on Biomedical Imaging, pp 317–320, 2002.

[29] G. Minami, Z. Xiong, A. Wang, S. Mehrotra, 3-D wavelet coding of video with arbitrary regions of support, IEEE Transactions on Circuits and Systems for Video Technology, 11(9), 2001.

[30] M. Pauly, M. Gross, Spectral processing of point-sampled geometry, Proceedings SIGGRAPH 2001, ACM Computer Graphics, pp. 379–386, 2001.

[31] M. Pauly, M. Gross, L. P. Kobbelt. Efficient simplification of point-sampled surfaces. *IEEE Visualization 2002*, Boston, 2002.

[32] F. Payan, M. Antonini, Multiresolution 3D mesh compression, Proceedings of IEEE International Conference in Image Processing (ICIP), 2002.

[33] J. Peng, C.-C. Kuo, Octree-based progressive geometry encoder, Internet Multimedia Management Systems IV, Proceedings of the SPIE, Volume 5242, pp. 301–311, 2003

[34] H. Pfister, M. Zwicker, J. van Baar, M. Gross, Surfels: Surface Elements as Rendering Primitives, Proceedings *SIGGRAPH-2000*, pp. 335–342, 2000.

[35] E. Praun, H. Hoppe, Spherical parameterization and remeshing, Proc. SIGGRAPH 2003, ACM Computer Graphics, 2003.

[36] J. Rossignac. Edgebreaker: Connectivity Compression for Triangle Meshes, *IEEE Transactions on Visualization and Computer Graphics* **5**(4), pp. 47–61, 1999.

[37] S. Rusinkiewicz, M. Levoy, QSplat: a multiresolution point rendering system for large meshes, Proceedings SIGGRAPH 2000, ACM Computer Graphics, pp. 343–352, 2000.

[38] H. Samet, A. Kochut. Octree approximation and compression methods, 1st International Symposium on 3D Data Processing Visualization and Transmission, *3DPVT-2002*, Padova, Italy, 2002.

[39] P. Sander, Z. Wood, S. Gortler, J. Snyder, H. Hoppe, Multi-chart geometry images, Eurographics Symposium on Geometry Processing 2003, Aachen, pp. 146–154, 2003.

[40] O. Sorkine, D. Cohen-Or, R. Goldenthal, D. Lischinski, Bounded-distortion piecewise mesh paramterization, IEEE Visualization'02, 2002.

[41] M. Stamminger, G. Drettakis, Interactive sampling and rendering for complex and procedural geometry, Proceedings Rendering Techniques 2001, Springer-Verlag, 2001.

[42] A. Szymczak, J. Rossignac, D. King, Piecewise regular meshes: Construction and compression, Graphical Models, V. 64, Issue 3/4, pp. 183–198, 2002

[43] G. Taubin, J. Rossignac, Geometric compression through topological surgery, ACM Transactions on Graphics, vol. 17(2), pp. 84–115, 1998.

[44] C. Touma, C. Gotsman, Triangle mesh compression, Proc. Graphics Interface 1998, pp. 26–34. 1998.

[45] G. T. Toussaint, Solving geometric problems with the rotating calipers, Procedings IEEE MELECON '83, pp. A10.02/1-4, 1983.

[46] M. Zwicker, H. Pfister, J. van Baar, M. Gross, Surface splatting, Proceedings SIGGRAPH 2001, pp. 371–378, 2001.

[47] M. Zwicker, M. Pauly, O. Knoll, M. Gross, Pointshop 3D: An Interactive System for Point-Based Surface Editing, *Proceedings SIGGRAPH-2002*, San Antonio, 2002.