Volume ? (?), Number ? pp. 1–15 preprint

Image-Based Surface Compression

Tilo Ochotta
University of Utah

Dietmar Saupe
University of Konstanz

Abstract
We present a generic framework for compression of densely sampled 3D surfaces in order to satisfy the increasing demand for storing large amounts of 3D content. We decompose a given surface into patches that are parameterized as elevation maps over planar domains and resampled on regular grids. The resulting shaped images are encoded using a state-of-the-art wavelet image coder. We show that our method is not only applicable to mesh- and point-based geometry, but also outperforms current surface encoders for both primitives.

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

1. Introduction

Surface compression has become an important field of study in graphics with the increasing demand on accessing 3D content. A surface encoder transforms an explicit surface representation into a compact bit-stream, which is then decoded at the receiver to generate a surface reconstruction. Since the direct encoding of the standard representation of 3D surfaces that are given by lists of (x,y,z)-coordinates and connectivity, leads to highly correlated data streams, more sophisticated methods are needed to decorrelate the data. In particular, the requirements of a practical and versatile compression scheme are:

Effectiveness - the encoder should produce bit-streams as compact as possible.
Efficiency - encoding and decoding should be efficient; especially the capability of fast decoding is required when the data need to be accessed in real-time.
Simplicity - the algorithmic design of both, encoder as well as decoder, should be simple and easy to implement on common platforms.

We propose a compression scheme that is capable of processing densely sampled surfaces. To approximate a given surface, the encoder decomposes the model into a set of patches, each of which is parameterized as an elevation map over a planar domain and resampled on a regular grid. The resulting representation consists of a set of images with arbitrary regions of support. The encoder compresses the shapes by using a binary image coder and the elevation images using a state-of-the-art shape-adaptive wavelet coder. The decoder extracts the binary images and reconstructs the elevation images by wavelet synthesis, followed by performing the deparameterization. Figure 1 shows the Igea model with 134k points with a partition of 7 patches and reconstructed models. Figure 2 summarizes our compression pipeline.

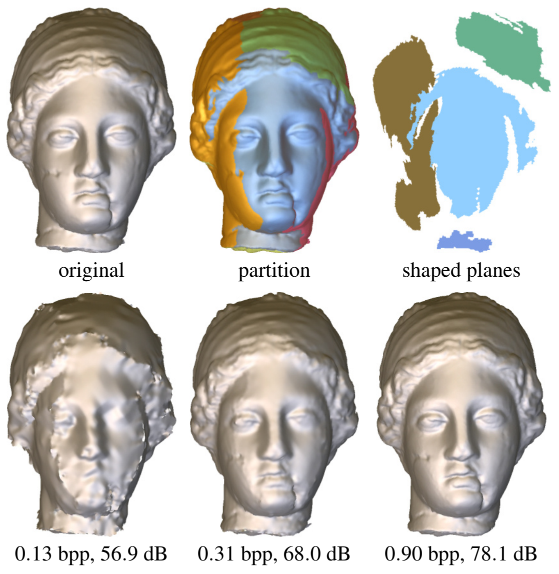

Figure 1: *Point-surface compression of Igea; top row: original model, partition, and the base planes for parameterization; bottom row: reconstructed models at various bit-rates in bits per point (bpp) and decoding fidelity in dB.*

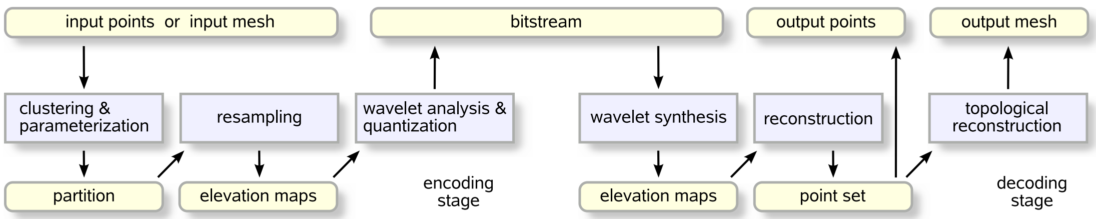

Figure 2: *Overview of our compression pipeline; the original model is given either as a point-based model or as a mesh; the encoder constructs a parameterization of the input model and processes the resulting shaped images with a shape-adaptive image coder; the model is reconstructed by decoding the bit-stream and combining the elevation maps to a complete surface; the output model can be reconstructed and rendered either as a point-based model or again as a mesh.*

Our compression framework fulfills the mentioned requirements as follows:

Effectiveness - in our system, we are able to profit from well developed existing image compression methods. Our method outperforms current approaches for compression of point-sampled geometry and is competitive with state-of-the-art mesh compression schemes, such as normal mesh compression.

Efficiency - due to the simplicity of the parameterization and efficient implementation of the wavelet synthesis, we obtain the decoded model in short time. We can reconstruct the Dragon model of about 460k points within less than two seconds on a common PC platform.

Simplicity - all stages in our pipeline rely on simple algorithms that can easily be implemented. In particular, we use an elementary parameterization in order to represent 3D surface patches as 2D images. The mapping of a point in parameter space into 3D space is performed by a single matrix multiplication.

Moreover, our method has the following features: Firstly, the capability to compress mesh- and point-based model using a unified framework, which allows selecting the appropriate primitive for each application. Secondly, the capability to produce approximations that rely on regular grids, whose resolution can be adjusted by the user or by the encoder itself. This allows for choosing the appropriate sampling density for a specific application. Thirdly, the capability to reduce the problem of 3D geometry compression to 2D compression; consequently, we can utilize existing high-performance methods from image coding.

In this work, we extend our method [OS04] in that we report much improved results for point-sampled geometry and extend our approach to mesh coding to demonstrate a surface coder that is independent of the underlying primitive. Additionally, we incorporate the variational shape approximation scheme [AG04], which improves the overall performance. We start with reviewing previous work in mesh- and point-based compression of 3D geometry in Section 2. In Section 3 and 4, we detail our pipeline. Results are presented in Section 5, followed by conclusions in Section 6.

2. Related Work

2.1. Mesh Geometry Compression

3D model compression was studied since Deering [Dee95] proposed to quantize all components of input mesh data to a certain number of bits. In practice, most computer graphics systems focus on polygon meshes, for which there are three kinds of data to transmit: geometry, connectivity, and attributes. Methods are needed to compress all three types of data. However, research has focused mainly on the geometry of the models, since it makes the largest part of the data. We now give a very brief overview of mesh coding techniques, for a more detailed review see [AG04].

In Taubin and Rossignac [TR98, Ros99] and Touma and Gotsman [TG98], the connectivity of the mesh is encoded separately. Vertex locations are predicted and the prediction residuals are entropy coded or vector quantized [CM02]. Connectivity coding was greatly improved [KPRW05], and out-of-core geometry coding for extremely large meshes was developed [IG03]. Karni and Gotsman [KG00] transform the geometry into a frequency domain by performing Laplacian analysis, followed by quantization of spectral coefficients. The reconstructed models contain less high-frequency components but still guarantee a high visual quality. Sorkine *et al.* [SCOT03] introduce an analog spectral quantization scheme, but they rather focus on removing low-frequency components that are not visible to the observer.

In some papers it was recognized that one could achieve excellent coding results when well developed methods from image wavelet coding are applied [KSS00, KG02, LCB03, SRK02]. 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. The coordinate data at the semi-regular mesh is three-dimensional by nature. Nonetheless, the construction of the multi-resolution semi-regular mesh can be organized so that only normal displacements are necessary to define the newly generated vertices of the next finer level, yielding so-called *normal meshes* [KG02]. In practice, however, the subdivision is algorithmically complex and not all detail displacement vectors can be expressed by a single scalar value.

2.2. Point-Based Methods

In 3D model acquisition [LPC*00], range scanners generate point clouds that can be rendered directly without triangulation, i.e. using surface elements (surfels, [PZvBG00]), which correspond to disks that locally approximate the surface for hole-free point-based renderings. Rusinkiewicz *et al.* [RL00] and Botsch *et al.* [BWK02] proposed real-time software renderers that provide efficient encodings of the data based on approximating the original points through centroids of occupied octree cells. Nevertheless, these approaches rely on the capability of rapid decoding to allow fast rendering that forbids employing enhanced and complex encoding strategies in order to achieve a compression performance comparable to that of mesh coders, e.g., [KSS00, KG02].

Some of the mesh compression strategies that are based on quantization of vertex coordinates without using connectivity are also suitable for compression of point sets, e.g., [Dee95, PK05]. However, the best performing methods [KG02] rely on a complete resampling of the surface, which requires an appropriate surface representation if the input data is only a point set. A direct meshing of the input point set is too difficult to construct or in most cases too expensive and thus motivates methods that achieve comparable compression performance for point sets without meshing.

In [FCOAS03] a multi-resolution compression scheme for point-sampled geometry was proposed that is based on an embedded sequence of point sets, starting with a base point set, which is encoded using a standard mesh compression method. Point set refinement is performed using so called moving least squares approximation over local coordinate frames. Points are inserted combining estimated point positions and encoded Δ-values. Waschbüsch *et al.* [WGE*04] parameterize a given point-based model in a hierarchical tree structure in order to recursively predict point positions from corresponding tree nodes. The tree is processed by the SPIHT encoder [SP96], yielding compact bit-streams. In addition, point positions, attributes, e.g., normal vectors, and color can be compressed.

A joint compression and rendering framework was proposed in *DuoDecim* by Krüger *et al.* [KSW05]. The point coordinates were quantized and encoded relatively to each other using a decomposition of the point set into runs. The resulting bit-stream was decoded on graphics hardware, which makes this method perfectly appropriate for real-time applications and for processing of very dense data sets.

2.3. Patch-Based Parameterization

Surface parameterization is an active field in geometry processing, for an overview see [FH05]. A parameterization of a surface is a mapping of a parameter space (subset in $\mathbb{R}^2$) to the 3D space. Lee *et al.* [LMH00] presented displaced subdivision surfaces, where the original data set is approximated by a smooth control mesh, which in turn is displaced by a scalar valued map. Sander *et al.* [SWG*03] proposed to represent a mesh by an atlas of charts that are designed to minimize distortion. The detail coefficients are three-dimensional and thereby, the parameterization is more complex, since it does not merely displace points orthogonally to a plane.

Ivanov and Kuzmin [IK01] proposed *spatial patches* as rendering primitive for meshes. The model is approximated by an atlas of regularly sampled charts that correspond to height fields, properly placed in 3D space. A similar representation for point-based models has been proposed by Pauly and Gross [PG01] for the purpose of spectral filtering. Because their height fields are resampled to complete rectangular images, which results in heavy chart overlapping, a blending technique is necessary when merging the processed patches for surface reconstruction.

Focusing on efficient surface partitionings, Cohen-Steiner *et al.* [CSAD04] proposed to cluster the surface and approximate it through a set of polygonal shaped proxies, each of which is equipped with a normal vector and a centroid, yielding an arbitrarily shaped flat surface region. Considering various error metrics, they discussed a strategy to minimize a global surface approximation error based on concepts from vector quantization theory. As shown later, we utilize this method to obtain height fields with possibly low distortions.

2.4. Shape-Adaptive Image Coding

Our compression method is based on approximating a given geometric model by a number of encoded elevation maps with arbitrary regions of support. Recently, wavelet-based shape-adaptive image coding has been proposed, in which wavelet coefficients corresponding to transparent image regions are considered insignificant [LL00]. We utilize shape-adaptive coding with binary set splitting [Fow04], which is easy to implement and yields the best rate-distortion performance in comparison to other approaches.

3. Compression Framework

We consider a given 3D geometric model as a finite point set $\mathcal{P} \subset \mathbb{R}^3$, e.g., points that were acquired by a 3D scanner. Given $\mathcal{P}$, we consider an associated surface $\mathcal{S} = \mathcal{S}(\mathcal{P})$ that provides a continuous approximation of the original surface. When the input model is a mesh, $\mathcal{S}$ is given by the piecewise linear interpolation of the points in $\mathcal{P}$. In cases where the model is only a set of points, we define $\mathcal{S}$ to be the moving least squares (MLS) surface of $\mathcal{P}$. The MLS surface is a smooth surface depending on the input point set $\mathcal{P}$ that is defined without piecewise parameterization. Computationally, a point $p \in \mathcal{S}(\mathcal{P})$ is found by applying a projection operator Ψ to an arbitrary point $q \in \mathbb{R}^3$, $p = \Psi(q)$. To compute the projection, a nonlinear equation must be solved by applying an iteration procedure. A point $p \in \mathcal{S}$ projects onto itself. Thus, the complete MLS surface is given by the set of fixed points, $S = \{p \mid p = \Psi(p)\}$. For details see [ABCO*03].

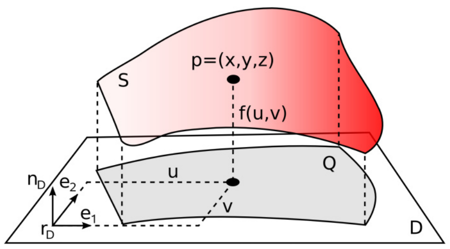

Figure 3: *Our parameterization corresponds to a mapping of points in parameter space (base plane) to the surface; three vectors e_1, e_2 and n_D build an orthonormal basis for each point over the plane, $(u, v, f(u, v))$; the orthogonal projection of the surface S onto the plane D provides a region Q which defines the support of the regularly sampled map that is constructed in our framework.*

3.1. Error Measure

When compressing a surface given by a point set $\mathcal{P}$, an encoder produces a bit-stream that is converted back into another point set $\hat{\mathcal{P}}$ at the decoder. In order to evaluate the quality of the approximation we follow previous work [KG02, LCB03] in the spirit of METRO [CRS98] and define a symmetric root-mean-square (rms) error and a peak-signal-to-noise ratio (PSNR) based on surfaces $\mathcal{S}$ and $\hat{\mathcal{S}}$ that are given by $\mathcal{P}$ and $\hat{\mathcal{P}}$ respectively.

The (one-sided) L_2-distance of $\hat{\mathcal{S}}$ as an approximation of $\mathcal{S}$ is given by

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

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

$$E_{rms}(\hat{\mathcal{S}}, \mathcal{S}) = \max\left(d(\hat{\mathcal{S}}, \mathcal{S}), d(\mathcal{S}, \hat{\mathcal{S}})\right). \tag{2}$$

In practice $d(\hat{\mathcal{S}}, \mathcal{S})$ is computed by sampling the surface $\hat{\mathcal{S}}$ and averaging squared distances. The actual distances $d(p, \mathcal{S})$ are estimated by applying the projection operator, $d(p, \mathcal{S}) \approx ||p - \Psi(p)||$, or are computed explicitly when $\mathcal{S}$ is defined by a mesh.

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

3.2. Parameterization

Given a subset $S \subseteq \mathcal{S}$ (a surface patch of $\mathcal{S}$), the goal of the parameterization is to find a mapping from a planar domain to 3D space, such that its graph approximates the patch. Given patch S, we consider a *reference plane* $D = D(S)$ defined by a normal vector n_D and a reference point r_D, $D = \{x \in \mathbb{R}^3 | \langle n_D, x - r_D \rangle = 0\}$, where $\langle \cdot, \cdot \rangle$ denotes the scalar product. A mandatory condition for S to be a height field is that the angles between the normal vectors $n_S(p), p \in S$ and the plane normal n_D do not exceed $90°$,

$$A(S) = \inf_{p \in S} (1 - \frac{1}{2} \|n_S(p) - n_D\|^2) \geq 0, \tag{3}$$

which follows from the cosine theorem. The aperture $A(S)$ is the cosine of the aperture angle of the normal cone defined by the normals on S. In practice we allow surface patches S with a decreased aperture angle, $A(S) \geq \varepsilon$, with $\varepsilon = \cos \alpha$, $0° < \alpha < 90°$. For computation of n_D, we use smallest enclosing balls [Gär99], which can be utilized to compute the normal cone, consisting of the normal vector n_D and an aperture angle ϕ. We set n_D to the normalized output of the algorithm in [Gär99], $\arg\min_{n \in \mathbb{R}^3} \max_{p \in S} \|n_S(p) - n\|$.

Given the plane $D = D(S)$, we consider the orthogonal projection of the surface patch S onto D, yielding $Q \subset D$, which we refer to as *the support* Q of surface patch S over plane D, see Figure 3. We now define a local coordinate system with origin in r_D and two span vectors e_1 and e_2 on D that form an orthonormal system, providing (u, v)- (or parameter-) coordinates for any point on D. We may choose scaling factors l_1, l_2, such that $l_1 e_1$, $l_2 e_2$, and the reference point r_D define a bounding box of the support Q on the plane. Additionally, we define $\hat{Q}$ as the set of (u, v)-parameter coordinates with corresponding points in Q.

The local coordinate system allows to express any point p on the surface patch S in parameter coordinates (u, v), applying an elevation function $f : \hat{Q} \to \mathbb{R}$. Any point $(u, v, f(u, v))$ with $(u, v) \in \hat{Q}$ can be thought of as an elevated point over the plane that corresponds to a point on the original surface patch S, see Figure 3.

3.3. Partitioning

The goal of the partitioning procedure is to decompose the surface $\mathcal{S}$ into a set of patches, such that each patch can be parameterized as an elevation map. The patches are collected in a partition $\mathcal{R} = \{S_i\}, \cup_i S_i = \mathcal{S}, S_i \cap S_j = \emptyset, i \neq j$.

We interpret the problem of finding such a partition as a clustering problem, where each cluster has to fulfill condition (3). In many clustering scenarios, the input data set is grouped such that for a given number of clusters, a local minimum of a certain global cost function is found [GG92]. In our setting, the cost function $\mathcal{J}$ for a given partition $\mathcal{R}$ should capture the encoding cost. We integrate the squared differences $||n_S - n_{D_i}||^2$, taking over the normal vectors $n_{S_i}(x)$ belonging to all points x on each surface patch S_i, and where $n_{D(S)}$ again denotes the normal vector of the corresponding reference plane $D(S)$,

$$\begin{aligned} \mathcal{J}(\mathcal{R}) &= \sum_{S \in \mathcal{R}} J(S), \quad \text{with} \\ J(S) &= \int_{x \in S} \|n_S(x) - n_{D(S)}\|^2 dx. \end{aligned} \tag{4}$$

The motivation behind this definition is the expectation that large normal variations lead to height fields, which are rough and, consequently require a higher encoding rate. For the evaluation of $J(S)$, we approximate the integral by a finite sum of appropriately weighted contributions at uniformly distributed sample points on the surface patch S.

In the following we describe and compare two partitioning methods that we utilize to appropriately cluster the input surface into patches.

Split-Merge Clustering

In the first version, we adopt the split-merge approach from our previous work [OS04]. In a nutshell, the complete surface initially belongs to one cluster, which is recursively split using principal component analysis until all clusters satisfy the normal cone condition. We use some fixed threshold $\alpha < 90^\circ$ as indicated in Section 3.2. After the splitting phase, adjacent clusters are merged as long as the the resulting merged surfaces fulfill the normal cone condition. In contrast to Equation (3), where the parameter α is chosen to adjust the maximum cone aperture angle, we choose a relaxed threshold β during the merging process, $\beta \geq \alpha$. The merging operations are implemented using a priority queue, which sorts possible merging candidates after increasing error increments. For details, see [OS04].

The parameters α and β are used to adjust the properties of the final partition. In general, a small α (e.g., $\alpha = 10^\circ$) leads to a partition with many small clusters after splitting. Applying a large angle β (e.g., $\beta = 70^\circ$) during merging leads to a smaller number of clusters in the final partition. However, the patches in this partition show irregular boundary shapes, see Figure 4(b), which may lead to increased coding costs. Increasing the parameter α reduces the number of clusters in the initial partition and produces final partitions where the patches have straight boundary curves, see Figure 4(d).

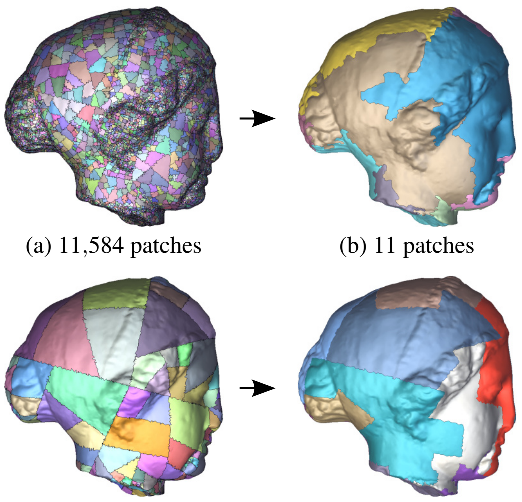

Figure 4: *Split-merge partitions: a large patch number after splitting leads to irregular boundaries after merging, (a,b); a smaller patch number after splitting straightens the boundary shapes but slightly increases the final patch count, (c,d).*

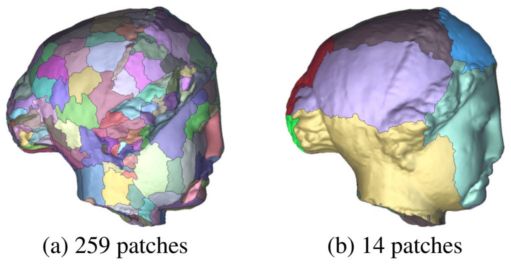

Figure 5: *Partitions obtained using the Generalized Lloyd's Algorithm for surfaces [CSAD04] for 259 patches (a) and 14 patches (b); in contrast to the split-merge, the patches do not show straight boundary shapes.*

Generalized Lloyd's Algorithm (GLA)

In [CSAD04] it was shown that excellent partitionings can be obtained using methods from vector quantization. In particular they applied Lloyd's algorithm [Llo82] to find a local minimum of the cost function (4). We adopt their method and briefly describe the procedure.

The search for the optimal partition is initiated by a number of randomly chosen seed triangles (or points) on the surface that correspond to cluster centroids, each of which is defined by a location on the surface and a normal vector and coherently defines a plane (similar to the reference plane from Section 3.2). The minimization of (4) is implemented in two steps that are iteratively applied and work as follows.

Cluster Flooding. Given the set of triangles that represent the cluster centroids, the goal is to assign all remaining triangles of the model to clusters in order to build a surface partition. This is achieved by employing a flooding technique, in which triangles are subsequently assigned to the clusters, using a priority queue. Specifically, for each centroid triangle consider all (three) adjacent triangles and sort each of them into a global priority queue according to the distance of its normal vector to the normal vector of the centroid. Then, remove the first triangle from the queue and assign it to the respective cluster, and insert further (at most two) adjacent triangles into the priority queue that have not yet been visited. This procedure is repeated until the queue is empty, meaning that all triangles were assigned to a cluster.

Centroid Fitting. Once a partition is found, the cluster proxy (plane) of each patch is refitted in the spirit of Lloyd iterations [Llo82]. Specifically, given a cluster of triangles, i.e. a surface patch S, the normal n_D of the plane $D(S)$ is set to the average of the normal vectors of the triangles in S, weighted according to the triangle areas. The plane reference point is formally set to the barycenter of S, although it does not affect the approximation error (4).

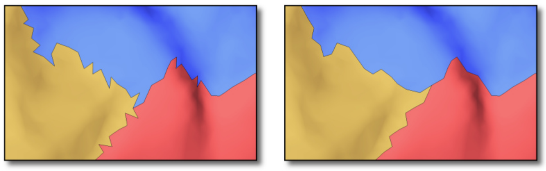

Figure 6: *A partition with unoptimized shape shows noisy edges (left); triangle reassignemts based on boundary length minimization reduces these artifacts (right).*

The procedure of cluster flooding and centroid fitting is repeated until convergence or after a given number of iterations is exceeded. Results of this partitioning method are shown in Figure 5. In contrast to the split-merge approach, the GLA distributes the patches on the surface more evenly, compare Figure 4(c) and Figure 5(a). We furthermore observed that for most tested models, the GLA achieves a smaller number of patches than the split-merge approach under a given patch aperture constraint, Equation (3).

Boundary Shape Optimization

The overall compression performance depends on the structure of the elevation maps in two ways. Firstly, elevation maps with large total variance in elevation values are disadvantageous for compression. This issue is addressed by minimizing (4). Secondly, complex and irregular boundary shapes produce higher coding costs for the binary masks than shapes with more regular boundaries.

To complete the surface partitioning, we remove noise artifacts at patch boundaries by decreasing the total boundary length. For any triangle with an edge along a patch boundary, we compute the decrease in boundary length when reassigning the triangle to the adjacent patch. In an iterative procedure, we reassign triangles as long as there are candidates that decrease the boundary length. Note that each reassignment step may affect further reassignments of boundary triangles in a local neighborhood. For efficient processing, we maintain a priority queue that holds the candidates according to their length decreases.

In addition to reassigning single triangles, we allow reassignments of connected triangle pairs, which improves the boundary denoising. This method may also be extended to triangle clusters of more than two triangles. Figure 6 shows that the shape optimization adequately removes noise artifacts at the patch boundaries.

3.4. Resampling

After constructing the partition, we consider the projections of the original points onto their respective patches, which yields an arbitrary point distribution for each patch. In order to apply the shape-adaptive wavelet coder, however, each surface patch needs to be resampled on a regular grid.

Given patch $S \in \mathcal{R}$, we define a sampling interval Δ_S and consider grid points in (u,v)-space,

$$G_S = \{q = (\Delta_S i, \Delta_S j) | (i,j) \in \mathbb{Z}^2, q \in \hat{Q}\}$$

that belong to the patch S. For each patch S, we compute the sampling interval Δ_S^0 such that the number of resampled points, $|G_S|$, is roughly equal to the number of original points in the patch. In order to control the total number of resampled points in the model, we set the sampling interval in each patch S to $\Delta_S = \mu \Delta_S^0$, where $\mu > 0$ is a scaling parameter. Consequently, a global parameter of $\mu = 1$ adjusts each grid such that the number of points in the resampled model is roughly equal to the number of original points.

In the resampling procedure, we compute for each grid point $q \in G_S$ a corresponding elevation value $f(q)$. If the original surface $\mathcal{S}$ is a mesh, this can be achieved by projecting the original triangles onto the patch plane and identifying the triangle, which contains the grid point on the plane. The elevation value can then directly be interpolated from the elevation values of the triangle vertices. In cases where $\mathcal{S}$ is a MLS surface, a more sophisticated method has to be employed as follows.

Similar the work in [AA03], the elevation values $f(q)$ are computed iteratively with a Newton-like method using the MLS projection operator Ψ (see Section 3). Figure 7 illustrates the procedure. For a given point q on the grid G_S, we consider the corresponding point $p \in \mathbb{R}^3$ on the plane and the line r through p and orthogonal to the plane. To initialize the iterative procedure, we project the nearest neighbor of p in $\mathcal{P}$ onto r, yielding p_1. Then, we project p_1 onto the MLS surface by applying Ψ and compute the tangential plane of the surface at the projection. Then the intersection of this tangential plane with the line r defines the new point p_2, which is used for the next MLS projection. This procedure is iterated until convergence, yielding the point $\Psi'(p)$ on the MLS surface and the desired functional value $f(q)$ for the grid point q. Applying this procedure to all grid points and patches yields the complete resampled point model $\hat{\mathcal{P}}$.

3.5. Encoding

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 side information, binary masks, and the scalar height field values as follows.

Side information; consisting of the base plane rectangle and its grid sampling resolution for each patch. The 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.

Binary masks; defining the support of the height field on each base plane rectangle. There are many methods for bitmap encoding. In our implementation we have used an

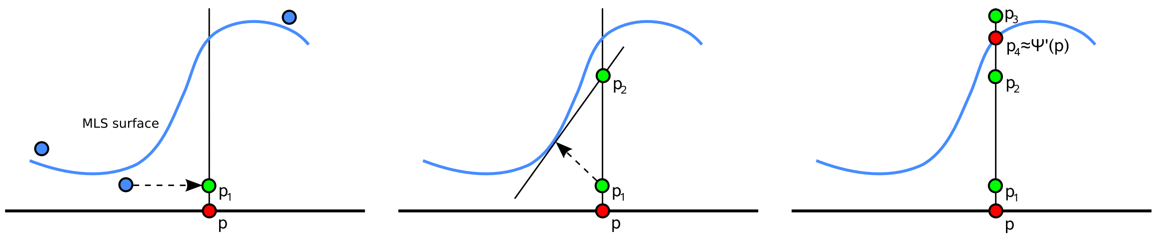

Figure 7: *Overview of the MLS-based resampling technique for point-based input models; the original point that is closest to grid point p is projected onto the line through p and normal of the base plane, yielding p_1 (left); the tangential plane of the surface at the projection $\Psi(p_1)$ is intersected with the line, providing p_2 (middle); the procedure is iterated until convergence, yielding the projected point $\Psi'(p)$ over the point p (right).*

adaptive context-based arithmetic coder [LR81] for the binary masks of the height field supports.

Elevation maps; consist of the scalar height field values. Our height fields are regularly sampled but have irregularly shaped supports. Thus, they can be regarded as grey scale images with irregular boundaries. In image compression, efficient methods for shape-adaptive image coding were developed [LL00]. The most successful methods are wavelet-based using bitplane coding with embedded tree-structured significance mapping as in the image coder SPIHT [SP96]. Fowler proposed such a coder, called BISK [Fow04], that showed superior performance and has been made available as part of QccPack library [Fow00]. We used this coder in our implementation for image-based surface compression.

In a nutshell, the BISK coder operates as follows. The input consists of a rectangular height field with the binary mask defining the support of the height field as side information. The height field is wavelet transformed using the 4-scale shape-adaptive 9/7 biorthogonal filter [LL00]. For the "opaque" coefficients, an embedded bitplane code is produced applying the paradigm of significance and refinement passes, which is common to most wavelet-based image coders. Initially, all coefficients are taken as insignificant. In a significance pass, coefficients with a magnitude greater than a threshold corresponding to the current bitplane are identified, marked as significant, and the sign of each of these coefficients is coded. In the refinement pass, for a bitplane, one additional bit for each previously marked significant coefficient is coded. In significance passes, entire sets of coefficients are tested jointly, and significant sets are recursively subdivided. In the BISK coder, this subdivision is a binary one, yielding a kd-tree subdivision of each coefficient band, and thus, it adapts to the shape of the image (resp. height field). For details see [Fow04].

3.6. Bit Allocation

Given a set of resampled patches, approximating the original surface, the shape-adaptive wavelet coder yields an embedded bit-stream for each individual patch. Consequently, for each patch the output bit-stream may be truncated providing a certain reconstruction quality at the decoder.

The straightforward bit allocation strategy is to distribute the bits among the patches evenly. Specifically, each resampled patch $\hat{P}_S$, $S \in \mathcal{R}$ receives a total number of

$$\frac{|\hat{P}_S|}{|\hat{\mathcal{P}}|}B \quad \text{bits,} \tag{5}$$

where B is the total bit budget given by the user and $|\hat{P}_S|$ and $|\hat{\mathcal{P}}|$ denote the number of resampled points in patch S and the complete reconstructed model $\hat{\mathcal{P}}$, respectively. This allocation approach is straightforward, and has the advantage that the complete bit-stream can easily be designed as an *embedded* stream by interleaving the individual patch streams.

A more sophisticated bit allocation method is given by rate-distortion optimization, which *optimally* distributes the bits among the patches such that the overall reconstruction quality is optimal. We achieve this by employing standard discrete Lagrangian optimization [Eve63].

For each resampled patch $\hat{P}_S$, $S \in \mathcal{R}$, let $\left(R_S^{(i)}, D_S^{(i)}\right)$, $i = 1,2,\ldots$ denote the (finitely many) rate-distortion points achievable by truncating the coder output at different positions numbered by $i = 1,2,\ldots$. The rates $R_S^{(i)}$ include bits for the base plane parameters, for the binary masks, and the truncated wavelet coefficient bit-stream. The distortion is given by the one-sided L^2 distance between the reconstructed surface patch the full original model, see Equation (1). The truncation index $i = i(S)$ for patch S is given by

$$i(S) = \arg \min_{i=1,2,\ldots} \left(R_S^{(i)} + \lambda \cdot D_S^{(i)}\right),$$

where $\lambda \geq 0$ is the Lagrangian parameter. This yields a solution with total rate $\sum_i R_S^{(i(S))}$ with minimal total square distortion $\sum_i D_S^{(i(S))}$ [Eve63]. In order to meet a prescribed total rate constraint, we vary the Lagrange parameter λ, using the bisection method.

The evaluation of the distortion $D_S^{(j)}$ for all j and $S \in \mathcal{R}$ is computationally expensive. We therefore propose to approximate the surface distortions by an image-based error measure. Specifically, we redefine

$$D_S^{(j)} = \sum_{q \in G_S} \left[f_S(q) - \hat{f}_S^{(j)}(q) \right]^2, \qquad (6)$$

where $f_S(q)$ is the elevation value of the original patch S at position (m,n), and $\hat{f}_S^{(j)}(q)$ denotes the corresponding reconstructed elevation value at bit-stream truncation index j. Equation (6) can easily be evaluated at any truncation index of the bit-stream and guarantees fast rate-distortion analysis.

While the bit allocation computes the number of bits that is assigned to each patch, we also utilize the method in order to compute the optimal resampling density. This is achieved by including the rate-distortion curves for prescribed grid resolutions and selecting the resolution of the identified rate-distortion point on the curve for each patch.

4. Postprocessing for Meshes

Our unified framework allows processing of mesh- and point-based geometry. For mesh input models, however, we consider an optimization scheme to improve the approximation quality, and a gap closing method, which provides hole-free reconstructed meshes.

4.1. Linear Least-Squares Fitting

If the input model is a mesh, the elevation values of resampled points are directly read off from the original surface $\mathcal{S}$. Together with the canonical meshing of the resampled points, this yields an interpolation of points on $\mathcal{S}$. In [HDD*93] the resulting average squared interpolation error was reduced by adjusting the resampled points appropriately, thereby turning the interpolation into an approximation. We also apply this linear optimization technique, adjusting the elevation values of the resampled points.

The optimization corresponds to the minimization of a quadratic term $\|Ax-b\|^2$ over x. The vector x holds the elevation values of the patch vertices that have to be adjusted. The vector b holds the true elevation values $f(p_i)$ of a larger number of sample points p_i that are regularly distributed over the patch plane. The matrix A holds in each line i the barycentric coordinates of sample point p_i with respect to the triangle that contains p_i. Thus, Ax contains the elevation values of the sample points p_i from the interpolation, and $\|Ax-b\|^2$ is the sum of the squared errors of all of these interpolated elevations. For an accurate approximation, the number of sample points has to be chosen sufficiently large (e.g., $\dim b = 5 \dim x$).

Least-squares optimization results are listed in Table 1. The numbers show the surface errors E_{rms} for various patch resolutions μ. The numbers in brackets are surface errors that correspond to the approximation errors before applying the least squares optimization. We observe a consistent improvement in approximation error for all resampling densities.

	$1/\mu = 0.6$	$1/\mu = 1.0$	$1/\mu = 1.4$
Venus	1.97 (2.35)	0.86 (1.12)	0.56 (0.70)
Rabbit	1.15 (1.37)	0.53 (0.63)	0.32 (0.37)
Sphere	1.80 (2.27)	0.72 (0.98)	0.53 (0.66)

Table 1: *Approximation errors E_{rms} obtained with the least-squares fitting method for Venus, Rabbit, and Sphere at various resolutions μ; the numbers in brackets correspond to approximation errors without applying the fitting method.*

4.2. Topological Reconstruction

If the model has to be reconstructed as a mesh, the regular structure of the elevation maps can be utilized to construct a mesh for each patch. We implement a canonical meshing by connecting four adjacent points in each patch by two triangles. However, the resulting mesh suffers from patch gaps due to the missing connectivity between the patches.

In the literature, there are several methods for repairing meshes with gaps that are comparable to ours. Borodin *et al.* [BNK02] proposed a gap-closing scheme, which iteratively projects vertices on the gap boundaries towards the opposing boundary and creates a triangle for each projection. In the algorithm of Sander *et al.* [SWG*03], a mesh is zippered by projecting boundary vertices onto cut paths that correspond to gap defining curves. In consecutive projection and unification operations, opposing boundary curves are projected to each other without inserting new triangles.

Both methods produce watertight high-quality models, while they rely on moving vertices on the boundaries of the patches. For the application of compression, however, we prefer to keep vertices fixed on their reconstructed positions, since they have been computed and transmitted under rate and distortion aspects. Hence, we propose a gap triangulation method, which is entirely based on triangle insertion operations under the constraint that vertices are fixed.

In this paragraph we continue to extend the description of our gap closing method in [OH06]. We initially construct the canonical mesh of the set of reconstructed points $\hat{P}_i$ of each individual patch i, yielding surfaces $\hat{S}_i$, $i = 0, 1, \ldots, |\mathcal{R}|$. Then we consider the boundary $\partial\hat{S}_i$, which is the set of points on edges that share only one triangle in the triangulation of $\hat{S}_i$ and also the boundary vertices that are incident to these edges. Our goal is to connect vertices on the border of each patch with edges on another patch. This provides new triangles that connect the patches. Similar to [BNK02], we consider projections of boundary vertices $v \in \partial\hat{S}_i$ onto $\partial\hat{S}_j$. Specifically, for each $\hat{S}_i$ and vertex $v \in \partial\hat{S}_i$ we compute

$$p_j(v) := \arg \min_{p \in \partial\hat{S}_j} \|p - v\|. \qquad (7)$$

Since $p_j(v)$ is a point on the boundary $\partial\hat{S}_j$, this point will either be on a boundary edge or a boundary vertex. In the first case we consider the pair $v_1, v_2 \in \partial\hat{S}_j$ of incident vertices to this edge. In the second case we choose one incident

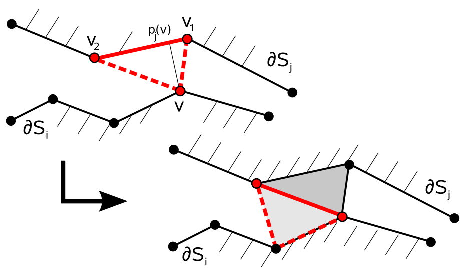

Figure 8: *Triangle insertion in our gap filling method; upper left: projection of border vertex v onto an opposing edge (v_1,v_2); lower right: triangles are inserted as long as they do not intersect with an existing part of the surface.*

edge to the vertex $p_j(v)$ and also consider the pair of incident vertices v_1 and v_2. We now form and insert a new triangle between the three vertices v, v_1 and v_2, see Figure 8. In a consecutive step we update the border information for edges and vertices that have been affected by this insertion step.

- The edge (v_1,v_2) becomes an inner edge since it is incident to two triangles.
- We have two new border edges, namely (v,v_1) and (v,v_2), if there is no previously inserted triangle with an edge (v,v_1) or (v,v_2), respectively.
- The vertex v is no longer a border vertex, but an inner vertex, once all edges incident to v have become inner edges.

When inserting a new triangle (v,v_1,v_2), we furthermore check for intersections with triangles incident to v, v_1 and v_2 to avoid overlapping triangles. This is mandatory, since in (7) there is no topological condition that prevents the triangle (v,v_1,v_2) to intersect with an existing part of the surface.

It should be noted that we also allow projections of border vertices $v \in \partial\hat{S}_i$ onto the same patch boundary $\partial\hat{S}_i$, as long as the edge that contains the projected point $p_i(v)$ is not incident to v. In our implementation, we additionally extend the formulation in Equation (7) to projections onto more than one nearby edge, which increases the number of candidates for each border edge and improves the overall performance, e.g., triangles with a large obtuse angle are not inserted.

Figure 9 shows the reconstructed Rabbit mesh without and with applying our gap filling method. We obtain robust and fast gap closing. To process the Rabbit model of 69,420 vertices and 133,129 faces, our algorithm needs less than 3 seconds for inserting 6,852 faces on a 2.8GHz PC platform.

5. Results and Discussion

We present results for the models Venus, Igea, Rabbit, Balljoint, which are available at the Cyberware repository [Cyb], and Dragon, David, XYZRGB Dragon, and Lucy, which were taken from the Stanford 3D Scanning Repository [Sta, LPC*00]. The model of the Shakyamuni statue is available at the Konstanz 3D Model Repository [KN].

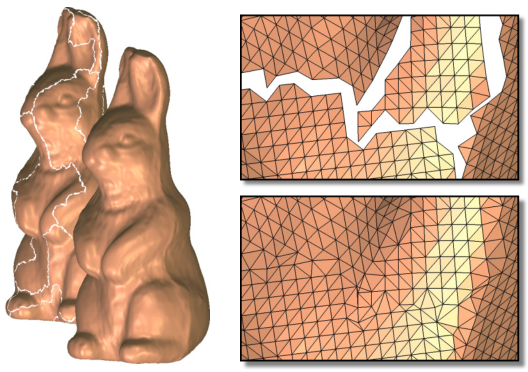

Figure 9: *Results that are obtained with our gap filling method; Rabbit without and with gap filling (left); and corresponding closeup views (upper and lower right).*

5.1. Partitioning

Table 2 lists the number of patches and the approximation error for the split-merge approach (Section 3.3) and the GLA clustering (Section 3.3). The partitions were obtained restricting the maximum aperture of each patch to 80°. Comparing the two partitioning methods, we observe that the GLA [CSAD04] outperforms the split-merge method. We obtain partitions with less patches that at the same time show a smaller approximation error, Equation (4).

In the following, we present results for point- and mesh-based models using the GLA partitioning, where we applied the aperture threshold of $\beta = 80°$. For some large models, however, we applied an extremely relaxed threshold, i.e. $\beta = 89°$ in order to reduce the total number of patches, which leads to better compression results at low bit-rates.

5.2. Point Model Compression

Table 3 lists detailed coding results for point models. For each model, the bit-stream consists of, firstly, a header, which holds side information and plane mapping parameters for each patch (the respective amount of bits is denoted by r_0); secondly, the coding costs of the binary images which

model	# points	split-merge		GLA [CSAD04]	
		# patches	error	# patches	error
Igea	134,345	14	0.42	**7**	**0.32**
Balljoint	137,062	26	0.41	**12**	**0.29**
Dragon	437,645	367	0.56	**150**	**0.22**
Shakyamuni	996,956	262	0.51	**133**	**0.23**
David	3,614,096	878	0.51	**692**	**0.09**

Table 2: *Comparison between the split-merge partitioning and the GLA clustering, (Section 3.3); the patch aperture angles are bound to 80°; the GLA method outperforms the split-merge approach by providing partitions with a smaller number of patches at a smaller approximation error, (4).*

reconstructed points	mask bits [%]	total rate [bpp]	E_{rms} $[10^{-4} d_B]$	PSNR [dB]	T_d [sec]
Igea, 134,345 points, 7 patches, $r_0 = 1{,}218$					
87,526	17.0	0.50	2.35	72.6	0.3
136,638	7.5	1.51	0.76	84.4	0.3
197,012	4.7	3.01	0.44	87.1	0.5
Balljoint, 137,062 points, 12 patches, $r_0 = 1{,}893$					
49,422	15.0	0.51	2.52	72.0	0.2
137,558	9.0	1.51	0.73	82.7	0.3
269,581	6.7	3.01	0.43	87.3	0.4
Dragon, 437,645 points, 150 patches, $r_0 = 20{,}253$					
154,189	58.4	0.29	13.3	57.5	0.8
431,633	21.5	1.55	1.03	79.7	1.2
848,723	16.6	3.05	0.62	84.2	1.7
Shakyamuni, 996,956 points, 133 patches, $r_0 = 18{,}224$					
268,089	24.3	0.22	2.50	72.0	1.6
947,262	12.6	1.02	0.32	89.9	2.2
1,366,866	7.9	2.02	0.14	97.1	2.8
David, 3,614,096 points, 692 patches, $r_0 = 93{,}693$					
562,419	33.0	0.18	1.30	77.7	4.6
1,270,573	24.9	0.38	0.45	86.9	8.2
3,544,249	18.0	0.98	0.21	93.6	12
Lucy, 14,020,068 points, 621 patches, $r_0 = 84{,}243$					
5,184,445	63.3	0.09	2.89	70.8	23
9,225,736	26.9	0.31	0.19	94.4	34
14,422,702	10.7	1.01	0.07	103	58

Table 3: *Detailed compression results for various point models; the bit-stream consists of header bits (denoted by r_0), bits for the binary images (masks), shown as percentage of the entire code length, and bits for wavelet coefficients; the last column shows the decoding and reconstruction time.*

define the elevation map supports; and thirdly, the bits for the wavelet coefficients, which constitute the largest part of the stream. The bit-rate is the total amount of bits divided by the number of input points. The error E_{rms} is the surface-to-surface distance of the original model and the reconstruction, see Section 3.1. We follow Equation (2) and sample both surfaces with a number of points that are projected onto the opposite surface. All errors in this work are expressed in units of $10^{-4} d_B$, where d_B is the diagonal length of the bounding box of the original model.

Comparing the results in Table 3, we remark two major observations. Firstly, the number of required patches is widely independent from the sampling density in the model, e.g., the Shakyamuni model with 996,956 points is twice as big as the Dragon model with 437,645 points, but needs only 133 patches, while the Dragon model takes 150 patches. This is obvious, since the patch layout reflects the complexity of the geometry rather than the sampling density. Secondly, the sampling density dominates the ratio between bit-rate and reconstruction quality, e.g., we observe a bit rate of about 0.4 bpp for the David model at a reconstruction fidelity of 87 dB, while at the same time, Igea takes a significantly higher rate of about 3 bpp at a lower reconstruction quality of 84.4 dB. Hence, our method aggressively removes redundancies when the model has a very dense sampling relative to the surface complexity, e.g., as it is the case for David.

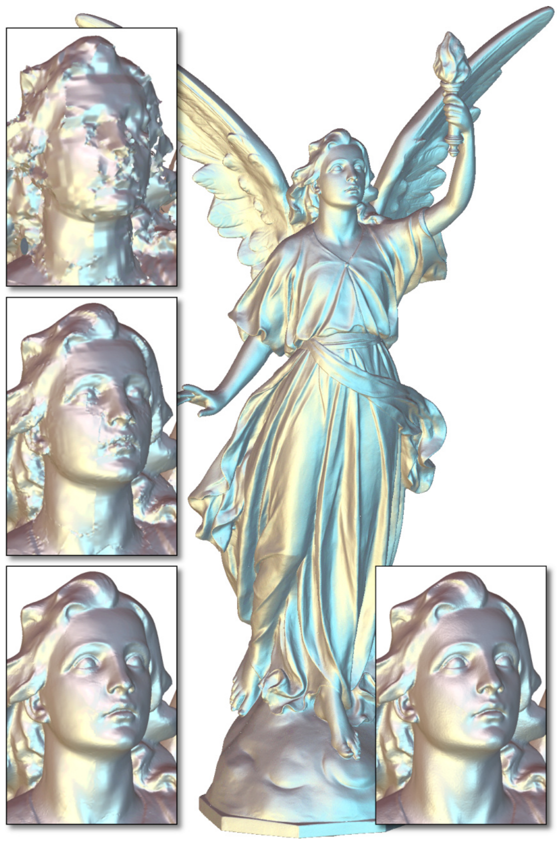
Figure 10: *Point-based compression of Lucy; global view: reconstruction at* 0.47 *bpp, which corresponds to a file size of* 804 *kByte; the close-up views on the left show reconstructions at* 0.09 *bpp* ($E_{rms} = 2.89$), 0.15 *bpp* ($E_{rms} = 0.62$), *and* 0.47 *bpp* ($E_{rms} = 0.14$) *(from top to bottom); the head on the bottom right corresponds to the original model.*

Figure 10 shows the compressed Lucy statue at 0.47 bpp at the global view. The close-up views show reconstructions at 0.09 bpp, 0.15 bpp, and again 0.47 bpp. A close-up view of the original is shown at the bottom right. For Lucy we obtain visually pleasant results already at very low bit-rates, e.g., around a half bit per point (compare the two close-up views at the bottom). This indicates that the original shows a high point density compared to the geometric complexity and our coder removes redundancies adequately.

Figure 11 shows the compression performance for the Stanford Dragon point model. The top row shows the partition with 150 patches (left) and rate distortion curves using the error measure Equation (2). The two curves are obtained using the split-merge partitioning and the GLA clustering. The right curves compare our results to previous coders [FCOAS03] and [WGE*04], using the error measure

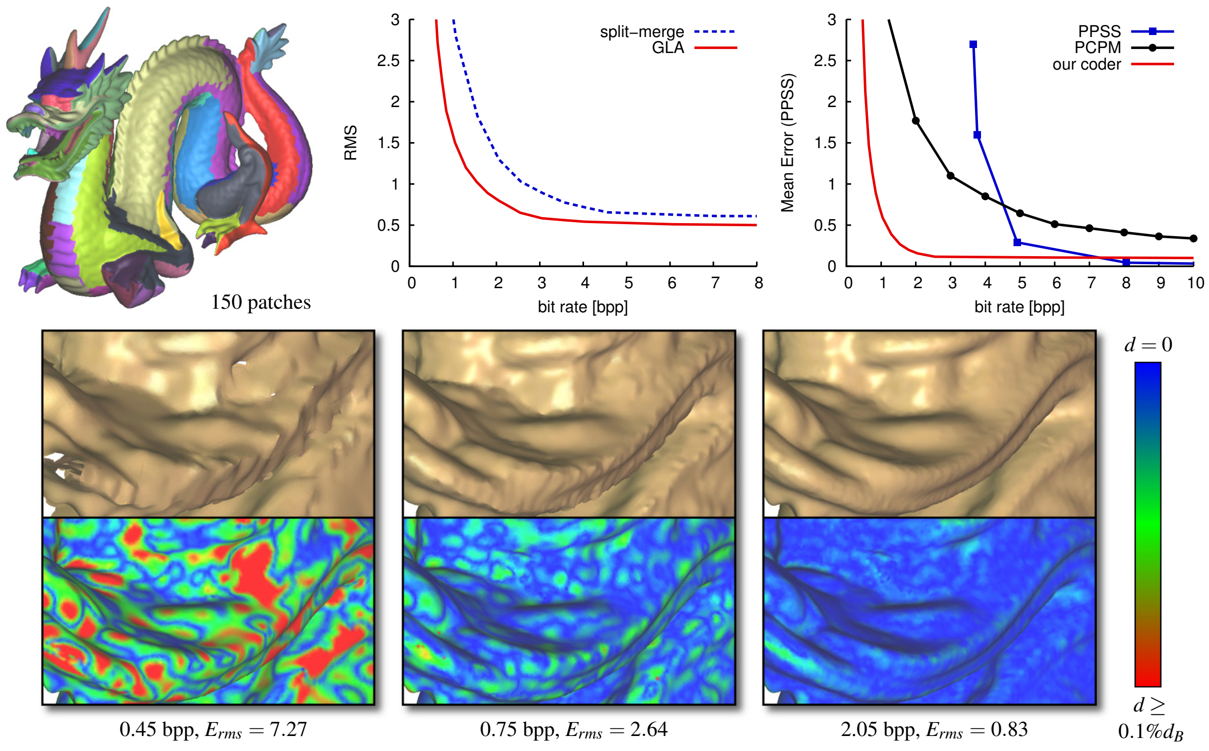

Figure 11: *Point-based compression of Dragon; top row: partition, performance curves for split-merge and GLA partitioning, and comparison of our method to [FCOAS03] and [WGE*04]; bottom row: decoded models at various bit-rates with decoding errors $d(\mathcal{S},\hat{\mathcal{S}})$, and visualizations of the distances of the original surface to the MLS surface of the reconstructed point set.*

in [FCOAS03]. A significant performance improvement can be observed. The bottom row in Figure 11 shows close-up views of the decoded Dragon at various bit-rates. The images below visualize the distance of the original to the reconstructed surface, where blue corresponds to a distance of zero and red correspond to distances greater or equal to 0.1 percent of the bounding box diagonal of the original model.

Figure 12 shows compression of David (3,614,098 points). The rate distortion curves indicate an excellent approximation quality at already 0.5 bits per point, which confirms that our method adequately removes redundancies for large models as for Lucy in Figure 10. Artifacts become visible in the global view only when reducing the bit-rate drastically, e.g., down to 0.11 bpp.

The close-up views in Figure 12 show that the rendering of the reconstruction is more blurry than that of the original. This is explained by the fact that our method degrades the sampling of the original model (which may be adaptive to surface features). This drawback can be overcome by considering the sampling density during the patch layout construction, e.g., the patches are forced to show similar sampling densities. Although, an improvement in visual quality would be expected, the total number of patches will increase, which may worsen the rate-distortion performance due to increased costs for side information and binary images.

5.3. Mesh Geometry Compression

In the following, we present results for mesh models. To evaluate the surface error, Equation (2), we adopt the METRO tool [CRS98]. The rates are given in bits per vertex (bpv).

Figure 13 shows results for the Shakyamuni mesh. The close-up views show the partition and reconstructions at various bit-rates. We observe that the reconstruction at 1 bpp is already close to the original model ($E_{rms} = 0.3 \cdot 10^{-4} d_B$) and provides good visual quality. The corresponding L_∞ error is $E_\infty = \max(\sup_{p \in \mathcal{S}} d(p,\hat{\mathcal{S}}), \sup_{p \in \hat{\mathcal{S}}} d(p,\mathcal{S})) = 4 \cdot 10^{-4} d_B$, which is still within a tolerance of, e.g., 0.5% of the bounding box diagonal length. The top right curve shows the rate-distortion performance, while the bottom curve shows the reconstruction error at 1 bpp for a varying number of patches. We found that the error is proportional to the number of patches down to a certain optimal number. This behavior is explained by the fact that a large number of patches leads to increased costs for side information, e.g., base plane parameters and binary images, which in turn reduces the bit budget for the wavelet coefficients and leads to larger errors. Further reduction of the patch number leads to elevation map artifacts, which worsens the distortion. We found that this behavior is typical. In general, there is an optimal number of patches for which the rate-distortion curve is the lowest.

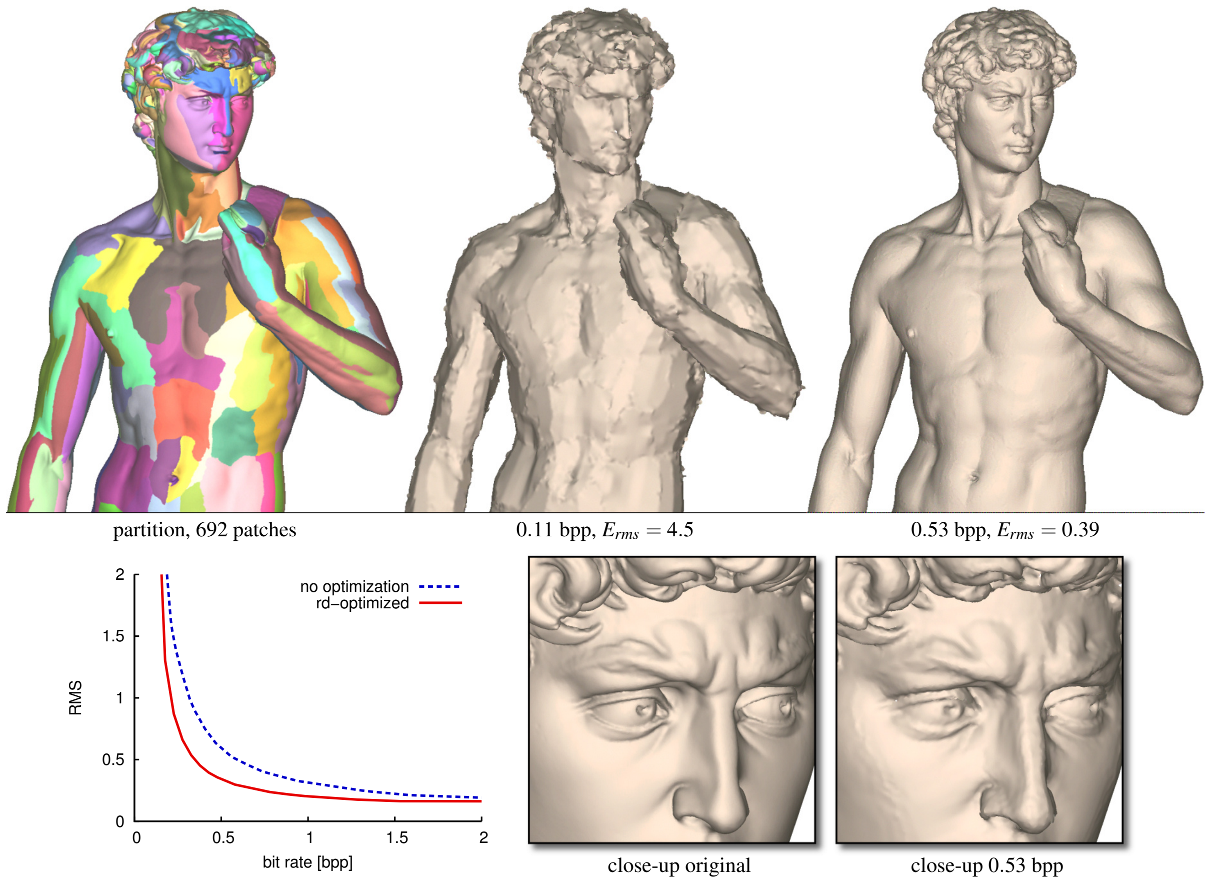

Figure 12: *Point-based compression of David; top: partition and two reconstructions; bottom: rate-distortion graphs without and with applying the optimal bit allocation scheme and close-up views of the original and a reconstruction.*

Figures 14 and 15 show compression results for the commonly used meshes Rabbit and Venus. The curves compare the performance of our coder to the state-of-the-art methods [KSS00] and [KG02]. Basically, the method in [KG02] provides a better rate-distortion performance at very low bit-rates. Nevertheless, our method provides results that are similar or slightly better than in [KG02] at bit-rates of 1 bpp and above. At these rates, the errors are sufficiently small, meaning that the reconstructions are close to the original model and are suitable for a practical application.

Table 4 lists all tested meshes and gives detailed coding results, including the amount of side bits r_0 and bits for the binary images, compare Table 3.

Comparison to Multi-Chart Geometry Images. Our method is similar to the approach proposed by Sander *et al.* [SWG*03], however, their parameterization is more complex than ours, since the detail coefficients are three-dimensional. Their work does not focus on compression, hence, we compare our method to theirs in terms of parameterization quality. To make an appropriate comparison, we reconstruct our model with three times as many vertices as in [SWG*03], since our detail coefficients are only one-dimensional. We reconstruct the Dragon model at 158,009 vertices and a reconstruction quality of 80.6 dB, which is comparable to the result in [SWG*03], where the Dragon model is reconstructed at 79.4 dB (with 52,059 vertices, leading to 156,177 detail coefficients).

Computational Costs. Table 5 shows timings for the individual steps in the pipeline for various models. For point models, we observe high computational costs for the resampling procedure, which is due to the fact that a number of MLS projections have to be applied to construct the elevation value for each sample point. The resampling is fast for meshes, since the elevation values can be read off directly from the original mesh. Moreover, there is a significant amount of run time needed for the rate-distortion optimization, since it includes many encoding and decoding steps to sample the rate-distortion curve for each patch.

The last column of Table 3 and Table 4 show timings that are required for decoding and reconstruction. For meshes, the gap closing has to be applied after decoding, which results in slower reconstructions than for points.

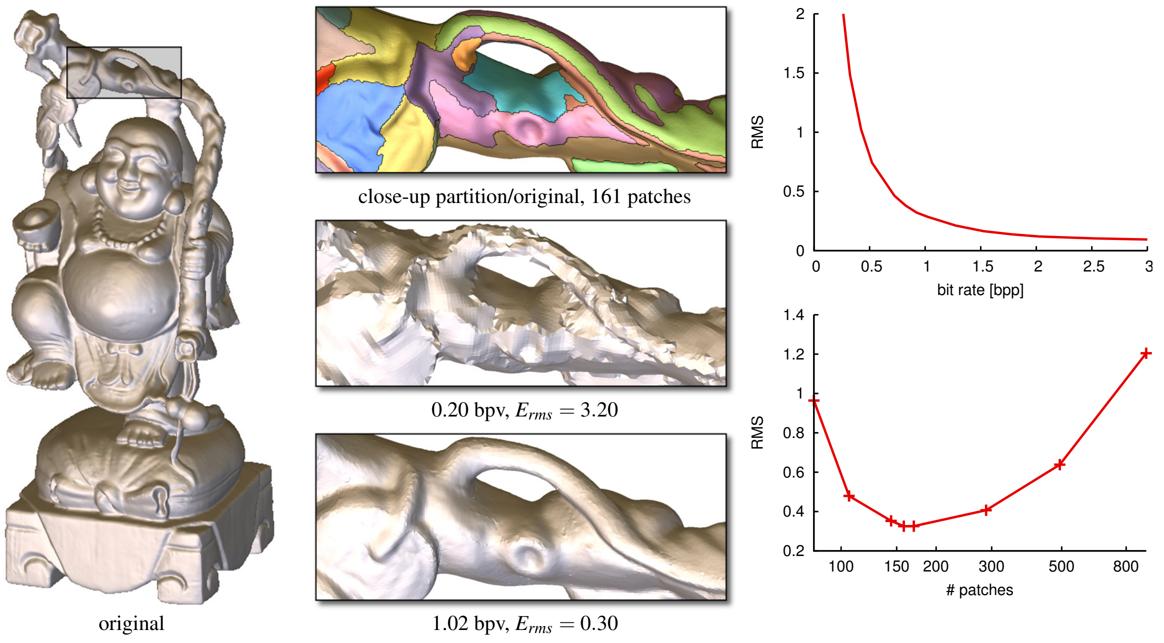

Figure 13: *Mesh compression of Shakyamuni, original model (left); the close-up views show the partition and reconstructions at* 0.2 *bpp and* 1 *bpp (middle); reconstruction errors for various bit-rates (top right); reconstruction errors for varying number of patches at 1.0 bpv show that there is an optimal number of patches for each model (lower right).*

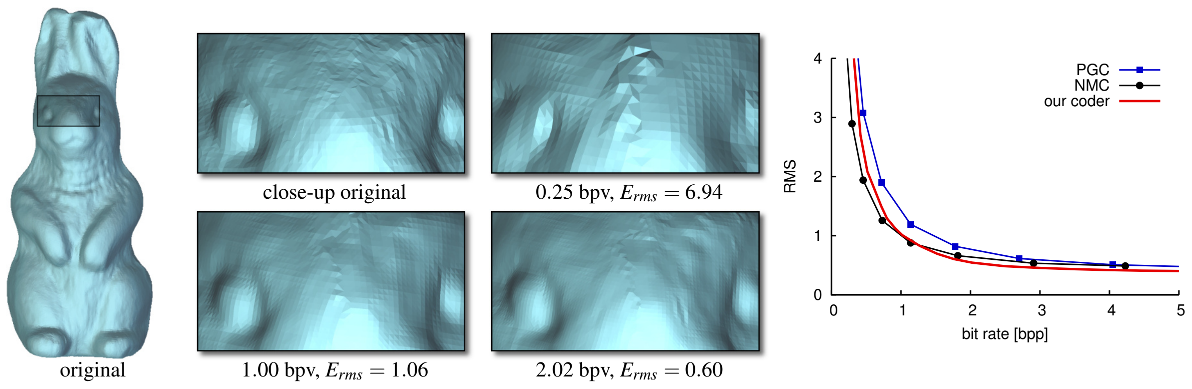

Figure 14: *Mesh compression as in Figure 13 for Rabbit (left); the close-up views show that boundary artifacts become visible at very low bit-rates (middle); the compression performance compared to the coders PGC [KSS00] and NMC [KG02] (right).*

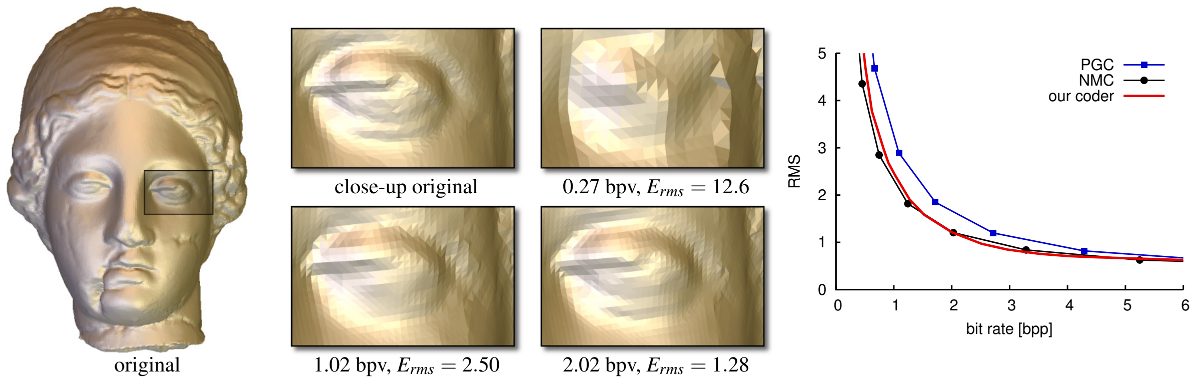

Figure 15: *Mesh compression as in Figure 14 for Venus with* 50,002 *vertices.*

reconstructed vertices	mask bits [%]	total rate [bpv]	E_{rms} $[10^{-4}d_B]$	PSNR [dB]	T_d [sec]
Venus, 50,002 vertices, 8 patches, $r_0 = 1{,}353$					
18,392	28.9	0.32	9.06	60.9	1.3s
51,519	12.8	1.27	1.91	74.4	2.7s
132,437	8.8	2.96	0.85	81.4	3.2s
Rabbit, 67,039 vertices, 6 patches, $r_0 = 1{,}083$					
24,587	18.2	0.42	2.70	71.3	1.5s
68,572	12.3	1.02	1.01	79.9	2.8s
134,832	7.3	2.49	0.46	86.8	3.4s
Balljoint, 137,062 vertices, 11 patches, $r_0 = 1{,}758$					
50,344	15.4	0.51	2.01	73.9	2.6s
140,619	13.4	1.01	1.01	79.9	3.8s
202,110	8.2	2.01	0.55	85.2	5.2s
Dragon, 437,645 vertices, 162 patches, $r_0 = 22{,}143$					
175,003	31.1	0.35	3.86	68.3	8s
288,677	17.6	0.95	0.91	80.8	12s
452,047	10.7	2.05	0.46	86.8	15s
Shakyamuni, 996,956 vertices, 161 patches, $r_0 = 22{,}008$					
308,527	34.4	0.22	2.35	72.6	10s
369,113	20.9	0.52	0.74	82.6	14s
1,481,024	14.4	1.27	0.26	91.7	28s
XYZRGB Dragon, 3.6M vertices, 534 patches, $r_0 = 72{,}363$					
1,370,951	43.1	0.22	1.56	76.1	57
3,295,725	27.5	0.52	0.49	86.2	82
7,304,838	9.2	2.52	0.11	99.2	120

Table 4: *Compression results for meshes, compare Table 3.*

Boundary Artifacts. The weakness of our framework is the occurrence of discontinuities near the patch boundaries. These become visible especially at very low bit rates, where the reconstruction error becomes large, Figure 16(a). Although the patch discontinuities do not worsen the compression performance with respect to the geometric error, they may lead to unpleasant visual results. To fix this problem, we propose to apply a smoothing filter to blend the patch boundaries in a post-processing step after decoding. Specifically, we identify all points on the patch boundaries and set their positions to the respective Gaussian smoothed version, which is obtained by computing the weighted average of point positions in a local neighborhood around each boundary point. Results of this procedure are shown in Figure 16. We observe that the discontinuities are widely removed while the geometric approximation quality is preserved.

6. Conclusions

We presented a framework for 3D surface compression. Our coder applies an elementary parameterization to decompose the surface into a number of regularly sampled height fields, which are encoded using state-of-the-art shape-adaptive image wavelet coding, thus, mapping the problem of 3D geometry coding to 2D image coding.

Our method is applicable to mesh- and point-based geometry, since our pipeline relies only on an appropriate surface representation of the 3D model. We obtained an improved rate-distortion performance in comparison with state-of-the-art point-based surface compression. If the reconstructed model is a mesh, our coder still achieves results that are comparable to those of the best mesh compression algorithms, e.g., normal mesh compression. However, in contrast to this method, our framework is based on an elementary parameterization, leading to simple implementations.

model	T_{clu}	T_{par}	$T_{res}^{(p)}$ / $T_{res}^{(m)}$	T_{opt}	T_{enc}
Venus	0.6s	0.03s	5.6s / 1.1s	14s	0.1s
Rabbit	1.1s	0.08s	12s / 1.5s	26s	0.1s
Balljoint	2.6s	0.1s	34s / 2.4s	43s	0.3s
Dragon	12s	0.3s	2.7m / 10s	2.2m	0.9s
Shakyamuni	34s	0.6s	6.2m / 23s	8.6m	1.6s

Table 5: *Timings for the following steps in the pipeline: GLA clustering (T_{clu}), parameterization (T_{par}), resampling ($T_{res}^{(p)}$ for the point-based model and $T_{res}^{(m)}$ for the mesh), rate-distortion optimization (T_{opt}), and encoding (T_{enc}).*

The limitation of our method is the occurrence of patch artifacts, which is common with region-based coding schemes, such as JPEG for images. Moreover, the resampling procedure widely destroys the (perhaps adaptive) sampling of the original model. Finally, our approach is not suitable for encoding diffuse and irregular objects, e.g., plant models. It rather aims at densely sampled surface-like 3D objects, i.e. models that were acquired by a laser range scanner.

In future work we may compare our method to previous coders with respect to error measures that model human visual perception more appropriately than the L_2 metric does. We may also embed our system in a out-of-core framework, in that we apply out-of-core partitioning as in [IG03], and perform the resampling step and the RD-optimization on each individual patch (in-core).

Acknowledgments

We thank the reviewers for their helpful comments to improve the paper. This work was supported by the DFG Graduiertenkolleg 1042 'Explorative Analysis and Visualization of Large Information Spaces'.

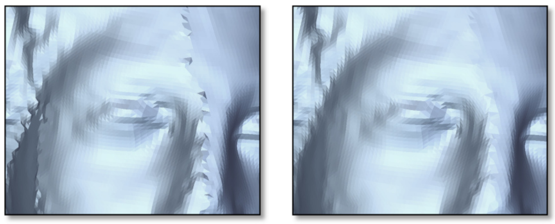

Figure 16: *Compressing models at very low bit-rates leads to visible artifacts near the patch boundaries, $E_{rms} = 3.91$ (left); applying a smoothing filter in these areas improves the visual quality, $E_{rms} = 3.86$ (right).*

References

[AA03] Adamson A., Alexa M.: Ray tracing point set surfaces. In *Proc. Shape Modeling International* (2003).

[ABCO*03] Alexa M., Behr J., Cohen-Or D., Fleishman S., Levin D., Silva C.: Computing and rendering point set surfaces. *IEEE Trans. on Computer Graphics and Visualization 9*, 1 (2003), 3–15.

[AG04] Alliez P., Gotsman C.: Recent advances in compression of 3D meshes. In *Advances in Multiresolution for Geometric Modelling*. Springer, 2004.

[BNK02] Borodin P., Novotni M., Klein R.: Progressive gap closing for mesh repairing. In *Proc. Computer Graphics International* (2002), pp. 201–213.

[BWK02] Botsch M., Wiratanaya A., Kobbelt L.: Efficient high quality rendering of point sampled geometry. In *Proc. Workshop on Rendering* (2002), pp. 53–64.

[CM02] Chou P., Meng T.: Vertex data compression through vector quantization. *IEEE Trans. on Visualization and Computer Graphics 8*, 4 (2002), 373–382.

[CRS98] Cignoni P., Rocchini C., Scopigno R.: Metro: measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2 (1998), 167–174.

[CSAD04] Cohen-Steiner D., Alliez P., Desbrun M.: Variational shape approximation. *ACM TOG 23*, 3 (2004).

[Cyb] http://www.cyberware.com/samples/.

[Dee95] Deering M.: Geometry compression. In *Proc. ACM SIGGRAPH* (1995), pp. 13–20.

[Eve63] Everett H.: Generalized lagrange multiplier method for solving problems of optimum allocation of resources. *Operations Research 11* (1963), 399–417.

[FCOAS03] Fleishman S., Cohen-Or D., Alexa M., Silva C. T.: Progressive point set surfaces. *ACM TOG 22*, 4 (2003).

[FH05] Floater M. S., Hormann K.: Surface parameterization: A tutorial and survey. In *Advances in Multiresolution for Geometric Modelling*. Springer, 2005, pp. 157–186.

[Fow00] Fowler J. E.: QccPack: an open-source software library for quantization, compression and coding. In *Applications of Digital Image Processing XXIII, Proceedings SPIE 4115* (2000), pp. 294–301.

[Fow04] Fowler J. E.: Shape-adaptive coding using binary set splitting with k-d trees. In *Proc. ICIP* (2004), vol. 2.

[Gär99] Gärnter B.: Fast and robust smallest enclosing balls. In *Proc. Symposium on Algorithms* (1999), pp. 325–338.

[GG92] Gersho A., Gray R. M.: *Vector quantization and signal compression*. Kluwer Academic Publishers, 1992.

[HDD*93] Hoppe H., DeRose T., Duchamp T., McDonald J., Stützle W.: Mesh optimization. In *Proc. ACM SIGGRAPH* (1993), pp. 19–26.

[IG03] Isenburg M., Gumhold S.: Out-of-core compression for gigantic polygon meshes. *ACM TOG 22*, 3 (2003), 935–942.

[IK01] Ivanov D. V., Kuzmin Y.: Spatial patches - a primitive for 3D model representation. *Computer Graphics Forum 20*, 3 (2001), 511–521.

[KG00] Karni Z., Gotsman C.: Spectral compression of mesh geometry. In *Proc. ACM SIGGRAPH* (2000), pp. 279–286.

[KG02] Khodakovsky A., Guskov I.: Compression of normal meshes. In *Geometric Modeling for Scientific Visualization*. Springer Verlag, Heidelberg, Germany, 2002.

[KN] http://www.inf.uni-konstanz.de/cgip/projects/surfac/.

[KPRW05] Kälberer F., Polthier K., Reitebuch U., Wardetzky M.: FreeLence - coding with free valences. *Computer Graphics Forum 24*, 3 (2005), 469–478.

[KSS00] Khodakovsky A., Schröder P., Sweldens W.: Progressive geometry compression. In *Proc. ACM SIGGRAPH* (2000), pp. 271–278.

[KSW05] Krüger J., Schneider J., Westermann R.: DUODECIM - a structure for point scan compression and rendering. In *Proc. Point-Based Graphics* (2005).

[LCB03] Lavu S., Choi H., Baraniuk R.: Geometry compression of normal meshes using rate-distortion algorithms. In *Proc. Symposium on Geometry Processing* (2003), pp. 52–61.

[LL00] Li S., Li W.: Shape-adaptive discrete wavelet transforms for arbitrary shaped visual object coding. *IEEE Trans. on Circuits and Systems for Video Technology 10*, 5 (2000), 725–743.

[Llo82] Lloyd S. P.: Least squares quantization in PCM. *IEEE Trans. on Information Theory 28* (1982), 129–137.

[LMH00] Lee A., Moreton H., Hoppe H.: Displaced subdivision surfaces. In *Proc. ACM SIGGRAPH* (2000), pp. 85–94.

[LPC*00] Levoy M., Pulli K., Curless B., Rusinkiewicz S., Koller D., Pereira L., Ginzton M., Anderson S., Davis J., Ginsberg J., Shade J., Fulk D.: The Digital Michelangelo Project: 3D scanning of large statues. In *Proc. ACM SIGGRAPH* (2000), pp. 131–144.

[LR81] Langdon G. G., Rissanen J.: Compression of black-white images with arithmetic coding. *IEEE Trans. Communications 29*, 6 (1981), 858–867.

[OH06] Ochotta T., Hiller S.: Hardware rendering of 3D geometry with elevation maps. In *Proc. Shape Modeling International* (2006), pp. 45–56.

[OS04] Ochotta T., Saupe D.: Compression of point-based 3D models by shape-adaptive wavelet coding of multi-height fields. In *Proc. Point-Based Graphics* (2004), pp. 103–112.

[PG01] Pauly M., Gross M.: Spectral processing of point-sampled geometry. In *Proc. ACM SIGGRAPH* (2001).

[PK05] Peng J., Kuo C.-C. J.: Geometry-guided progressive lossless 3D mesh coding with octree (ot) decomposition. *ACM TOG 24*, 3 (2005), 609–616.

[PZvBG00] Pfister H., Zwicker M., van Baar J., Gross M.: Surfels: surface elements as rendering primitives. In *Proc. ACM SIGGRAPH* (2000), pp. 335–342.

[RL00] Rusinkiewicz S., Levoy M.: QSplat: a multiresolution point rendering system for large meshes. In *Proc. ACM SIGGRAPH* (2000), pp. 343–352.

[Ros99] Rossignac J.: Edgebreaker: connectivity compression for triangle meshes. *IEEE Trans. on Visualization and Computer Graphics 5*, 4 (1999), 47–61.

[SCOT03] Sorkine O., Cohen-Or D., Toledo S.: High-pass quantization for mesh encoding. In *Proc. Symposium on Geometry Processing* (2003), pp. 42–51.

[SP96] Said A., Pearlman W. A.: An image multiresolution representation for lossless and lossy image compression. *IEEE Trans. on Image Processing 5*, 9 (1996), 1303–1310.

[SRK02] Szymczak A., Rossignac J., King D.: Piecewise regular meshes: construction and compression. *Graphical Models 64* (2002), 183–198.

[Sta] http://graphics.stanford.edu/data/3Dscanrep/.

[SWG*03] Sander P., Wood Z., Gortler S., Snyder J., Hoppe H.: Multi-chart geometry images. In *Proc. Symposium on Geometry Processing* (2003), pp. 146–154.

[TG98] Touma C., Gotsman C.: Triangle mesh compression. In *Proc. Graphics Interface* (1998), pp. 26–34.

[TR98] Taubin G., Rossignac J.: Geometric compression through topological surgery. *ACM TOG 17*, 2 (1998), 84–115.

[WGE*04] Waschbüsch M., Gross M., Eberhard F., Lamboray E., Würmlin S.: Progressive compression of point-sampled models. In *Proc. Point-Based Graphics* (2004).