Multiresolution Surface and Volume Representations

Oliver G. Staadt

Computer Science Department
University of California, Davis
One Shields Avenue, Davis, CA 95616, USA
staadt@cs.ucdavis.edu

Summary. We present a wavelet-based geometry compression pipeline in the context of hierarchical surface and volume representations. Due to the increasing complexity of geometric models used in a vast number of different application fields, new methods have to be devised that enable one to store, transmit and manipulate large amounts of data. Based on a multiresolution wavelet representation, we have developed a complete compression pipeline suitable for geometric data on uniform grids in two and three dimensions. Local and global oracles in wavelet space are employed to control the approximation error in lossy compression settings. Two geometry simplification schemes, which are able to build hierarchical mesh representations, are an essential part of the pipeline. In contrast to the two wavelet-based approximation schemes, we have devised the progressive tetrahedralization method, an extension of the popular progressive meshes into volumetric settings. We compare the three approximation schemes with each other using several two- and three-dimensional data sets and provide an extensive error and performance analysis. These results emphasize the individual strengths and weaknesses of each of the discussed methods and concepts.

1 Introduction

During the past decade, the complexity of geometric models in various application fields has increased steadily. With the wide availability of powerful computer systems, many conventional design and production processes are either simulated or carried out entirely on computers. Good examples include the wide field of product design in almost any engineering discipline. The design cycle for many consumer products including cars, toys, appliances, and so on, typically incorporates highly detailed geometric models. Moreover, the simulation of natural phenomena requires the creation of very large and complex geometric models for further computational processing and scientific visualization.

Even though the performance of state-of-the-art computer systems is still increasing in terms of computational power and memory availability, the complexity of geometric models, driven by the demand of applications, is at least increasing at the same rate. Several solutions have been proposed to attack these problems, which can more or less be classified into the following overlapping areas: Geometry simplification, geometry compression and hierarchical mesh representations.

Non-hierarchical geometry simplification schemes have successfully been employed for many years to reduce the number of elements in geometric models. The

use of simplified models can speed-up visualization and further numerical processing. Those method allow for the generation of high-quality approximations at a single level of resolution. If the user requests a mesh at a different resolution, however, the whole simplification process has to be repeated from the start.

A completely different approach is to optimize the memory requirements for storing and transmitting large models. Geometry compression methods reduce the memory footprint of a model without compromising its geometrical or topological properties. We distinguish between lossy and lossless compression methods. A certain loss of precision can usually be tolerated for compressing positional information of individual vertices constituting the model. Connectivity information, which specifies the relation between these vertices and defines the model's topology, should be compressed without loss of information.

Although non-hierarchical simplification methods and geometry compression schemes often allow for progressive reconstruction and transmission over networks, they do not inherently define a hierarchical representation of the model. For very large data sets, however, it might be advantageous to inspect or even edit the data at different levels, and hierarchical representations provide build-in support for such applications. Note the difference between "linear" progressive schemes, which also enable the reconstruction of different levels, and fully hierarchical methods: The latter contain additional information about dependencies between successive levels, thus allowing to switch consistently between them. This is especially desirable in multiresolution editing applications, where changes applied at one approximation level are automatically propagated throughout the hierarchy.

The main focus of this paper is a survey and comparison of the aforementioned surface and volume representations, highlighting their advantages and shortcomings.

2 Multiresolution Approximations

In this section we discuss the mathematical fundamentals of the preprocessing we employ for data preconditioning. B-spline wavelets are used as a precoding transform since they combine various advantageous features, such as vanishing moments, continuous approximation, bounded interval definition, linear time algorithms, and localization. For reasons of readability, we first review some basics of cardinal B-spline wavelets. However, our attention is mostly directed to the definition of global oracles, that is, schemes to reject unimportant coefficients. Our global oracle consists of a greedy algorithm resulting from an elaborate analysis of L^2-errors in semiorthogonal settings [10]. Additionally, we will demonstrate how local oracles reject coefficients in unimportant spatial regions and thus enable the construction of *electronic magnifying glasses* for interactive data inspection. For reasons of simplicity, we perform all computations for 1-D functions, but the results extend straightforwardly to higher dimensions.

2.1 Cardinal B–Spline Wavelets

B–spline wavelets were initially introduced by Chui [3], and were extended to bounded intervals by [8] and [22], while nonuniform knot sequences were addressed for instance by [2]. Due to a rich variety of literature in this area, we restrict our introduction to those topics essential for an understanding of our framework.

B–spline wavelets can be constructed from a multiresolution hierarchy of cardinal B–spline scaling functions. Semiorthogonality invokes an additional degree of freedom, however. Thus, approaches as in [8] or [22] end up in slightly different construction schemes. We adapted the methods of Quak et. al [8] to construct B–spline wavelets of arbitrary order bounded to the interval.

Assuming the reader is familiar with some fundamentals of discrete wavelet transforms (DWT), the implementation of the forward transform is carried out by sequences of projection operators $\mathbf{A}^m$, $\mathbf{B}^m$, where $m = 1 \ldots M$ stands for the decomposition level. An initial function $f(x)$ is mapped from the higher resolution approximation space V^m onto a lower resolution space V^{m+1} and onto its orthogonal complement space W^{m+1}. Given the coefficient vectors $\mathbf{c}^m$ and $\mathbf{d}^m$ for the scaling functions $\varphi_i^m(x)$, and wavelets $\psi_i^m(x)$ in the 1-D setting, with

$$c_i^m = \langle f, \varphi_i^m \rangle \qquad d_i^m = \langle f, \psi_i^m \rangle \,, \tag{1}$$

($i : 1 \ldots N/2^m + \text{order} - 1$, order: B–spline order,) the decomposition is carried out by matrix operations

$$\mathbf{c}^{m+1} = \mathbf{A}^m \mathbf{c}^m \qquad \mathbf{d}^{m+1} = \mathbf{B}^m \mathbf{c}^m. \tag{2}$$

Due to the semiorthogonality, we require the inverse operators $\mathbf{P}^m$ and $\mathbf{Q}^m$ to compute the reconstruction with

$$\mathbf{c}^m = \mathbf{P}^m \mathbf{c}^{m+1} + \mathbf{Q}^m \mathbf{d}^{m+1}. \tag{3}$$

It can be easily proven [8] that the operators relate to each other by

$$\left[\frac{\mathbf{A}^m}{\mathbf{B}^m}\right] = \left[\mathbf{P}^m | \mathbf{Q}^m\right]^{-1}. \tag{4}$$

In the case of cardinal B–spline wavelets, sparse operators $\mathbf{P}^m$ and $\mathbf{Q}^m$ come along with dense matrices $\mathbf{A}^m$ and $\mathbf{B}^m$. In order to construct linear time algorithms for both decomposition and reconstruction, it is sufficient to know the sequences $\mathbf{P}^m$ and $\mathbf{Q}^m$ to perform an additional base transform of the coefficients into their duals $\tilde{\mathbf{c}}^m$ and $\tilde{\mathbf{d}}^m$ using the inner product matrices $\mathbf{\Phi}^m$ and $\mathbf{\Psi}^m$.

Note that the decomposition involves solutions of the sparse linear systems of type $\mathbf{\Psi}^m \cdot \mathbf{d}^m = \tilde{\mathbf{d}}^m$ for each iteration and $\mathbf{\Phi}^M \cdot \mathbf{c}^M = \tilde{\mathbf{c}}^M$ for the last iteration step. Fortunately, this can be accomplished in linear time as well. For brevity we abandon all mathematical details associated with the construction of these transforms and refer the reader to [8].

2.2 Oracles

One of the most important applications of wavelets is data compression. Instead of using all wavelets ψ_{mp} for reconstruction, it is possible to truncate the approximation. An oracle is used to determine the set of wavelets that are omitted in the truncated approximation.

Global Oracles. A global oracle rejects unimportant wavelet coefficients from the approximation while minimizing a given error norm.

First, let us recall finite dimensional orthogonal approximations for $L^2(\mathbb{R})$. The basis $\{\varphi_i(x)\}_{i=1\ldots N}$ expands a function $f(x)$ as

$$f(x) = \sum_{i=1}^{N} c_i \varphi_i(x), \tag{5}$$

with coefficients $c_i \in V_{m_0}$. Given an integer $1 \leq K < N$, we try to find the best approximation of f with the truncated approximation f'. The L^2 error of $f'(x)$ is determined by the linear combination of the rejected basis functions

$$\begin{aligned} \|f(x) - f'(x)\|_{L^2}^2 &= \left\| \sum_{i=K+1}^{N} c_i \varphi_i(x) \right\|_{L^2}^2 \\ &= \left\langle \sum_{i=K+1}^{N} c_i \varphi_i(x), \sum_{i=K+1}^{N} c_i \varphi_i(x) \right\rangle \\ &= \sum_{i=K+1}^{N} |c_i|^2 \end{aligned} \tag{6}$$

Note that the orthogonality of $\langle \varphi_i, \varphi_j \rangle = \delta_{ij}$ cancels out all intermediate terms in Eq. (6), which simplifies the relation. In other words, the magnitude of the coefficient c_i corresponds exactly to the fraction of energy provided by the corresponding basis φ_i. Hence, the L^2-energy of a function $f(x)$ is computed by

$$\|f(x)\|_{L^2}^2 = \langle f(x), f(x) \rangle = \int_{-\infty}^{\infty} |f(x)|^2 \, dx. \tag{7}$$

A globally optimal compression can thus be achieved by sorting the coefficients by their magnitude and by rejecting the K smallest ones. Obviously, this strategy yields an L^2-optimal oracle for orthogonal bases and can be computed in $O(N \log N)$ using a fast sorting algorithm. Unfortunately, in the semi-orthogonal case of B–spline wavelets, the intermediate terms in Eq. (6) are not canceled out, which complicates the construction of an oracle. Maximum distance norm oracles have been proposed by Stollnitz and others [22] for biorthogonal wavelets. Gross [10] and Staadt et al. [21] propose a global oracle for the semi-orthogonal setting. For more details, we refer the reader to these references.

Local Oracles. A local oracle allows one to control the approximation locally in interesting regions. Here, the spatial localization of the basis functions enables one to accentuate particular wavelets while suppressing the influence of others. In this understanding, a straightforward local oracle consists of a weighting function which operates on the coefficients of the transform. A first approach to this is given in Gross et. al [12] who employed a Gaussian weighting. The basic idea is to assume some ellipsoidal weighting area as a local region of interest in the spatial domain. Localization of the wavelet transform allows for the projection of scaled and translated versions of it into wavelet space, where individual coefficients are influenced. The initial version presented in [12], however, did not consider the support regions of individual basis functions, and can lead to some artifacts by rejecting wavelets ranging into the region of interest. Therefore, the method has been extended in [21] by computing the support of the basis functions and by using endpoint-interpolating B-wavelets.

3 Data Compression

We designed a compression pipeline by employing a wavelet-based sub-band codec. The forward compression proceeds as follows: The data set is normalized, transformed into wavelet space and both local and global approximation errors are controlled by the oracles introduced in Section 2.2. Sorting of the individual channels of the wavelet transform (WT) converts the multi-dimensional array into a 1-D data vector, which is quantized and encoded subsequently. Conversely, the decompression pipeline inverts the procedure and prepares the data for subsequent geometric reconstruction. For reasons of brevity, we will only present a short overview of the compression pipeline. For more in-depth coverage we refer to [21, 19].

The main steps of the compression pipeline comprise the following:

1. *Normalization:* As opposed to applications such as image compression, the order of magnitude for geometric data is usually not known in advance, but may vary between data sets and application areas. For that reason, the data are normalized, that is the original values c_{orig} are mapped to the interval $[0, \ldots, 1]$ using a simple mapping function $c_{norm} = \frac{c_{orig} - c_{max}}{c_{max} - c_{min}}$
2. *Wavelet Transform:* The subband codec used in this approach employs cardinal B–spline wavelets as introduced in Section 2.1. These basis functions are very well suited for the application of geometry compression, since they share some very important properties: (1) Endpoint-interpolation, (2) a high number of vanishing moments, (3) compact support, and (4) and efficient implementation.
3. *Oracle:* Local and global oracles, which have already been introduced in Section 2.2, are used to derive a truncated series of basis functions. In spite of the fact that semi-orthogonal B–spline wavelets are used for this codec, orthogonal oracles based on the L^2-norm are employed. They can only approximate semi-orthogonal L^2-oracles. Gross [10], however, presents a comparison between

orthogonal and semi-orthogonal L^2-oracles, which provides for a justification of this approximation.

4. *Linearization:* To prepare the data for bandwise progressive transmission, the multidimensional coefficient array is mapped into an 1-D vector. Here, the array is traversed from the most significant scaling function coefficients to the high frequency bands representing fine grained detail. This linearization step is carried out similar to the "zig-zag"-traversal of the DCT coefficients in the JPEG image compression standard [15].
5. *Quantization:* The quantization step comprises a multiplication of the initial floating point coefficients with a factor of 2^{n-1}, where n represents the number of bits to be assigned for each coefficient. Subsequent rounding operations transform the floating point value into signed integer formats of size n. Let c_{float} be a coefficient. Its quantized version c_{quant} is obtained by applying a scalar midtreat quantizer of the form $c_{quant} = \text{round}\left(2^{n-1} \cdot c_{float}\right)$
6. *Encoding:* An important task in the proposed compression pipeline is to convert the quantized integer vector into a bitstream of data. Therefore, an entropy coding scheme in the spirit of JPEG [15] is employed. Assuming that many of the coefficients will be equal to zero, encoding is carried out as follows: All non-zero coefficients are represented by two-tuples, where the first element represents the number of bits required to encode the second one. The second element contains the data value itself. All negative numbers are thus replaced by their absolute values, where in the case of a positive number the first bit is cleared. This enables one to encode of the sign elegantly.

4 Quadtree-Based Vertex Removal

The quadtree-based vertex removal scheme is a very fast and efficient method for further reducing the complexity of the data. This bottom-up strategy starts with a dense set of vertices and subsequently removes nodes according to a user-specified error threshold. This process eventually constructs a quadtree data structure which is triangulated by employing an efficient table look-up. The importance of a vertex is determined by analyzing detail coefficients at dyadic points reconstructed from wavelet space. For that reason, a single-step inverse wavelet transform has to be designed, which enables one to reconstruct the different sub-bands in wavelet space individually. Forward transform and compression are carried out in the same fashion as proposed in Section 3. A detailed description of the single-step inverse wavelet transform can be found in [12].

4.1 Vertex Removal in Regular Meshes

So far, some mathematical criteria have been elaborated for approximating a surface data set, sampled on a regular grid, using a multiresolution hierarchy. In order to build an adaptive surface triangulation, however, it is necessary to remove unimportant mesh vertices and to find a triangulation of the remaining ones. The basic

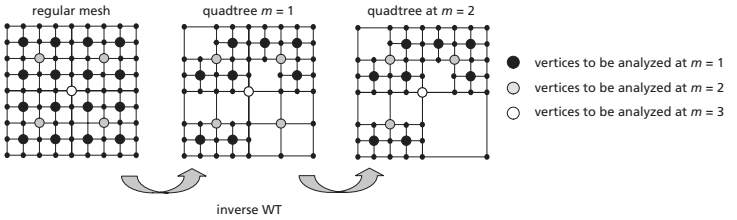

Fig. 1. Bottom-up construction of a quadtree from a regular mesh by analyzing the detail signals of the wavelet transform at each dyadic vertex.

criterion by which a mesh vertex is labeled as unimportant is given by the mathematical framework of the wavelet transform. In contrast to existing methods [14] which base on linear spline wavelets, this scheme can be generalized to any type of wavelet. Therefore, the criteria require more elaboration. Keeping in mind that any triangulation of the surface provides a planar approximation, the error between the original surface function $f(x, y)$ and the linear interpolant provided by a triangle has to be bound. Supposing furthermore that the initial data is expanded by wavelet bases, the detail signal in iteration m is used to decide on whether or not each $2^m + 1$th mesh vertex is necessary for the triangle approximation. First, every second vertex is visited and the value of the detail signal Δf^1 of iteration $m = 1$ is analyzed. If the detail signal in some neighborhood of vertex n is sufficiently small, then the vertex is considered not to be important and the approximation can be accomplished by a linear interpolation between vertex $n-1$ and $n+1$. This scheme can now be applied recursively by subsequent computation of the detail signals Δf^m, $m = 1, \ldots, M$ and by visiting all dyadic vertices at positions $n = 2^m k + 1$. Once the detail signal is sufficiently small and the adjacent vertices in step $m - 1$ are already removed, the current vertex is labeled as well.

As a consequence, this procedure results in recursively building a quadtree representation of the initial mesh by removing dyadic vertices. Figure 1 illustrates the thinning method which finally figures out a quadtree representation of the mesh vertices. The nodes of the quadtree contain either pointers to some child-nodes, or in case of leaves, point to the entries of a vertex list.

4.2 Vertex Removal Criteria

To finally decide on whether a mesh vertex can be removed or not, we propose the following three criteria, which additionally help to preserve the topology of the tree. A vertex will only be removed from the mesh if all criteria are met:

1. *Wavelet–criterion:* A vertex at iteration level m can be removed if the sum of the squares of its difference signal and those within a 4–neighborhood at resolution m are less than an upper bound ϵ.
2. *Resolution–criterion:* A vertex at iteration level m can be removed if the four surrounding vertices at level $m - 1$ have been removed in a previous step.

3. m *to* $m-2$ *criterion:* A vertex can be removed if the resulting cell is not adjacent to any cell of a level higher than $m-2$. Thus, growth is restricted to cell transitions from m to $m-2$, which simplifies the triangulation algorithm.

Once the tree is built from the above procedure, the quadtree cells have to be triangulated. A generic problem arising from meshing hierarchies of rectangular surface patches is the occurrence of cracks. A crack occurs if adjacency of quadtree cells at different depth and, hence, different resolution has not been considered. The surface may break up, holes may appear and any consistency required for normal interpolation gets lost. Non-manifold surfaces will be the result. We use a scheme based on a two-level look-up table to triangulate the quadtree cells, which has been introduced in [11, 12].

The quadtree-based vertex removal method introduced in this section has some limitations regarding progressivity, adaptivity, and easy extension into higher dimensions. The bottom-up approach requires successful transmission of all first-level detail coefficients before any vertices can be analyzed and eventually be removed. For a 2-D data set, the amount of data to be transmitted is equal to three quarters of the whole data. Furthermore, since only transitions between two consecutive levels are permitted, the adaptivity of the approximated mesh is limited. Extension into three dimensions is possible, but topological complexity and size of the look-up-tables would be increased substantially. In addition, the simplicity and elegance of the algorithm for two-dimensional data would not be maintained.

5 Delaunay-Based Vertex Insertion

5.1 Vertex Insertion in 1-D

In order to construct a vertex insertion strategy, we first consider the 1-D setting. Here, the problem reduces to finding a strategy for the reduction of line segments in piecewise linear approximations. Inspired by the algorithm of [7] we extended these ideas in [21] and modified the method to a recursive and progressive algorithm, illustrated in Figure 2. It starts by connecting the first point of a curve, $\mathbf{P}_0$, with the last point $\mathbf{P}_k$. All intermediate points representing the curve are compared against the line segment $\overline{\mathbf{P}_0\mathbf{P}_k}$ and the point with the largest distance, for instance $\mathbf{P}_2$, is identified. If its distance exceeds a predefined threshold ϵ_0, the vertex is considered *important* and labeled. We split the initial line segment in two halves, on each of which the algorithm can be applied recursively. Obviously, the quality of the removal can be controlled by the distance threshold. The advantage of this extension to the original method lies in the tree type refinement of the vertex analysis coming along with the recurrence relations.

The distance can be computed in different ways, where, however, the computation of the vertical distance, such as depicted in Figure 2c, is computationally much more expensive for general multidimensional settings. Therefore, we recommend computing the y–distance (see Figure 2a) to approximate nonparametric data.

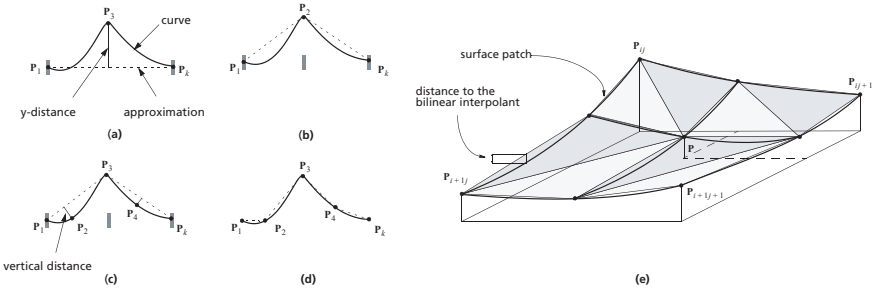

Fig. 2. Recursive algorithm assuming a smooth representation of the underlying curve: (a) $\mathbf{P}_2$ has largest vertical distance; (b) new approximation after insertion of $\mathbf{P}_2$; (c) example for vertical distance metric; (d) final result; (e) extension towards multiple dimensions exemplified for non-parametric data: 2-D version. The underlying B–spline patch is outlined in bold. A new vertex is inserted at position $\mathbf{P}$ and the distance is computed with respect to the bilinear interpolant of $\mathbf{P}_{ij}, \mathbf{P}_{ij+1}, \mathbf{P}_{i+1j}, \mathbf{P}_{i+1j+1}$.

5.2 Generalization to Higher Dimensions

Generalizations of the method towards multidimensional nonparametric data is straightforward. Starting from an initial grid, as in Figure 2e, the algorithm seeks the vertex $\mathbf{P}$ with the maximum distance and subdivides the field into four (in 2-D) or eight (in 3-D) subcells on which the method is applied recursively. In these cases the distances to the bilinear and trilinear interpolants of the cell vertices are computed, respectively.

Recalling the multiresolution B–spline approximation of the data motivates the extension of the algorithm towards a channelwise progressive vertex insertion. Therefore, the algorithm analyzes mesh vertices progressively and labels unimportant points as new data comes in. In 2-D, for instance, the basic idea is to start from an initial vertex field of resolution 2^{m-M} in each direction, where M represents the maximum iteration. The vertices are provided by the scaling function approximation $f^M(x, y)$ and are processed further by our algorithm. To define a distance metric, we assume a bilinear interpolant between the vertices which approximates the B–spline scaling function representation. If the difference signal $\Delta f^m(x, y)$ is received, the resolution is refined by two and all newly inserted vertices are checked conforming to our distance metric. If required, they will be inserted. In order to compute the intermediate vertices for each iteration, an inverse wavelet transform has to be applied to all coefficients at iteration level m as soon as they are received and decompressed.

For subsequent triangulation in 2-D and 3-D we employ Delaunay triangulation libraries such as [1, 18].

6 Progressive Tetrahedralizations

The wavelet-based vertex removal and insertion schemes introduced in Sections 4 and 5 are very well suited for efficient representations of various kinds of uniform data in multiple dimensions. An increasing demand for purely irregular and unstructured mesh representations, however, shows the requirement for the development of new methods providing even greater flexibility.

Progressive meshes [13] and its generalizations to higher dimensions [16] proofed to be an extremely powerful notion for the efficient representation of triangulated geometric objects at different levels-of-detail. Although a general formulation for arbitrary triangulations has already been given in [16], the special case of progressive tetrahedralizations (PT) is of enormous practical importance, since it can be used as a sophisticated representation for a large variety of computations [4, 23, 20].

This section presents a method for the hierarchical and progressive representation of tetrahedral meshes with arbitrary connectivity and we elaborate on some pitfalls and fallacies people might get caught in when trying to implement the method of edge collapsing for tetrahedral meshes. Specifically, we address the issue of defining appropriate cost functions. Unlike the elegant geometric approach presented in [13], we must account for volume and application specific properties, such as volume preservation, gradient estimation of the underlying data or aspect ratio of the simplex. In addition, we devised a sequence of tests to ensure a robust and consistent progressive tetrahedralization, which have first been introduced in [20]. Dey and colleagues [6] have independently presented an in-depth analysis of topology preserving edge collapses. For reasons of brevity, we refer the reader to [13, 16] for a general introduction to progressive meshes and its higher-dimensional variants.

In the tetrahedral setting, degeneration of tetrahedra into lower dimensional simplices is prohibited. The set of cells sharing an edge e_i will be called $\{icells_i\}$. Thus, an edge split adds the tetrahedra in $\{icells_i\}$ to the list of active elements. Conversely, the set of non-vanishing cells affected by the associated edge collapse is called $\{ncells_i\}$. In order to compute a sequence of robust, non-degenerate and consistent meshes, the following aspects have to be considered: (1) *Cost functions* which determine the order of edge collapse ($ecol$) operations depending on desired mesh optimization criteria. (2) *Feature edges* which should be preserved can be tested during preprocessing. (3) *Intersections* and *inversions* of tetrahedra inside and outside of $\{icells_i\} \cup \{ncells_i\}$ have to be processed at run time.

6.1 Cost Functions

Various elegant algorithms [9, 13] based on the edge-collapse/vertex-split paradigm use cost functions optimized for triangular surfaces, often accounting for distance measures, triangle shape, and others. In tetrahedral meshes, however, we have to redefine the terms of the cost function considering other features, like volume preservation or gradients. Trotts et al., e.g., describe a nice approach in [23]. Although many different measures are conceivable to control the simplification pro-

cess, the following ones yield a good balance between required degrees of freedom and the difficulty of parameter optimization. Thus, in our setting, for each edge $e_i = (\mathbf{v}_a, \mathbf{v}_b)$, the associated edge collapse operation $ecol_i(a,b) : M^i \leftarrow M^{i+1}$ is assigned the following cost:

$$\Delta E(e_i) = \Delta E_{grad}(e_i) + \Delta E_{vol}(e_i) + \Delta E_{equi}(e_i) + \Delta E_{normal}(e_i). \quad (8)$$

The first term $\Delta E_{grad}(e_i) = |s_a - s_b|$ forms a simplified measure for the difference of underlying scalar volume function along the edge e_i. Hence, edges with considerably differing scalar attributes are assigned high costs. This term of the cost function applies only to volumetric data sets with scalar attributes. The second term

$$\Delta E_{vol}(e_i) = \left(\sum_{T_j \in \{ncells_i\}} vol(T_j) - vol(\bar{T}_j) \right) + \sum_{T_j \in \{icells_j\}} vol(T_j)$$

penalizes volume changes and thus avoids that the mesh is shrinking. $\bar{T}_j$ denotes a tetrahedron after the collapse and $vol(T_j)$ its volume.

In many applications it is required that tetrahedra sustain equilateral shape. $\Delta E_{equi}(e_i)$ can be employed to balance the edge length of individual tetrahedra:

$$\Delta E_{equi}(e_i) = \sum_{T_j \in \{ncells_i\}} \left(\sum_{\{a,b\} \in T_j} (l_{a,b} - m_j)^2 - \sum_{\{a,b\} \in \bar{T}_j} (l_{a,b} - \bar{m}_j)^2 \right)$$

with $l_{a,b} = |\mathbf{v}_a - \mathbf{v}_b|$ and $m_j = 1/|T_j| \sum_{\{a,b\} \in T_j} l_{a,b}$.

Note that the initial mesh M^n will usually be generated from some triangulation scheme. Depending on the application context and the desired mesh features it can be advantageous to include $\Delta E_{edgelen}(e_i) = |\mathbf{v}_a - \mathbf{v}_b|$ into the cost function thereby enforcing short edges to be collapsed earlier.

For triangular surface meshes, the difference between the vertex normals of $\mathbf{v}_a$ and $\mathbf{v}_b$ are a good estimate of the local curvature of the mesh. If e_i lies within a region of high curvature, an edge collapse would flatten that region. Hence, this edge should be associated with an appropriate cost value defined as $\Delta E_{normal}(e_i) = (\mathbf{n}_a \cdot \mathbf{n}_b)|l_{a,b}|^2$, where $\mathbf{n}_i$ denotes the normal vector in vertex $\mathbf{v}_i$.

6.2 Consistency Tests

Unfortunately, brute force selection of edges according to the cost function from above can introduce mesh inconsistencies like degeneration, folding, intersection, or loss of individual features. It is possible, however, to avoid such inconsistencies by carrying out a combination of tests that can either be carried out during preprocessing or dynamically at run-time.

In a preprocessing step, all vertices and edges on the mesh boundary are marked. Table 1 lists the five possible combinations of boundary edge and vertices. Only

Table 1. The five possible combinations of boundary edges and vertices.

case	e	$\mathbf{v}_a$	$\mathbf{v}_b$	**valid**
1				yes
2		boundary		optional
3			boundary	optional
4		boundary	boundary	no
5	boundary	boundary	boundary	yes

cases one and five pass the consistency test for legal edge collapses. Allowing for edge collapses in cases two and three may result in "dents" on the mesh boundary, thus, they are optional. Interior edges (case one) have to be further checked dynamically for possible intersections and cell orientation changes.

A normal flipping heuristic [17] can be generalized to circumvent folding or self-intersection of cells. This can easily be implemented by testing for sign changes of surface normal vectors and by analyzing the volume of all tetrahedra $T_j \in \{ncells_i\}$ before and after the collapse. For more details see [19].

Starting from the observation that edge collapses can cause global intersections, we employ the method first introduced in [20] for dynamically testing for cell intersections. We assume that the input mesh is intersection-free and connected. We make no assumption about the genus of the mesh, though. The problem can then be divided into checking interior edge and mesh boundary edges.

After passing the static tests described above, we can ensure that any non-boundary edge e_i cannot have a boundary vertex as endpoint. If the set $\{icells_i\} \cup \{ncells_i\}$ contains no boundary edge, its boundary forms a polytope entirely wrapping the edge. A collapse of the edge, however, does not affect the boundary of the polytope, whose disjoint triangulation is given by the tetrahedra $\bar{T}_j \in \{ncells_i\}$. Thus, intersections can only occur with boundary cells and intersection tests can be restricted to the mesh boundary.

In the general case of boundary edges, we have to perform triangle–triangle intersection tests, which can be carried out as follows: First, we define the set of triangles $\{f_k\}$ containing all boundary faces of tetrahedra $T_j \in \{ncells_i\}$. These are the faces which can change after $ecol_i$ since they all share the new vertex $\bar{\mathbf{v}}_a$. Thus, they are our prime candidates for intersection with other boundary faces. To avoid testing these faces against all other boundary tetrahedra, we introduced a more efficient iterative method in [20], which will be used for the experiments in Section 7.

7 Experimental Results and Comparison

This section comprises two comparative experiments for the surface and volume approximation methods in the previous sections. We will analyze potential strengths and weaknesses of each of the methods. This comparison will focus on the geometric approximation performance of the three methods. Although the PT method can process a wide range of unstructured surface and volume data sets, we will restrict

these experiments to uniform input data, due to the constraints of the tensor product wavelet transform used by the other two methods.

7.1 Surface Approximation

In this section, we will investigate the approximation behavior of the quadtree-based vertex removal, the Delaunay-based vertex insertion and progressive meshes. We have selected a digital terrain data set of the Matterhorn, which has been cropped to 256 by 256 vertices. The original resolution of the data set is 701 by 481 vertices with 25 meter spacing. For the transformation into the progressive mesh representation, the following cost function terms have been used: Normal energy, edge length energy and equilateral energy, weighted with 50, 1 and 5, respectively. The total transformation took 2'58" on an SGI Octane R10000. The reconstruction parameters for all three methods have been adjusted in order to yield the same number of triangles for each of the five reconstruction levels. The geometric approximation errors have been calculated with Metro [5] and the error visualization is depicted in Figure 5.

We can see from the statistics and the error visualization that both, the progressive mesh and the Delaunay-based vertex insertion methods outperform the quadtree-based vertex removal. Due to the greater adaptivity of the first two methods, they achieve better approximation results than the latter method, which is constraint by the two-level look-up-table triangulation. A closer look at the quadtree-based method, however, reveals that its performance is still satisfactory. The relative error at the coarsest approximation level is below 0.15 percent.

7.2 Volume Approximation

For the final experiment, a medical CT volume data set of a child's skull has been employed. We have cropped the original data to a representative fragment with 32 by 32 by 32 voxels or 178,746 tetrahedra. Figure 4 depicts a semi-transparent view of this model.

This data set has been processed with both, the Delaunay-based vertex insertion and the progressive tetrahedralization methods. Performance statistics for this experiment are shown in Figure 3. The timing figures for the PT do not include the initial transformation into the PT representation, which took 10'40" without and 41'3" with full intersection testing. Three tetrahedral intersections were detected during the process and the corresponding collapse operations have been evaded. The original mesh and its approximation have both been sampled at 1,000,000 randomly distributed positions to determine the approximation error.

It is interesting to see that the lines of the distortion graph for both methods take an almost identical course. The performance of the PT methods, however, drops off for the last sample at 10,000 tetrahedra. This behavior can be explained by the fact that the order of priority of the edge collapses is determined by a greedy algorithm. After large number of successful edge collapse operations, it becomes more an more

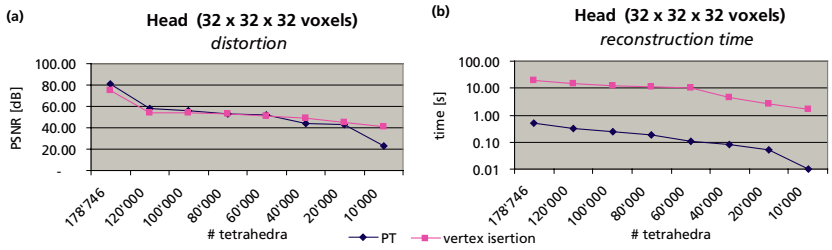

Fig. 3. Performance statistics for the volume comparison experiment; (a) PSNR of both methods for decreasing number of tetrahedra; (b) reconstruction times (note the logarithmic scale of the time axis).

difficult to select edges with low cost without compromising the correct topology of the mesh. This fact can usually be neglected for two-manifold meshes where the topology is much simpler, but it has a considerable influence on the volumetric setting. A global optimization scheme would certainly improve the mesh quality at low approximation levels, but this is probably not feasible for large applications.

Without taking preprocessing into account, the PT method clearly outperforms the Delaunay-based vertex insertion scheme in terms of reconstruction time by almost two orders of magnitude. The bottleneck of the latter scheme, however, is the Delaunay tetrahedralization which only performs in $O(N \log N)$, where N is the number of voxels. A faster tetrahedralization scheme could significantly improve the overall performance.

8 Conclusions

In this paper we have presented a comparison of different geometry compression and reconstruction methods for uniform data in two- and three-dimensional setting. Results from these methods have been compared to the popular progressive mesh algorithm and our extension to unstructured tetrahedral meshes.

Bottom-up Vertex Removal. The main advantage of the approach described in Section 4 is its fast and elegant implementation, which does not require any geometric calculations upon triangulation. Due to the constraints of the look-up-table, however, the semi-structured approximation limits the adaptivity of the resulting mesh. This is reflected in the numerical results presented in Section 7.1, which show that we cannot achieve the same performance as with the other two methods presented in this paper. Furthermore, the bottom-up nature of the algorithm prevents us from allowing for progressive transmission and reconstruction without precomputing the hierarchy. Before we can start with the construction of the quadtree, we have to receive all detail information at the finest level, which comprises two thirds of the original data (uncompressed).

Top-down Vertex Insertion. Using the top-down method from Section 5 instead of a bottom-up approach opens the possibility for channelwise progressive reconstruction. By transmitting coarse wavelet channels first, which only comprise a small subset of the total number of coefficients, we can refine the geometric representation adaptively. Since we are not constrained by a look-up-table, but employ Delaunay triangulation to determine the mesh connectivity, we can generate approximations which adapt better to the original data. In terms of the geometric approximation error, this method clearly outperforms the quadtree-based scheme. On the other hand, the algorithmic complexity of the vertex insertion method, which is largely determined by the cost of the Delaunay triangulation, is considerably higher. Hence, for very large data sets and applications where fast triangulation on the fly is more important than very low approximation error, the quadtree-based method is still attractive.

Progressive Tetrahedralizations. The accuracy of the progressive mesh and the progressive tetrahedralization methods is comparable to that of our Delaunay-based vertex insertion scheme. Direct comparison for surface triangulations has shown that we could even achieve better results with progressive meshes. This does not come as surprise, because we were able to employ data-specific cost functions which govern the transformation from the input mesh into the progressive mesh representation. Similar results could be obtained for tetrahedral meshes. For very coarse approximations, however, we have realized that the greedy method that determines the ordering of the edge collapse operations, is additionally constrained by the topological considerations described earlier. For that reason, we could not obtain the same accuracy as with our vertex insertion scheme under those circumstances. Another advantage of progressive tetrahedralizations, besides its fine grain progressivity, is its very short reconstruction time – even for larger models. This clearly compensates for the time consuming transformation comprising cost function evaluations, consistency checks and edge collapses, which can be carried out offline as a preprocessing step, though.

Acknowledgments

We would like to thank Markus Gross and the members of the Computer Graphics Lab at ETH Zürich for their support of this work. We would further like to thank the Swiss Federal Office of Topography for the DHM-25 model of the Matterhorn and Advanced Visual Systems Inc. for the CT volume data set.

References

1. C. Barber, D. Dobkin, and H. Huhdanpaa. The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software*, January 9 1995.

2. M. D. Buhmann and C. A. Micchelli. Spline prewavelets for non-uniform knots. *Numerische Mathematik*, 61(4):455–474, May 1992.
3. C. K. Chui and J. Z. Wang. A cardinal spline approach to wavelets. *Proc. Amer. Math. Soc.*, 113:785–793, 1991.
4. P. Cignoni, D. Costanza, C. Montani, C. Rocchini, and R. Scopigno. Simplification of tetrahedral meshes with accurate error evaluation. In *IEEE Visualization 2000*, pages 85–92, Oct. 2000.
5. P. Cignoni, C. Rocchini, and R. Scopigno. Metro: Measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, June 1998.
6. T. Dey, H. Edelsbrunner, S. Guha, and D. Nekhayev. Topology preserving edge contraction. Technical report, Raindrop Geomagic Inc., Research Triangle Park, NC, 1998.
7. D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to present a digitized line or its caricature. *The Canadian Cartographer*, 10(2):112–122, December 1973.
8. E. Quak and N. Weyrich. Decomposition and reconstruction algorithms for spline wavelets on a bounded interval. *Appl. Comput. Harmon. Anal.*, 1(3):217–231, 1994.
9. M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Computer Graphics (SIGGRAPH '97 Proceedings)*, pages 209–216, Aug. 1997.
10. M. H. Gross. L^2 optimal oracles and compresion strategies for semi-orthogonal wavelets. Technical Report 254, Computer Science Department, ETH Zurich, 1996.
11. M. H. Gross, R. Gatti, and O. Staadt. Fast multiresolution surface meshing. In *Proceedings of IEEE Visualization '95*, pages 135–142. IEEE Computer Society Press, 1995.
12. M. H. Gross, O. G. Staadt, and R. Gatti. Efficient triangular surface approximations using wavelets and quadtree data structures. *IEEE Transactions on Visualization and Computer Graphics*, 2(2), June 1996. ISSN 1077-2626.
13. H. Hoppe. Progressive Meshes. In *Computer Graphics Proceedings, Annual Conference Series, 1996 (ACM SIGGRAPH '96 Proceedings)*, pages 99–108, 1996.
14. J. M. Lounsbery. *Multiresolution Analysis for Surfaces of Arbitrary Topological Type*. PhD thesis, University of Washington, Seattle, 1994.
15. W. B. Pennebaker and J. L. Mitchell. *JPEG Still Image Data Compression Standard*. Van Nostrand Reinhold, New York, 1993.
16. J. Popović and H. Hoppe. Progressive simplicial complexes. In *Proceedings of SIGGRAPH 97*, pages 217–224. ACM SIGGRAPH / Addison Wesley, Aug. 1997. ISBN 0-89791-896-7.
17. R. Ronfard and J. Rossignac. Full-range approximation of triangulated polyhedra. In *Proceedings of EUROGRAPHICS '96*, pages C67–C76, 1996.
18. J. Shewchuk. Triangle: A two-dimensional quality mesh generator and delaunay triangulator. Technical report, Carnegie-Mellon Institute, 1995.
19. O. Staadt. *Multiresolution Representation and Compression of Surfaces and Volumes*, volume 12 of *Selected Readings in Vision and Graphics*. Hartung-Gorre Verlag, Konstanz, Germany, 2001. ISBN 3-89649-707-3.
20. O. G. Staadt and M. H. Gross. Progressive tetrahedralizations. In *Proceedings of IEEE Visualization '98*, pages 397–402, 1998.
21. O. G. Staadt, M. H. Gross, and R. Weber. Multiresolution compression and reconstruction. In *IEEE Visualization '97*, pages 337–346, Nov. 1997.
22. E. J. Stollnitz, T. D. DeRose, and D. H. Salesin. *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann, San Francisco, CA, 1996.
23. I. J. Trotts, B. Hamann, K. I. Joy, and D. F. Wiley. Simplication of tetrahedral meshes. In *Proceedings of IEEE Visualization '98*, pages 287–296, 1998.

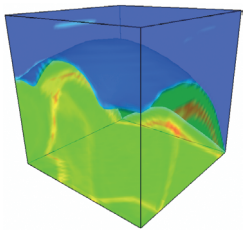

Fig. 4. Fragment of the child's skull data set which is employed for the comparison of the Delaunay-based vertex insertion and the progressive tetrahedralization methods. (Data set courtesy of Advanced Visual Systems Inc.)

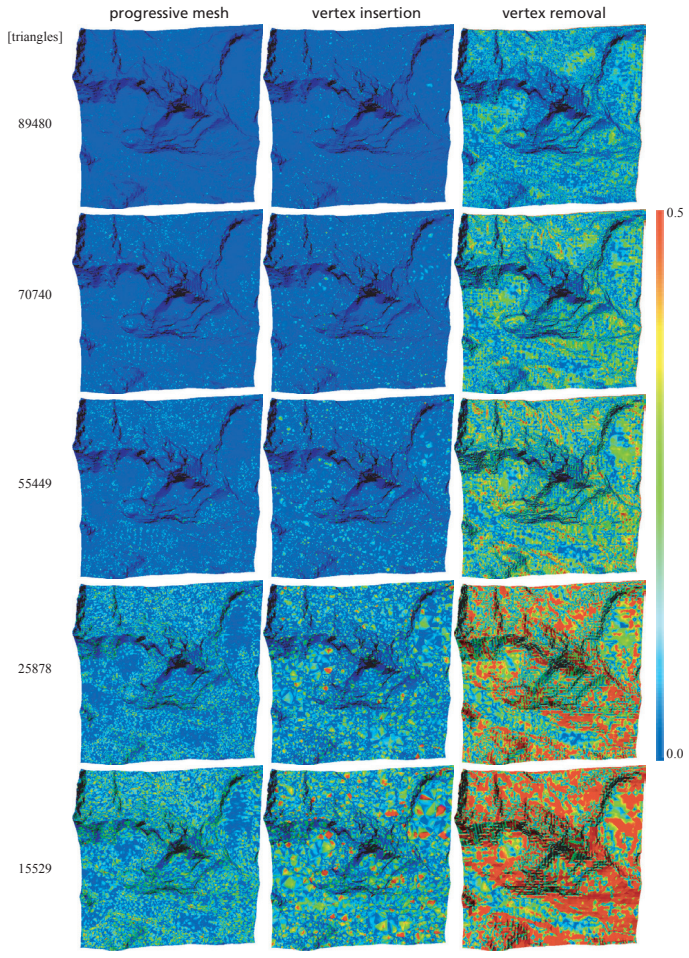

Fig. 5. Error visualization of the Matterhorn model for the three different mesh approximation methods at various reconstruction levels. The magnitude of the absolute mean-square geometric error corresponds to the color bar on the right.