Half-Edge Multi-Tessellation: A Compact Representation for Multi-Resolution Tetrahedral Meshes

Emanuele Danovaro, Leila De Floriani
Department of Computer and Information Sciences (DISI),
University of Genova, Via Dodecaneso 35, 16146 Genova - Italy
{danovaro,deflo}@disi.unige.it

Abstract

This paper deals with the problem of analyzing and visualizing volume data sets of large size. To this aim, we define a three-dimensional multi-resolution model based on unstructured tetrahedral meshes, and built through a half-edge-collapse simplification strategy, that we call a Half-Edge Multi-Tessellation (MT). *We propose a new compact data structure for a half-edge MT, and we analyze it with respect to both its space requirements and its efficiency in supporting Level-Of-Detail (LOD) queries based on selective refinement.*

1 Introduction

Several applications, including scientific visualization, medical imaging, and finite element analysis, deal with increasingly large sets of three-dimensional data describing scalar fields, called volume data sets. In order to analyze volume data sets of large size and to accelerate their rendering, a multi-resolution approach can be used. Multi-resolution meshes have been used for describing surfaces and two-dimensional height fields (see [6] for a survey). They encode the steps performed by a simplification process within a compact structure, in such a way that a virtually continuous collection of simplified meshes at different Levels-Of-Detail (LODs) can be extracted on-line. By applying a multi-resolution approach to tetrahedral meshes, we may have the resolution (i.e., the density of the cells) of the approximating mesh varying in different parts of the field domain (e.g., inside a box, or along a cutting plane), or in the proximity of interesting field values. This will enable a user to interactively explore large volume data using simplified approximations, and to inspect specific areas of interest.

In the computer graphics and finite element literature, a lot of research efforts have been devoted to nested tetrahedral meshes generated by recursive decomposition, which are suitable for dealing with regularly distributed data points (see [7, 8, 13, 14, 16, 21]). Applying nested decompositions to irregularly-distributed data points would require resampling, with problems in dealing with non-convex domains and with the spatial variation of such data. On the other hand, multi-resolution models based on irregular tetrahedral meshes are desirable since they are highly adaptive and can capture the shape of the field domain accurately even at the lowest resolution. But, no much research has been performed on such models. There have been proposals in the literature for simplification algorithms for irregular tetrahedral meshes, based on edge collapse [1, 9, 20], or on vertex insertion [11, 18], and on multi-resolution models, based either on a progressive [9, 17] or on a multi-level approach [2, 15].

In [5], we have defined a general multi-resolution model based on d-dimensional simplicial complexes, called a *Multi-Tessellation (MT)*, which is both dimension- and application-independent, and provides a framework for continuous multi-resolution modeling based on meshes. Here, we exploit the ideas underlying such general-purpose model to define a specific three-dimensional multi-resolution model based on unstructured tetrahedral meshes, and built through a specific edge-collapse simplification strategy, that we call a *Half-Edge Multi-Tessellation (MT)*. We propose a new compact data structure for a half-edge MT. We show that this data structure provides very good compression ratios not only with respect to a data structure which encodes a general-purpose three-dimensional Multi-Tessellation, but also with respect to encoding the original mesh at full resolution. The data structure is analyzed with respect to both its space requirements and its efficiency in supporting the primitives for implementing *selective refinement*, i.e., the process of extracting variable-resolution meshes in an incremental way.

2 Background

A *volume data set* consists of a set V of points spanning a domain D in the three-dimensional Euclidean space, with a field value f associated with each of them. A *tetrahedral mesh* Σ is a connected set of tetrahedra such that the union of all tetrahedra in Σ covers D, any two distinct tetrahedra have disjoint interiors and the intersection of the boundaries of any two tetrahedra of Σ, which have a non-empty intersection, consists of lower dimensional simplexes which belong to the boundary of both tetrahedra. Although, theoretically, the number m in tetrahedra in a mesh Σ can be quadratic in the number n of vertices of Σ, in practice, we have $m \approx 6n$.

Given a volume data set S, an *approximated* tetrahedral mesh is a mesh Σ' having m' ($m' < m$) tetrahedra and vertices at a subset V' of the original data set V, with n' ($n' < n$) points. A scalar field f' is defined on Σ', similarly to f, with the convention that values of f and f' are the same on each vertex that belongs to both V and V'. The approximation error associated with Σ' is the error that we perform in using Σ' instead of Σ for describing S. The *error* associated with each tetrahedron is a combination of the *field error* and of the *domain error*. In the simplification algorithm that we use [1], the field error at a tetrahedron σ is computed as the maximum of the absolute value of the difference between the actual field value at the points of $V \setminus V'$ inside σ and the field value at the same points linearly interpolated within σ. The domain error at a tetrahedron σ is computed as the maximum value of the one-sided Hausdorff distances of the points of the domain from tetrahedron σ, and it is not null only if σ is close to the boundary of Σ.

3 The Multi-Tessellation

A *Multi-Tessellation (MT)* $M = (\Sigma_b, U_R, R)$ consists of an initial mesh Σ_b subdividing the domain, that we call the *base mesh*, a set of updates $U_R = \{u_1 \dots u_k\}$, and a relation R of direct dependency among updates.

An *update* applied to a a mesh Σ consists of a pair of meshes $u = (u^-, u^+)$, where u^- is a sub-mesh of Σ, and Σ can be modified by replacing u^- with u^+ in such a way that u^+ fills the hole left in Σ after the removal of u^- and that the resulting mesh is still a tetrahedral mesh. Relation R of direct dependency is defined as follows: an update u_i depends on an update u_j iff $u_i{}^-$ removes some tetrahedra introduced by $u_j{}^+$. The transitive closure of relation R is a partial order. The updates in M will be also called the *nodes* of the MT. The mesh at the full resolution, that we term the *reference* mesh, can be obtained by applying all updates in U to the base mesh. Figure 1 (a) shows a simple example of a two-dimensional MT.

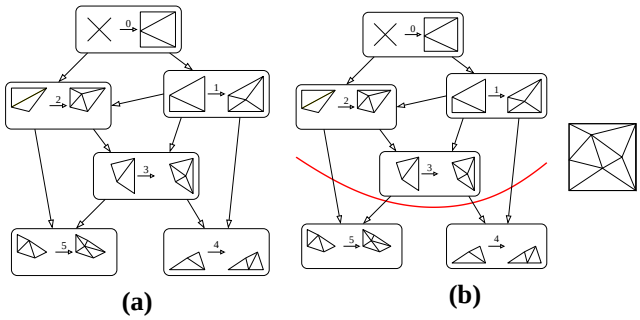

Figure 1. (a) An example of a two-dimensional MT. (b) A consistent set with the corresponding extracted mesh

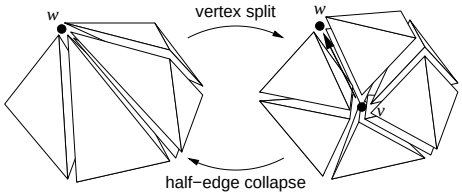

Figure 2. An example of a half-edge collapse.

We say that a subset U of nodes of an MT is *consistent* if, for every node $u \in U$, each node u' such that u'^+ precedes u^+ is also in U. The updates which form a consistent subset U can be applied to the base mesh in any total order that extends the partial order, thus producing a mesh at an intermediate Level Of Detail (LOD), that we denote with Σ_U. Figure 1 (b) shows an example of a consistent set and of the corresponding extracted mesh.

A half-edge MT is a Multi-Tessellation based on a specific update, called a *half-edge collapse*, that consists of contacting an edge $e = (v, w)$ of u^+ into one of its extreme vertices, say w. The reverse modification of a half-edge collapse is a vertex split, which expands vertex w into an edge e by inserting the other extreme vertex v of e (see Figure 2). In [3] we have defined an instance of a three-dimensional Multi-Tessellation based on a *full-edge collapse*. A full-edge collapse consists of contracting an edge e, with extreme vertices v' and v'', to a new vertex v (often the midpoint of e). The data structure proposed in [3] is specific for full-edge collapses, since it exploits the fact that each collapse generates a new vertex, and thus, it is not suitable for encoding half-edge collapses. Also, a full-edge collapse has the disadvantage of producing larger updates in comparison with those generated by a half-edge collapse (see Section 5).

4 A Data Structure for a Half-Edge MT

There are two basic ingredients in encoding a Half-Edge Multi-Tessellation: encoding the direct dependency relation, and encoding the update. The base mesh is encoded as a standard data structure for tetrahedral meshes which stores connectivity and face-adjacencies between tetrahedra.

The direct dependency relation is described as a DAG encoded by using a technique proposed by Klein and Gumhold [12]. For each node in the DAG, which corresponds to an update u, a cyclic linked list, called a *loop*, is defined, which contains the update u followed by all its direct ancestors in the DAG. A node u appears in its own loop and in all the loops defined by its direct descendants. Thus, at a node u, we store the number of loops to which u belongs, and, for each loop to which u belongs, a forward pointer implementing the linked list plus the loop identifier which is used to identify the loop each node belongs to. The total number of links to describe the arcs of the DAG is thus equal to $n + a$, where n is the number of vertices in the reference mesh (and thus an upper bound to the number of nodes in the MT), and a is the number of arcs in the DAG. Experimentally, we have found that a is equal to $5n$ on average, for MTs built through half-edge collapse and, thus, we have evaluated the cost of storing the DAG to be equal $6n \log n + 35n$ bits.

The encoding of an update u requires storing information for performing vertex splits and half-edge collapses, i.e., replacing u^- with u^+ and vice-versa. To perform a vertex split, we need to store the coordinates of the vertex v introduced, the value of the field at v, an error value, $\varepsilon(u)$, which provides an estimate of the approximation error associated with u and is computed as the maximum of the errors associated with the tetrahedra forming u^+, plus a compact encoding of the topological structure of u^-. Note that we usually store the error associated with an update and not with each tetrahedron forming it to obtain a more economical representation. To perform a half-edge collapse, we need to encode the vertex w on which edge $e = (v, w)$ is contracted. Since an update u corresponds to the insertion of a vertex v, updates and vertices are re-numbered in such a way that a node u and its corresponding vertex v have the same label. Thus, the relation between u^+ and v is encoded at a null cost.

Encoding the topology of u^- requires encoding a face f of the star-shaped polyhedron Π bounding u^- plus a bit stream which describes a traversal of the tetrahedral subdivision u^- starting at f. A boundary face f is described by a tetrahedron σ_{u^-} in u^- containing f plus the index of f within σ_{u^-}. Such an index can be encoded with 2 bits. Tetrahedron σ_{u^-} is encoded as described below.

Mesh u^- is described as a *tetrahedron spanning tree* rooted at σ_{u^-}, encoded as a bit stream following an approach similar to [10]. The tetrahedron spanning tree is constructed as follows. We start from σ_{u^-} in u^-, and traverse the graph formed by the tetrahedra of u^- and by their faces in breadth first. A face of a tetrahedron, which is common to another tetrahedron in u^- is labeled 1, it is labeled 0 otherwise. In this process, exactly three faces are labeled for each tetrahedron (this is also true for the initial tetrahedron σ_{u^-}, since we know one of its faces, i.e., f). If u^- contains k tetrahedra, then the bit stream contains $3k$ bits, since the length of the bit stream does not need to be stored. Our experiments have shown that we can safely assume $k = 12$. Then, in order to perform a vertex split , we start from the encoded boundary face f, and use the bit stream as a mask to retrieve all tetrahedra of u^- by visiting them in the same sequence as they have been visited when creating the tetrahedron spanning tree.

In order to perform a half-edge collapse on an edge $e = (v, w)$, we need to identify vertex w among the vertices of u^+. Then, we traverse the star of vertex v to identify the faces of the boundary polyhedron Π of u^+ not containing w. Those faces will be connected to w to form the tetrahedra in u^-. To identify vertex w in u^+, we need to encode a tetrahedron σ_{u^+} in the star of v containing w and then the index of w in σ_{u^+}. Storing such an index requires 2 bits.

We describe now how we encode tetrahedra σ_{u^-} in u^- and σ_{u^+} in u^+. When applying an incremental algorithm for selective refinement, a tetrahedron $\sigma \in \Sigma_U$ is generated either during refinement or during coarsening. Thus, a tetrahedron σ is labeled with one bit to discriminate between the two cases, and with an integer, which uniquely identifies σ among the tetrahedra that have been inserted in Σ_U together with σ. The integer label is encoded on $\log P$ bits, where P denotes the maximum number of direct descendants and ancestors of a node.

When we perform an update u^+ in Σ_U, we label the new tetrahedra (i.e., those of u^+) in an (arbitrary) order which is always the same every time the update is performed, and such that tetrahedron σ_{u^+} containing f is the first one. Similarly, when we perform an update u^-, we label the tetrahedra of u^- in an (arbitrary) order which is always the same every time the update is performed, and such that tetrahedron σ_{u^-} containing vertex w is the first one. Thus, for each update u, we encode an index (on $\log P$ bits) that identifies the tetrahedron $\sigma_{u^-} \in u^-$ having f as one of its faces, and an index (on $\log P$ bits) that identifies the tetrahedron $\sigma_{u^+} \in u^+$ having w as one of its vertices. Encoding such information requires $2 \log P$ bits per update, i.e., $10n$ bits, since the construction algorithm enforces P to be equal to 32, and the number of updates is bounded by n.

In order to retrieve σ_{u^-} from the index stored in u, we need to know the direct ancestor u' of u such that $\sigma_{u^-} \in u'^+$. Update u' is encoded at no cost by adopting the convention that, in the DAG encoding, the list of the direct ancestors of u contains u' in its first position. The index

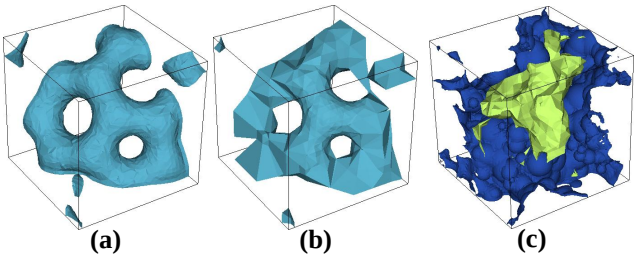

Figure 3. (a) Uniform LOD extraction: error threshold equal to 0.1% of the field range. (b) Variable LOD based on a region of interest: mesh extracted with a threshold equal to 0.1% of the field range in a selected box, and arbitrary large outside (the isosurface for a field value equal to 105.000 is shown). (c) Variable LOD based on field values: a mesh extracted with a threshold equal to 0.1% of the field range on the tetrahedra intersected by the isosurface with field value equal to 1.27 (shown in dark gray), and arbitrary large outside. The second isosurface, with a field value equal to 1.45 (shown in light gray), illustrates the lower resolution of the mesh in the region formed by the tetrahedra that do not intersect the selected isosurface.

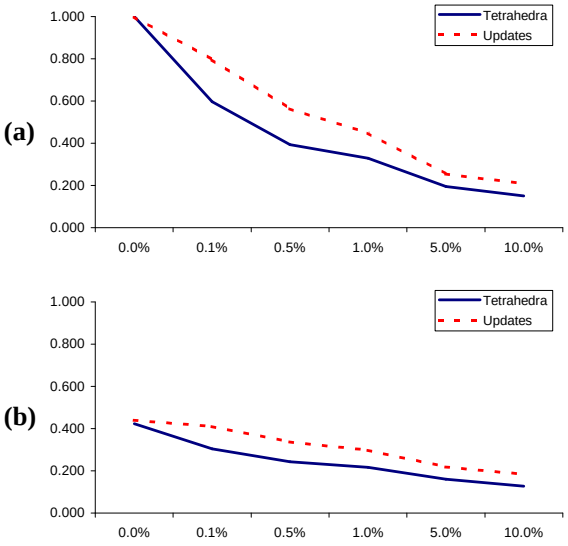

Figure 4. (a) Number of tetrahedra in a mesh at a uniform LOD with different error thresholds extracted from a half-edge MT with errors on updates and with errors on tetrahedra (with respect to the number of tetrahedra in the reference mesh). (b) Number of tetrahedra in a mesh at a variable LOD with different error thresholds extracted from a half-edge MT with errors on updates and with errors on tetrahedra (with respect to the number of tetrahedra in the reference mesh).

stored with an update u to identify σ_{u^-} is the same as the label assigned to σ_{u^-} when σ_{u^-} is created in the current mesh by performing update u'^+. This information, together with node u', enables us to retrieve σ_{u^-} in the current mesh when σ_{u^-} is in Σ_U because it has been created by performing u'^+. If σ_{u^-} has been added to Σ_U by performing u^-, then σ_{u^-} is simply retrieved as the tetrahedron with the minimum label among those inserted by u^-. In a similar way, σ_{u^+} is retrieved from the index stored in u.

The cost of encoding the connectivity information for an update u is equal to 50 bits, since 36 bits are required to encode the tetrahedron spanning tree, 10 bits are required to encode the tetrahedra and 4 bits are required to encode face f and vertex w. The cost of encoding geometric information, and the field values is equal to 8 bytes, while the error value is encoded on 2 bytes. Thus, the storage cost for the information associated with a single update contributes for a cost of 130 bits. Therefore, the total cost of a half-edge MT data structure (including the cost of encoding the direct dependencies) is equal to $165n + 6n \log n$ bits. If we associate an error value not just with each update, but with each tetrahedron, the total cost of storing the error values is equal to $24n$ bytes. Thus, the cost of storing a half-edge MT would increase in this case to $441n + 6n \log n$ bits, i.e., of about 50%.

The storage cost of the half-edge MT is equal 15% of the space needed by encoding a three-dimensional MT in which the tetrahedra are explicitly stored as a 4-tuple of vertices. It is about 26% of the cost of storing the reference mesh with connectivity and face-adjacency information, and 53% of the cost of encoding the reference mesh in an indexed structure which encodes only connectivity information. Storing a half-edge MT costs 1/5 more than encoding a full-edge MT with errors associated with updates. On the other hand, the half-edge MT is about 70% of the full-edge MT when we consider MTs with errors associated with tetrahedra.

5 Results

In the experiments shown here the construction of a half-edge MT has been performed by using the algorithm in [1]. We have used two regular volume data sets: *Smallbucky*, which is a portion of the well-known regular Bucky-Ball data set and has 32,768 vertices. *Plasma*, which is a large synthetic data set with 262,144 vertices, and two irregular data sets: *Flame*, with 19611 vertices, and *Fighter*, with 13832 vertices.

We have found that the number of tetrahedra in u^- is equal to 10.5, while the number of tetrahedra in u^+ is equal to 16, on average. Thus, the total number of tetrahedra in a half-edge MT is equal to $16n$. The overhead introduced by a half-edge MT, evaluated as the ratio between the number

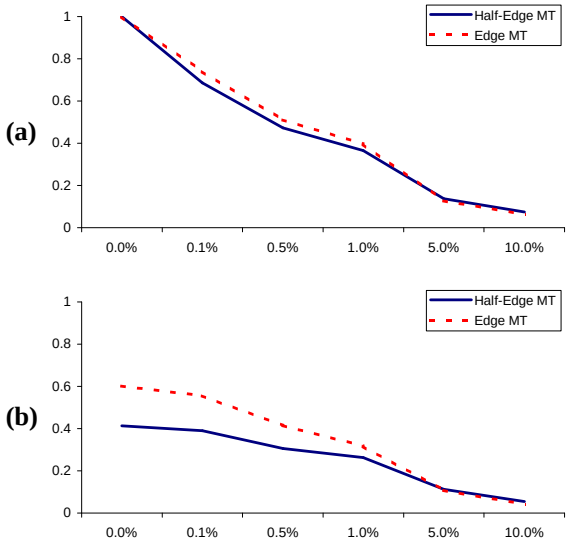

Figure 5. (a) Number of tetrahedra (with respect to the number of tetrahedra of the reference mesh) in a mesh extracted from a half-edge and from a full-edge MT at a uniform LOD with different error thresholds. (b) Number of tetrahedra (with respect to the number of tetrahedra of the reference mesh) in a mesh extracted from a half-edge and from a full-edge MT at a variable LOD based on field value with different error thresholds.

of tetrahedra in the MT and the number of tetrahedra in the reference mesh (which is equal to $6n$) is about 2.65. In a full-edge MT the number of tetrahedra in u^- and u^+ is, on average, equal to 27 and 33, respectively. The overhead introduced by a full-edge MT is about 5.5.

We have evaluated the shape of the tetrahedra by using two measures commonly applied in the finite element literature: the *circumradius-to-shortest-edge* ratio r (where the circumradius is the radius of the circumsphere of a tetrahedron), and the minimum solid angle α associated with a tetrahedron [19]. Our experiments have shown, on average, a value of r equal to 1.38, and a value of α equal to 19.6. Note that, in a Delaunay tetrahedral mesh, r is equal to 1.046, and α is equal to 20.5, on average.

We have performed the experiments based on the Level-Of-Detail (LOD) queries defined in [3] as instances of the general selective refinement query (see Figure 3). The comparisons in the different experiments are in terms of the number of tetrahedra in the extracted mesh (i.e., its size) with respect to the number of tetrahedra in the reference mesh. The size of the extracted mesh is directly related to the complexity of the queries as the execution time of the selective refinement algorithms depends on such parameter. We have referred the size of the extracted mesh to that of the reference mesh in order to compare the results obtained with different data sets.

In the first experiment we have compared the number of tetrahedra in meshes extracted at a uniform LOD and at a variable LOD based on field values, with different error thresholds, from a half-edge MT with errors on the updates, and an MT with errors on the tetrahedra (see Figure 4). The results show that the size of a mesh extracted from an MT with error on tetrahedra is between 25% and 30% smaller than the size of a mesh extracted from an MT with error on updates in both tests.

In the second experiment, we have compared the performances of a half-edge MT and of a full-edge MT, both storing the errors with the updates, for uniform LOD extractions (see Figure 5 (a)), and for extractions at variable LODs based on the field values (see Figure 5 (b)). At a variable LOD the size of the meshes extracted from the half-edge MT is about 20% less that the size of those extracted from the full-edge MT, while this percentage reduces to 8% for extractions at a uniform LOD. This is motivated by the larger size of the updates in a full-edge MT and thus by a lower number of dependency links in the DAG describing it.

6 Concluding Remarks

We have proposed a new data structure for a class of multi-resolution tetrahedral meshes, called a Half-Edge Multi-Tessellation. This data structure is not only considerably more compact than an efficient implementation of a general MT data structure, but it acts as a compression mechanism also with respect to storing the original mesh at full resolution. It is also more selective than a data structure developed for encoding multiresolution tetrahedral meshes built through full-edge collapses. Moreover, meshes with connectivity and adjacency information can be extracted from a half-edge MT at no extra cost. In [4], we have compared irregular multiresolution tetrahedral meshes with regular ones built through tetrahedron bisection.

We have extended the data structure presented here to represent MTs in which updates are general vertex insertions/removals. In this way, an MT could be built by incremental top-down refinement on a Delaunay mesh. To this aim, we need to develop heuristics for dealing with volume data spanning a non-convex domain.

Finally, we plan to apply the work presented in this paper to perform LOD operations on irregular meshes describing 3D scalar fields in a client-server environment. The use of our compact data structure will enable both a progressive and a selective download of the extracted mesh by a client and allows for a dynamic selective refinement at each new request from a client.

7 Acknowledgments

This work has been performed while Leila De Floriani has been visiting the Computer Science Dept. of the University of Maryland, College Park (MD). This work has been partially supported by the project funded by the INDAM National Group on Scientific Computation (GNCS) on "Data Structures and Algorithms for Scientific Data Visualization" and by the project funded by the National Research Council of Italy on "Efficient Modeling and transmission of three-dimensional objects and scenes" under contract #CNRC00FE45_004.

References

[1] P. Cignoni, D. Costanza, C. Montani, C. Rocchini, and R. Scopigno. Simplification of tetrahedral volume with accurate error evaluation. In *Proceedings IEEE Visualization 2000*, pages 85–92, Salt Lake City, UT, October 2000.

[2] P. Cignoni, L. De Floriani, C. Montani, E. Puppo, and R. Scopigno. Multiresolution modeling and visualization of volume data based on simplicial complexes. In *Proceedings 1994 Symposium on Volume Visualization*, pages 19–26, Washington, DC, October 1994.

[3] P. Cignoni, L. De Floriani, P. Magillo, E. Puppo, and R. Scopigno. *TAn2* - visualization of large irregular volume datasets. Technical Report DISI-TR-00-07, Department of Computer and Information Science, University of Genova (Italy), 2000. (submitted for publication).

[4] E. Danovaro, L. De Floriani, M. Lee and H. Samet. Multiresolution tetrahedral meshes: an analysis and a comparison In *Proceedings Shape Modeling International 2002*, Banff, Canada, May 17-22, 2002, in print.

[5] L. De Floriani, E. Puppo, and P. Magillo. A formal approach to multiresolution modeling. In R. Klein, W. Straßer, and R. Rau, editors, *Geometric Modeling: Theory and Practice*, pages 302–323. Springer-Verlag, 1997.

[6] M. Garland. Multiresolution modeling: Survey & future opportunities. In *Eurographics '99 – State of the Art Reports*, pages 111–131, 1999.

[7] R. Grosso and G. Greiner. Hierarchical meshes for volume data. In *Proceedings of the Conference on Computer Graphics International 1998 (CGI-98)*, pages 761–771, Los Alamitos, California, June 22-26 1998. IEEE Computer Society.

[8] G. Greiner and R. Grosso. Hierarchical tetrahedral-octahedral subdivision for volume visualization. *The Visual Computer*, 16:357–365, 2000.

[9] M.H. Gross and O.G. Staadt. Progressive tetrahedralizations. In *Proceedings IEEE Visualization'98*, pages 397–402, Research Triangle Park, NC, 1998. IEEE Comp. Soc. Press.

[10] S. Gumhold, S. Guthe, and W. Straßer. Tetrahedral mesh compression with the cut-border machine. In *Proceedings IEEE Visualization'99*, pages 51–58. IEEE, 1999.

[11] B. Hamann and J.L. Chen. Data point selection for piecewise trilinear approximation. *Computer Aided Geometric Design*, 11:477–489, 1994.

[12] R. Klein and S. Gumhold. Data compression of multiresolution surfaces. In *Visualization in Scientific Computing '98*, pages 13–24. Springer-Verlag, 1998.

[13] M. Lee, L. De Floriani, M., and H. Samet. Constant-time neighbor finding in hierarchical meshes. In *Proceedings International Conference on Shape Modeling*, pages 286–295, Genova (Italy), May 7-11 2001.

[14] M. Ohlberger and M. Rumpf. Adaptive projection operators in multiresolution scientific visualization. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):74–93, 1999.

[15] R. Pajarola, J. Rossignac, and A. Szymczak. Implant sprays: Compression of Progressive Tetrahedral Mesh Connectivity. In *Proceedings IEEE Visualization '99*, pages 299–305, 1999. IEEE Comp. Soc. Press.

[16] V. Pascucci and C. L. Bajaj. Time critical isosurface refinement and smoothing. In *Proceedings 2000 Symposium on Volume Visualization*, pages 33–42, October 2000.

[17] J. Popovic and H. Hoppe. Progressive simplicial complexes. In *ACM Computer Graphics Proceedings, Annual Conference Series, (SIGGRAPH '97)*, pages 217–224, 1997.

[18] K.J. Renze and J.H. Oliver. Generalized unstructured decimation. *IEEE Computational Geometry & Applications*, 16(6):24–32, 1996.

[19] M. Rivara and C. Levin. A 3d refinement algorithm suitable for adaptive and multi-grid techniques. *J. Comp. Appl. Math*, 8:281–290, 1992.

[20] I.J. Trotts, B. Hamann, and K.I. Joy. Simplification of tetrahedral meshes with error bounds. *IEEE Transactions on Visualization and Computer Graphics*, 5(3):224–237, 1999.

[21] Y. Zhou, B. Chen, and A. Kaufman. Multiresolution tetrahedral framework for visualizing regular volume data. In *Proceedings IEEE Visualization'97*, pages 135–142. IEEE Computer Society, 1997.