Multiresolution Tetrahedral Framework for Visualizing Regular Volume Data

Yong Zhou, Baoquan Chen, and Arie Kaufman*

Center for Visual Computing
and Department of Computer Science
State University of New York at Stony Brook
Stony Brook, NY 11794-4400

Abstract

We present a multiresolution framework, called Multi-Tetra framework, that approximates volume data with different levels-of-detail tetrahedra. The framework is generated through a recursive subdivision of the volume data and is represented by binary trees. Instead of using a certain level of the Multi-Tetra framework for approximation, an error-based model (EBM) is generated by recursively fusing a sequence of tetrahedra from different levels of the Multi-Tetra framework. The EBM significantly reduces the number of voxels required to model an object, while preserving the original topology. Our approach provides continuous distribution of rendered intensity or generated isosurfaces along boundaries of different levels-of-detail, thus solving the crack problem. Our model supports typical rendering approaches, such as Marching Cubes, direct volume projection, and splatting. Experimental results demonstrate the strengths of our approach.

CR Categories and Subject Descriptors: I.3.3 [Computer Graphics]: Picture/Image Generation – Display Algorithms; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling – Surface, solid, and object; I.4.10 [Image Processing]: Image Representation – Hierarchical, Volumetric representations.
Keywords: volume visualization, multiresolution volume, level of detail, isosurface extraction, volume subdivision, polygon simplification

1 Introduction

Many computer applications routinely generate volumetric models consisting of large numbers of voxels, which greatly exceed the capabilities of the typical graphics workstations, not only in memory requirements, but also in rendering speed. In order to accommodate complex volume data and accelerate rendering, methods for approximating the original data set by using a multiresolution model have been proposed [CDM+94, WV94]. Multiresolution representation of a volume can be used to generate multiple volume models at different levels of detail. For a given precision, display techniques are employed to select and render certain levels of the model instead of using the original data set.

*Email:{yzhou,baoquan,ari}@cs.sunysb.edu

In this paper we propose a new multiresolution tetrahedral (called Multi-Tetra) framework that can be employed by visualization techniques, such as Marching Cubes [LC87], Polygon Projection [ST90] and Splatting [LH91], to render regular volume data. By using the Multi-Tetra framework, a new volume, i.e., an error-based model with fewer sample points and voxels, can be established for approximating the original data set. Such a framework provides an efficient mechanism for continuous level of detail display. The algorithm has the following features:

- **Efficiency:** Large reduction in the number of volume elements with little or no loss in image quality
- **Practicality:** Different levels of the Multi-Tetra framework can be represented in a uniform way; it is easy to set up and index; any level of detail model based on the given error threshold can be obtained on the fly; furthermore, the model supports different precisions in different regions
- **Versatility:** Supports efficient extraction of isosurfaces, direct volume rendering through projective techniques with adaptive resolution levels, as well as the development of progressive and multiresolution rendering approaches
- **Continuity:** Easily solves the problem of crack between different levels of detail, allowing continuous changes of sample points density or generated isosurfaces
- **Locality:** High frequency data regions, such as transitions between bone and tissue in CT data sets, do not have a widespread global effect on the complexity of the model

The rest of the paper is organized as follows. After introducing related work in Section 2, we will discuss our Multi-Tetra framework (Section 3), including building an embedded and uniformly represented multiresolution framework (Section 3.1), an error-based level of detail model generation (Section 3.2) and approximation error analysis (Section 3.3). Necessary data structures and implementation details will also be given in Section 3.2. The experimental results are shown in Section 4.

2 Related Work

There have been a large number of methods proposed to speed up the visualization process using approximated models. Most are polygon-based algorithms, i.e., polygonal mesh simplification methods, which significantly reduce the complexity of polygons [SZL92, HDD$^+$93]. We are interested in volume data simplification rather than that of the graphics output, such as Schroeder's decimation of triangle meshes [SZL92].

Several researchers use the octree model to represent volume data sets, trying to skip empty regions or regions of no interest [LEV90, WG90]. In a sense, this is the early development of multiresolution models. In addition, researchers have used the octree model to approximate volume data in terms of surface construction. Renben's adaptive Marching Cubes and Raj's octree-based decimation are two examples [SZK95, SFY96]. Although both methods could reach a certain iso-polygon simplification effect, they suffer from the same crack problem. For connection on shared boundaries by two adjacent volume elements with different resolutions, they artificially add some patches to retile the crack - achieving a certain visual smoothness, but geometrically and topologically unreasonable, since there is no robust theory to prove it. Actually, the traditional octree model can not support continuous changes between different levels of detail, as any subdivision along a cell or subvolume results in a subdivision of an adjacent element if adjacent continuity must be retained, i.e., the octree-based model violates the above continuity.

Unlike the octree-based methods, Wilhelms and Van Gelder's multi-dimensional tree [WV94] provides spatial and temporal efficiencies for rendering large data sets. As the authors pointed out, however, their model does not support continuity between neighboring regions when they use the projective method to render volume data. Taosong He et al.'s voxel-based topology simplification algorithm [HHVW96] also provides a simple, robust and practical multi-resolution volume hierarchy, but it suffers from the same problem as the multi-dimensional tree does.

Cignoni's multiresolution modeling [CDM$^+$94] and, especially, Lindstrom's height field model [LKR96] are closely related to our Multi-Tetra framework. The coarsest paradigm of Cignoni's is the tetrahedralization of the convex hull of original data sets, followed by adding unprocessed nodes one-by-one, with the greatest error positioned first in the previously generated model. This process repeats itself until a given precision is met. This was a good idea, for they adopted tetrahedra as a basic volume element, appropriate for unstructured data sets. However, each time a node is added, the related part of the previous model needs to be tetrahedralized again; the computation is expensive. They improved the algorithm by preprocessing; the whole model of different precisions is uniformly embedded in a sequence of tetrahedra. It supports boundary continuity and versatility, but it is very difficult to maintain continuous transitions between two levels.

Lindstrom et al. present excellent work in real time, continuous level of detail rendering of height fields [LKR96]. In terms of polygonal meshes, their model supports continuity, locality, practicality. This is, however, a 2D polygon-based algorithm.

Some applications do not require the satisfaction of all the above properties, and most contemporary volume-based approaches fail to meet at least one of these properties. However, a volume-based multiresolution model that supports all of these features is of great importance in volume visualization. The Multi-Tetra framework proposed in this paper satisfies all of the above properties, borrowing some of Cignoni's ideas - using tetrahedra as a basic operation element, recursively building the uniform framework - and meanwhile, extending Lindstrom's subdivision mechanism to 3D case.

Our framework is set up based on a recursive partitioning of the volume domain into tetrahedra. The partitioning operation involves only vertex indices and not the physical coordinates of the vertices. In addition, the partitioning process does not add any new sample points, hence, it does not need interpolation for values of sampling points. It can be represented by a full binary tree and stored in a linear array, allowing for instant indexing. Unlike the traditional method [HHVW96] that uses a certain level of detail to approximate the original configuration (so an interpolation is required for the correspondence between different levels), our error-based model is obtained by visiting multilevels of tetrahedra according to given precisions. The model allows for smooth changes in resolution across areas of different levels during the rendering process, using different precisions in the same framework.

3 Multi-Tetra Framework

The Multi-Tetra framework is a level-of-detail approximation of a regular data volume. It consists of a set of uniformly represented and embedded tetrahedra, generated by a recursive subdivision of the volume. Through fusion operation on the multiresolution framework, an error-based model – consisting of a subset of non-embedded tetrahedra extracted from the Multi-Tetra framework – can be obtained to approximate the volume. In other words, a simplified model with fewer tetrahedra can be used as a substitute for speeding up the rendering of the volume data.

In this section, we first discuss the establishment of the multiresolution framework, followed by the generation of a concrete model, i.e., an error-based model, to meet given precisions. Essentially, the building process of the framework is the subdivision of a volume, while the error-based mode generation is a selective fusion of tetrahedra. The initial input is a regular volume data set. For the subdivision mechanism, the data set is extended to dimension $(2^N + 1) \times (2^N + 1) \times (2^N + 1)$ by padding it with slices of zero.

3.1 Building the Multi-Tetra Framework

The generation of the Multi-Tetra framework occurs through a recursive subdivision of the volume data. The subdivision process includes initial subdivision and recursive subdivision, starting with the bounding box of the volume. These can be described as a vertex-adding process. Each step picks up a vertex from the original volume and puts it into an already generated framework, thus dividing a tetrahedron into two. Each vertex can be added only once and the subdivision continues until all vertices are added to the framework.

3.1.1 Initial Subdivision

The initial Multi-Tetra framework is the volume bounding box. For the sake of the following explanation, imagine that the box has eight vertices which coincide with the eight corners of the volume, where the inside is empty. Also, each face of the box parallels a coordinate plane.

For the initial subdivision, we add the center point of the volume to the initial framework – the volume box, and connect it to all the vertices of the box, forming 6 pyramids. Then we divide each pyramid into two tetrahedra by connecting the diagonal of the base face, generating 12 tetrahedra, such as the dashed tetrahedron in Fig. 2. It is important to note that each added vertex has the same index as before. A vertex index is: $i + j * (2^N + 1) + k * (2^N + 1) * (2^N + 1)$, where (i, j, k) is its grid index.

3.1.2 Recursive Subdivision

Next, we recursively subdivide each previously generated tetrahedron. Basically, at each step, the recursive subdivision procedure adds the midpoint of an edge of a tetrahedron. The point added is the one with an index equal to the average of index values of the two endpoints (i.e., the dotted points in Fig. 2). The result is that the tetrahedron is bisected into two by the plane passing through the added point and its opposite edge. At this point, our subdivision is similar to Hebert's decomposition for finite element meshes[HEB94]. In the following, the added points are called dividing points (DP).

For any generated tetrahedra after the initial subdivision, regardless of how they are recursively bisected in the recursive subdivision, their configuration must fall into one of three cases (Fig. 1):

- Case 1: There is only one face parallel to a coordinate plane and there exists only one edge L of the face NOT parallel to any coordinate axis
- Case 2: There is only one face parallel to a coordinate plane and there exists only one edge L of the face parallel to a coordinate axis
- Case 3: There are only two faces parallel to coordinate planes (the edge that does not belong to any of these two faces is denoted by L)

In Fig. 1, for any case, edge AB of tetrahedron ABCD corresponds to edge L. For each different case, the midpoint of corresponding edge L is selected as the dividing point. We refer to the subdivision of Case n as Step n $(n = 1, 2, 3)$.

Here is the procedure for the actual subdivision process. Note that after the initial subdivision, the 12 generated tetrahedra all belong to Case 1, so we first perform Step 1 subdivision (Fig. 2). After Step 1, the configuration of newly generated tetrahedra are in Case 2; then we perform Step 2 and move on to Step 3 subdivision. After the subdivision of Case 3, the tetrahedra configurations recursively return to Case 1. We perform Step 1 subdivision again if needed.

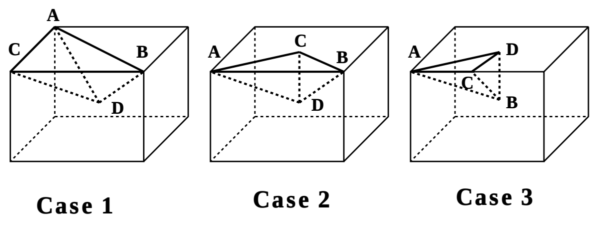


Figure 1: *Generated tetrahedra by the recursive subdivision must fall into one of the above three cases*

As mentioned above , a volume has the dimension of $(2^N + 1) \times (2^N + 1) \times (2^N + 1)$. Now assume the simplest case: $N = 1$ (i.e., for a $3 \times 3 \times 3$ volume). After the initial and Step 1 and Step 2 subdivisions, all the points of the original volume are added to the generated framework, and the subdivision ends. If $N = 2$ (i.e., for a $5 \times 5 \times 5$ volume), another three steps – i.e., Step 3, Step 1, successively followed by Step 2, are required. Generally, if $N = n$ (n is an arbitrary integer), n Steps 1, n Steps 2 and $n-1$ Steps 3 are required. The whole subdivision process is illustrated in Fig. 2; the dashed tetrahedron is one to be considered for next step subdivision.

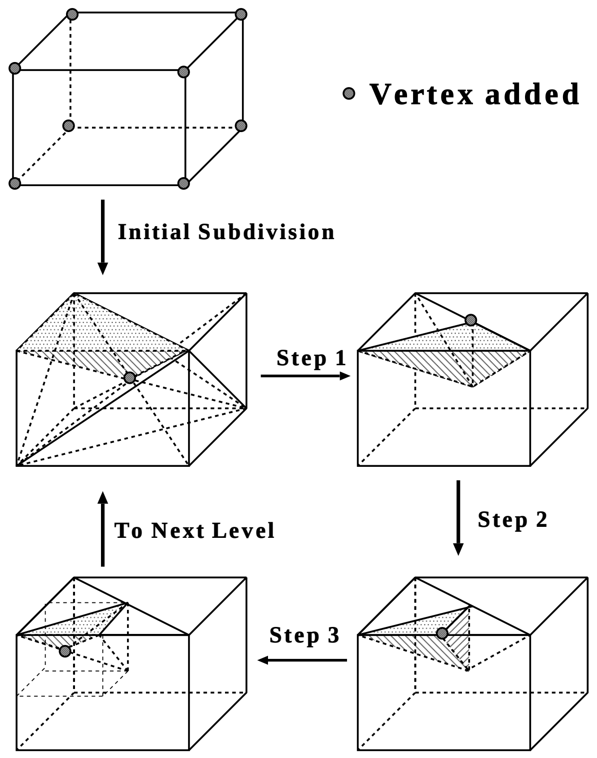


Figure 2: *The entire subdivision process*

The implementation of the recursive subdivision can follow two ways: *breadth priority subdivision* (BPS) and *depth priority subdivision* (DPS). BPS means the same subdivision step is implemented for all the tetrahedra in the same level before the next step is taken, while DPS means the same tetrahedron descends through all the steps before the other tetrahedra are subdivided.

BPS is a straightforward process; after each BPS step, all the tetrahedra undergo the same subdivision – a tetrahedron breaks into two. After one level is finished, the process proceeds to the next level. No matter which order is taken, when the subdivision finally stops, each original cell has been divided into six tetrahedra.

Note that during the subdivision, no physical coordinates and only vertices' indices are involved. In implementation we store all intermediate data. The subdivision can be represented by a full binary tree, easily be stored by an array; each item of the array corresponds to a tetrahedron structure. Next, we show how to employ this framework – the Multi-Tetra framework – to obtain an error-based model approximating the volume within given precisions.

3.2 Error-Based Model Generation

As we mentioned before, the purpose of the volume-based multiresolution representation is to approximate the original data set using fewer elements or tetrahedra. How to find a set of tetrahedra (i.e., an error-based model) from the Multi-Tetra framework is described next.

According to our Multi-Tetra framework, a set of tetrahedra in the same level of the binary tree is a brute-force approximation of the original data sets – the higher the resolution level the set of tetrahedra comes from, the more precise the approximation is. Obviously, visiting and visualizing leaf nodes is tantamount to traditional cell-by-cell rendering approaches, except that each cell is replaced by six tetrahedra. If we can merge tetrahedra, visiting a relatively small number of elements, the rendering time will be reduced geometrically in relationship to the degree of reduction.

Our approach is a bottom-up tetrahedra fusion, starting with the highest resolution level, i.e., leaf nodes of the tree, as in [LH91]. Each fusing operation is limited to two children tetrahedra descended from the same parent if the fusion criteria are met. The result is that two children are replaced by their parent. The resulting tetrahedra can be considered for further simplification in a recursive manner. The fusion continues until the fusion criteria are violated.

The fusion criteria are based on two factors: one is geometrically an approximation error; the other is a topologically adjacent relationship. Suppose the density of any point within a tetrahedron can be evaluated by trilinear interpolation of vertex densities. When two small tetrahedra are replaced by a large one, the approximation error occurs. There are many ways to evaluate the error depending on the rendering techniques. For instance, for isosurface extraction, the isovalue is a parameter of the error evaluation. In the current implementation, we used the method for isosurface extraction called Marching Tetrahedra [ZCT95]. The details will be given in Section 3.3.

From another perspective, the fusion can be regarded as the dividing point (DP) removal. Merging two tetrahedra means to remove the corresponding DP, and for boundary continuity, the necessary condition for removing the DP is that all the tetrahedra pairs with this DP should be qualified for fusion. The question is how to locate the tetrahedra sharing the same DP so that we can perform an approximation test.

For the cases generated by different steps, the number of shared tetrahedra pairs is distinctly different. Since each dividing point is associated with an edge, the tetrahedra pairs sharing a corresponding edge are what we want. By referring to Fig. 1 again, in each case, the number of tetrahedra sharing edge AB is our answer. Thus, we have the following results: for Case 1, the number is 4; Case 2, the number is 8, and for Case 3, it is 6. Generally, if a dividing point is shared by n pairs of tetrahedra, it is call n-connected (here suppose the dividing point is inside the volume, not on the boundaries; in that case, n is relatively smaller). Thus, for any case, all the dividing points are shared by limited tetrahedra, meaning the local fusion of tetrahedra does not have a widespread global effect on the complexity of our model.

For each dividing point we set up a dividing point structure, which stores the addresses of the tetrahedra sharing the dividing point and the maximum approximation error. For fusion implementation, the whole binary tree is represented by an array; each item of the array corresponds to a node, only storing vertex indices, not addresses of their children's nodes since the children addresses can be easily obtained by a simple calculation.

A dividing point table also has to be established with each level of the trees corresponding to a dividing point chain. Each item of the chain corresponds to a dividing point structure. The head pointers to chains are stored in the dividing point table.

Before fusing dividing points, maximum approximation errors are calculated starting from the highest level in a bottom-to-up order. Instead of checking the nodes in the trees, only the dividing point table is visited. For each dividing point, the error is taken as the maximum between the current level error and the error already stored in the error item from its children. The current level error is the maximum among the errors of tetrahedra pairs sharing the corresponding dividing point. Once the error evaluation for a dividing point is completed, its parent's error needs to be updated. After all dividing points are processed, we ascend to the next immediate level. This mechanism guarantees that if dividing point's children can not be fused, or any one pair of tetrahedra sharing it can not be fused, current fusion fails.

Once error evaluation is accomplished, an approximate model can be obtained according to a given precision. The model consists of nodes with their approximation errors less than the error threshold, while their parents' error exceeds the threshold. The model can be displayed by visualization methods. In Section 4, the isosurfaces extracted from the Multi-Tetra framework are given.

3.3 Error Analysis

We use the Marching Tetrahedra method [ZCT95] for isosurface extraction. Isosurface patches within a tetrahedron have three configurations shown in Fig. 3 (disregarding symmetrical cases). Unlike the Marching Cubes, the Marching Tetrahedra has no ambiguity in the reconstruction. Now, suppose a tetrahedron is bisected corresponding to edge L. For Cases a and b in Fig. 3, the subdivision derives Cases a1 and a2, and Cases b1 and b2, respectively, while for Case c different cases are derived, i.e., Cases c1-c3 (see Fig. 4). For convenience, we assume a point holds a positive sign if the density exceeds the isosurface threshold; otherwise, it holds a negative sign.

By analyzing all the cases in Fig. 4, we can say the densities of endpoints of edge L and the dividing point have substantial influence on the geometric and topological distribution of isosurfaces, especially when two endpoints hold the same sign, and the dividing point has an opposite sign, such as Case a2, Case b2, Case c3. In such a situation, the isosurface topology changes after fusion; a large enough error value is returned, meaning no fusion is allowed.

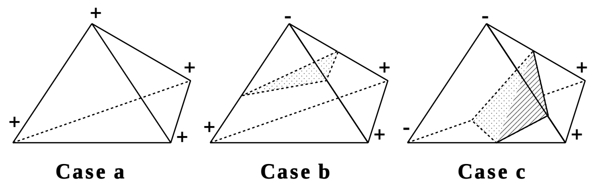


Figure 3: *Isosurface configurations within a tetrahedron*

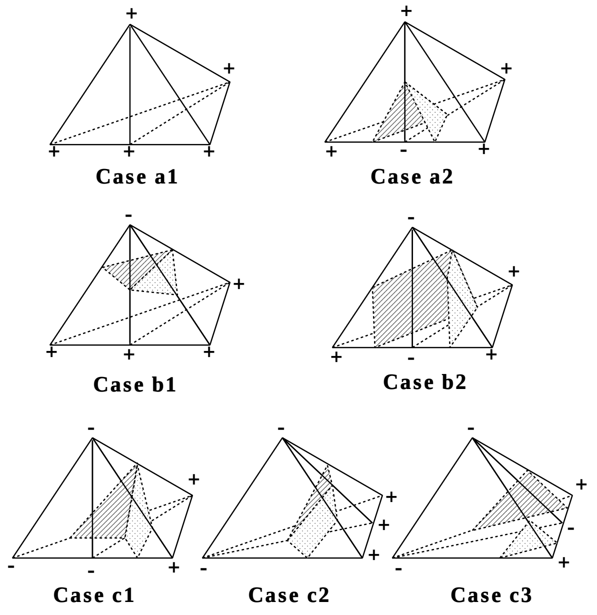


Figure 4: *Isosurface configurations after subdivision of the three cases in Fig. 3*

In other cases, we check the iso-point changes along four segments which connect the dividing point to four vertices of the tetrahedra. For a segment, if two endpoints hold different signs, a change of the position of the iso-point occurs when the dividing point density instead of the average density value of edge L endpoints is used. The distance of two iso-points can be calculated as the measure of the change. If two endpoints of a segment hold the same sign when using either the dividing point density or the average density, the distance is regarded as zero. Obviously, if all the segments have the distance value of zero, no approximation error happens. We take the maximum of all distance values as the approximation error. The alternative is to replace the physical distance with the projected distance, resulting in a viewpoint-dependent error evaluation. Our experiments show that taking the physical distance is efficient. It deserves to be noted that the above error evaluation is based on a prerequisite – the finest resolution tetrahedra mesh is most accurate, although its topology and geometry, in a sense, change relative to the original data structures, due to our subdivision mechanism.

4 Results

Our algorithm has been implemented in C with OpenGL. Results were obtained running on an SGI Power Challenge – R10000 CPU, 3GB main memory with Infinite Reality graphics. We applied the algorithm on two data sets: a voxelized box data and a medical CT head data set. Table 1 shows the time in seconds for the subdivision and fusion procedures for each data set. Although the two data sets have different sizes, both have been extended to the dimension of $129 \times 129 \times 129$. Since the subdivision process is independent of the actual data set values, they hold the same subdivision time. Based on this observation, the subdivision procedure can be realized as a preprocess for further simplifying the run-time calculation. On the other hand, the fusion procedure depends on the complexity of the data set, as most of the time is spent on error evaluation. The difference in fusion time in Table 1 is the result of a difference in data complexity.

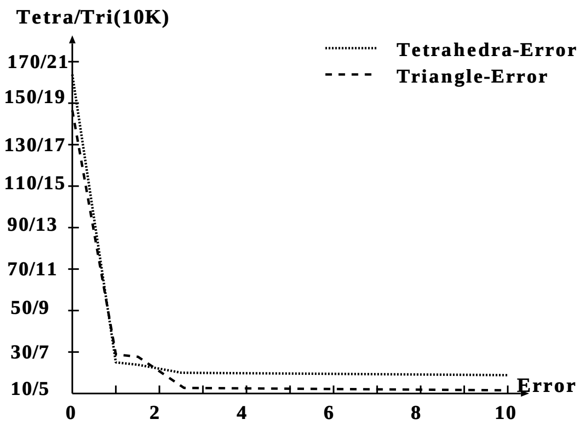
Table 1: Framework subdivision and fusion time (in seconds) for two data sets

Data Set	Resolution	Subdivision	Fusion
Box	$62 \times 127 \times 67$	90	19
CT Head	$128 \times 128 \times 113$	90	55

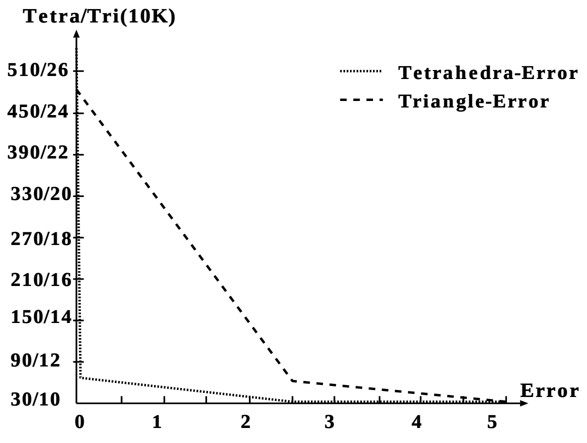
Fig. 5 shows the relationship between approximation error and the number of tetrahedra and triangles after the fusion for the Box and CT Head data sets. The triangles are generated by the Marching Tetrahedra technique shown in Fig. 3 with the same isovalue of 55.5 for both data sets.

When the approximation error is zero, that means no fusion occurs, so we get the original representation of the data set. As can be seen from Fig. 5, once the error is larger than zero, the number of tetrahedra approximating the original data set are quickly reduced. Due to the constraint of adjacent continuity, the fusion converges to a fixed model as the error increases to a certain value.

All the images are generated with an isovalue of 55.5. Figs. 6 - 9 correspond to errors: 0.0, 1.5, 2.5 and 4.0 (in voxel size), respectively. Figs. 10 (a) and (b) (see also color pictures) are with errors: 0.0 and 4.0, respectively. Fig. 11 (see also color pictures) is created using different errors: the left part of the head is specified with an error of zero, while the right part with an error of 4.0. The smooth change along the boundary of different levels-of-detail shows one of the advantages of our model. Observing Figs. 10 (a) and (b), the areas with fewer data changes in



(a)



(b)

Figure 5: *Approximation error vs. the number of tetrahedra and triangles: (a) the Box data set, (b) the CT Head data set*

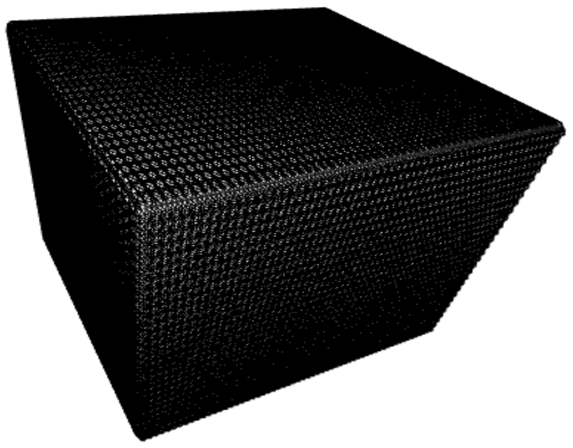


Figure 6: *The initial isosurface mesh consisting of 182,376 triangles extracted from 1,646,568 tetrahedra with error=0*

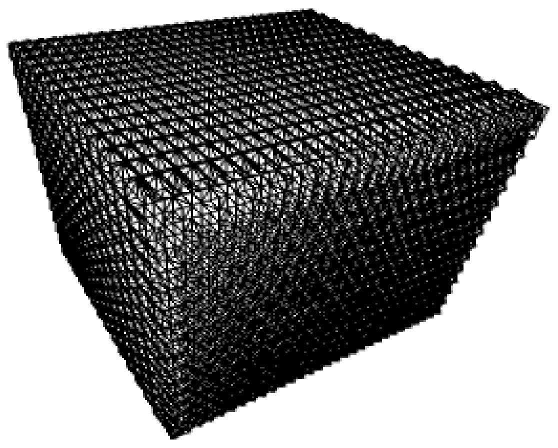


Figure 7: *Isosurface mesh consisting of 68,788 triangles extracted from 138,042 tetrahedra with error=1.5) (i.e., 91.6% fusion of tetrahedra)*

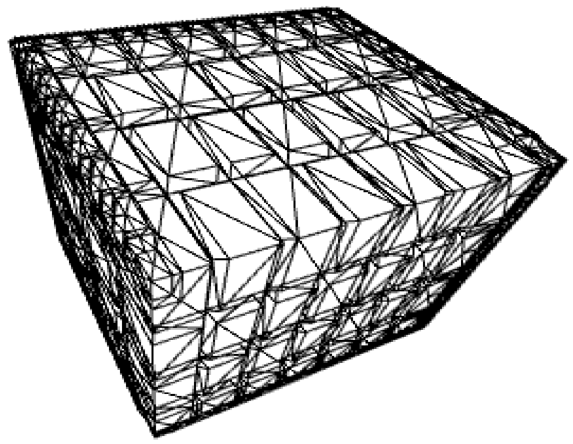


Figure 8: *Isosurface mesh consisting of 52,194 triangles extracted from 115,210 tetrahedra with error=2.5) (i.e., 93.0% fusion of tetrahedra)*

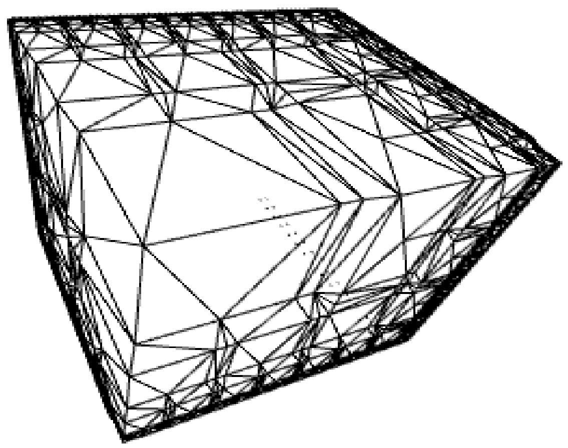
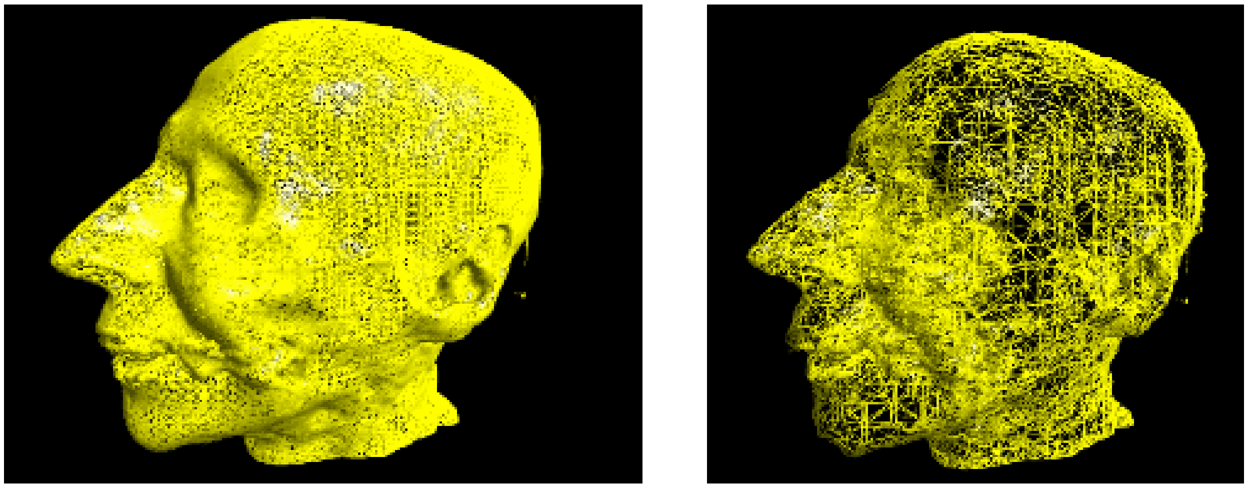
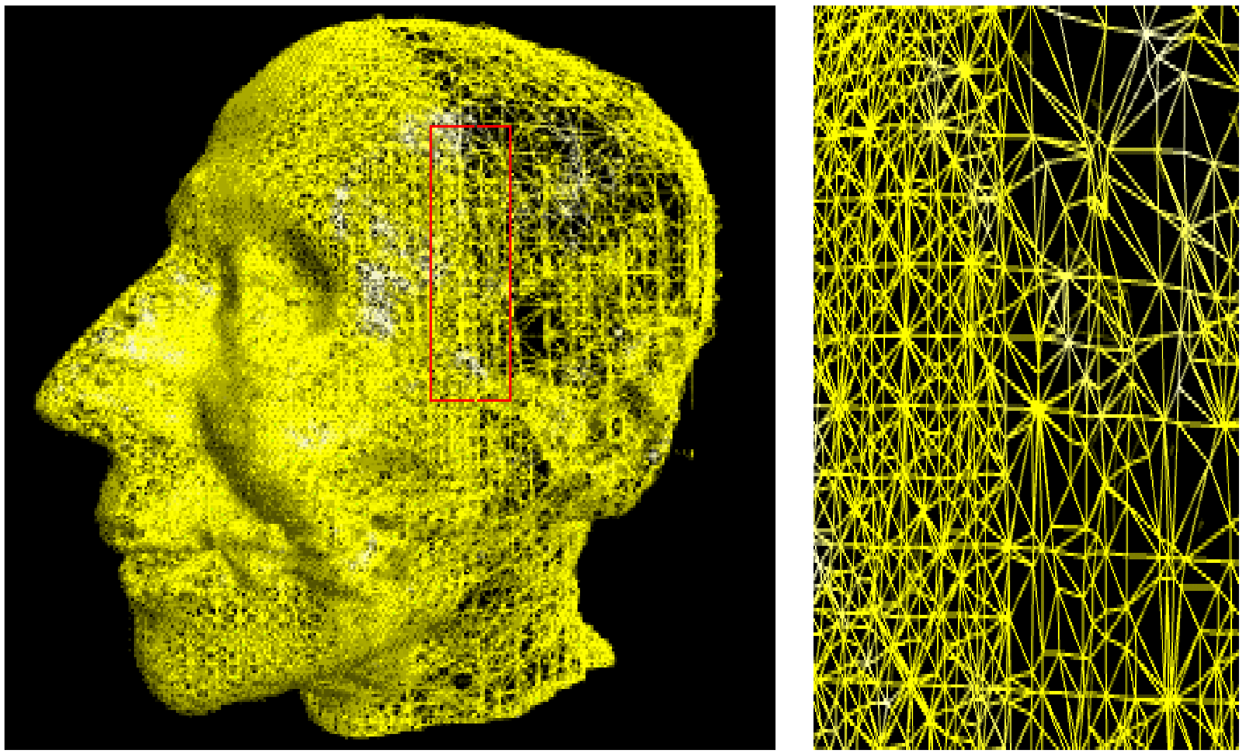


Figure 9: *Isosurface mesh consisting of 51,490 triangles extracted from 114,556 tetrahedra with error=4.0) (i.e., 93.04% fusion of tetrahedra)*



(a) (b)

Figure 10: *(a) Initial isosurface mesh consisting of 255,256 triangles extracted from 5,390,896 tetrahedra with error=0, (b) Isosurface mesh consisting of 89,300 triangles extracted from 304,468 tetrahedra with error=4.0 (i.e., 94.35% fusion of tetrahedra)*



(a) (b)

Figure 11: *(a) Isosurface mesh consisting of 184,228 triangles extracted from 2,891,024 tetrahedra with head left side error=0 and right side error=4.0 (i.e., 46.37% fusion of tetrahedra), (b) Zoom-in to the marked region in (a)*

Fig. 10 (a) are greatly simplified in Fig. 10 (b), while the regions of high frequency data remain mostly unchanged.

5 Conclusions

We have presented a volume-based multiresolution tetrahedral framework, the Multi-Tetra framework, and an error-based model. The framework is characterized by several features: compact representation, easy and fast query. Its most important property, differing from traditional algorithms, is that it provides an efficient mechanism for natural correspondence from one level to another, rather than the correspondence by interpolation of different levels in traditional multiresolution methods. Furthermore, the error-based model generated by the recursive fusion operation on the framework allows smooth, continuous changes between different levels-of-detail. This means we can display volume areas of interest by providing different approximation error thresholds within one image. For example, areas far away from the viewpoint, or of no interest, can be given loose error precision. As an application of the Multi-Tetra framework, isosurface extraction and error evaluation are discussed. Our method completely solves the crack problems which exist in traditional methods. In addition, the framework also provides possibilities for supporting typical volume rendering algorithms – direct projection, adaptive and progressive splatting – which are currently being investigated.

Our approach not only simplifies volume data, but also polygon meshes generated by traditional surface extraction methods. In other words, simplification of volume directly results in the simplification of generated polygon meshes.

6 Acknowledgements

This work has been supported by the National Science Foundation under grants CCR-9205047 and MIP-9527694.

References

[CDM+94] P. Cignoni, L. D. Floriani, C. Montoni, E. Puppo, and R. Scopigno. Multiresolution modeling and visualization of volume data based on simplicial complexes. *1994 Symposium on Volume Visualization*, ACM SIGGRAPH, pages 19–26, October 1994.

[HDD+93] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh optimization. *Computer Graphics (SIGGRAPH'93 Proceedings)*, volume 27, pages 19–26, August 1993.

[HEB94] D. J. Hebert. Symbolic local refinement of tetrahedral grids. *Journal of Symbolic Computation* , volume 17, pages 457–472, 1994.

[HHVW96] T. He, L. Hong, A. Varshney, and S. Wang. Controlled topology simplification. *IEEE Transactions on Visulization and Computer Graphics*, volume 2, No. 2, pages 171–184, 1996.

[LC87] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3D surface construction algorithm. *Computer Graphics (SIGGRAPH'87 Proceedings)*, volume 21, pages 163–169, July 1987.

[LKR96] P. Lindstrom, D. Koller, W. Ribarsky, L. F. Hughes, N. Faust and G. Turner. Real-time, continuous level of detail rendering of height fields. *Computer Graphics (SIGGRAPH'96 Proceedings)*, volume 30, pages 109–118, August 1996.

[LEV90] M. Levoy. Efficient ray tracing of volume data. *ACM Transactions on Graphics*, volume 9, No. 3, pages 245–261, July 1990.

[LH91] D. Laur and P. Hanrahan. Hierarchical splatting: A progressive refinement algorithm for volume rendering. *Computer Graphics (SIGGRAPH'91 Proceedings)*, volume 25, pages 285–288, July 1991.

[SFY96] R. Shekhar, E. Fayyad, R. Yagel and J. F. Cornhill. Octree-Based Decimation of Marching Cubes Surfaces. *IEEE Visualization'96*, pages 335–342, October 1996.

[ST90] P. Shirley and A. Tuchman. A polygonal approximation to direct scalar volume rendering. *Computer Graphics (San Diego Workshop on Volume Visualization)*, volume 24, No. 5, pages 63–70, November 1990.

[SZK95] R. Shu, C. Zhou, and M. S. Kankanhalli. Adaptive marching cubes. *The Visual Computer*, volume 11, No. 4, pages 202–217, 1995.

[SZL92] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of triangle meshes. *Computer Graphics (SIGGRAPH'92 Proceedings)*, volume 26, pages 65–70, July 1992.

[WG90] J. Wilhelms and A. V. Gelder. Octrees for faster isosurface generation extended abstract. *Computer Graphics (San Diego Workshop on Volume Visualization)*, volume 24, No. 5, pages 57–62, November 1990.

[WV94] J. Wilhelms and A. V. Gelder. Multi-dimensional trees for controlled volume rendering and compression. *1994 Symposium on Volume Visualization*, ACM SIGGRAPH, pages 27–34, October 1994.

[ZCT95] Y. Zhou, W. Chen, and Z. Tang. An elaborate ambiguity detection method for constructing isosurfaces within tetrahedral meshes. *Computers & Graphics*, volume 19, No. 3, pages 355–364, 1995.