Abstract

Recent advances in volume scanning techniques have made the task of acquiring volume data of 3-D objects easier and more accurate. Since the quantity of such acquired data is generally very large, a volume model capable of compressing data while maintaining a specified accuracy is required. The objective of this work is to construct a multi-resolution tetrahedral representation of input volume data. This representation adapts to local field properties while preserving their discontinuities. In this paper, we present an accuracy-based adaptive sampling technique to construct a multi-resolution model, which we call a tetrahedral adaptive grid, for hierarchical tetrahedrization of C^1 continuous volume data. We have developed a parallel algorithm of tetrahedral adaptive grid generation that recursively bisects tetrahedral grid elements by increasing the number of grid nodes, according to local field properties and such as orientation and curvature of isosurfaces, until the entire volume has been approximated within a specified level of view-invariant accuracy. We have also developed a parallel algorithm that detects and preserves both C^0 and C^1 discontinuities of field values, without the formation of cracks which normally occur during independent subdivision. Experimental results obtained using a PC cluster system demonstrate the validity and effectiveness of the proposed approach.

1. Introduction

Recent advances in volume scanning techniques have made the task of acquiring volume data of 3-D objects easier and more accurate. The problem of representing, reconstructing and visualizing such data has received rapidly growing attention in computer graphics [Kau01] [Nie01]. Since the quantity of such acquired data is generally very large, a volume model capable of compressing data while maintaining a specified accuracy is required. Thus, researchers have been faced with the problem of constructing accuracy-based volume models that can be used efficiently in various visual tasks. We address the problem of tetrahedral decomposition of input volume data. Our goal is to automatically construct a hierarchical tetrahedral representation of continuous smooth volume data. Our adaptive representation provides an accurate and efficient method for graphical rendering of volume data. Hierarchical volume models have the advantages of being simple to obtain from input data and of being able to approximate any volume at an arbitrary degree of accuracy. Such hierarchical models have been developed based on various criteria [CJ86] [ZCK97] [THJW99].

In the last decade, new tetrahedra-based approaches [Nie97] [JM.95] [Bey95] [NHR99] [GMPV02] [TG00] to constructing hierarchical models had been introduced since the simplest and most robust cells are tetrahedra in 3D. One major and inherent difficulty with hierarchical tetrahedrization techniques is that cracks may be formed in the volume approximation when each tetrahedron is subdivided independently, thus making parallel implementation rather difficult.

The crack problem has approached by several methods. Mauback proposed the method [JM.95], which has been used in [TG00], performs a local subdivision, rest, and then...
repairs the crack by propagating this split out through the mesh. The method of Bey [Bey95] uses a combination of two types of subdivision to avoid cracks and poorly-shaped tetrahedra. Nielson recently proposed a new approach [NHR99], which rather uses a Coons patch local model that covers over the crack. These new tetrahedra-based approaches had shown promise, however, many computational and analytical research issues, such as parallel implementation and view-invariant accuracy criteria for approximation of smooth volume data, etc., have been remained.

In this paper, we present an accuracy-based adaptive sampling technique to construct a multi-resolution model, we call a tetrahedral adaptive grid, for hierarchical tetrahedrization of $C^1$ continuous volume data. A tetrahedral adaptive grid is a straightforward extension to 3D of 2D adaptive mesh [TF93][Tan95], which was proposed for construction of a tetrahedral adaptive grid generation that recursively bisects tetrahedra elements by increasing the number of grid nodes according to local volume properties, such as orientation and curvatures of isosurfaces, until the entire volume has been approximated within a specified level of view-invariant accuracy.

We have also developed a parallel algorithm that detects and preserves both $C^0$ and $C^1$ discontinuities of field values, without the formation of cracks. This crack handling algorithm collects field value discontinuity information by recursively expanding the neighborhood of adjacent tetrahedra until the discontinuities are observed. The boundary reached by this recursive expansion defines the 3D region of reference for a grid element. This local definition of a bounded region of reference allows each grid element to be subdivided independently, and concurrently using multiple processors. Thus, the parallel computation of hierarchical tetrahedrization with no cracks is performed in bounded time and space.

Experimental results obtained using a PC cluster system demonstrate the validity and effectiveness of the proposed approach.

2. Tetrahedral Adaptive Grid Generation

We first give an overview of the tetrahedral adaptive grid algorithm, which is a straightforward extension to 3D of Adaptive mesh [TF93][Tan95], as shown in Fig. 1 and Fig. 2. The adaptive mesh was proposed for construction of an adaptive representation of free-formed smooth surfaces from input range images, according to view-invariant local surface properties such as surface orientation and curvatures. Next, we briefly describe a recursive algorithm of hierarchical binary subdivision, which was also proposed by Mauback [JM.95]. Then, we present the discontinuity-handling algorithm for parallel adaptive subdivision.

2.1. Overview of the algorithm

An input to the algorithm is i) a coarse regular hexahedral grid, and ii) a view-invariant accuracy criterion for isosurface approximation. A grid is given as a set of nodes uniformly located along x, y and z coordinates of the volume space, and consists of cubic cell elements of uniform vol-
ume \( \text{InitCubeSize}^3 \). The 3D region bounded by each cubic cell elements is initially approximated with a set of six root straight tetrahedra as shown in Fig. 3. Then, according to the local field properties observed at the bounding nodes, the root tetrahedra are recursively bisected independently in the region of rapid field variation. This subdivision process is repeated until the entire volume is approximated with the given accuracy criterion, \( \text{Acc\_Thresh} \).

The size of \( \text{InitCubeSize} \) is arbitrarily chosen. That is, the initialization of the grid is irrelevant to the accuracy of final tetrahedrization, because every discontinuity is eventually detected and preserved by the crack handling algorithm for discontinuities, which we present in the next section.

![Figure 3: Initial tetrahedrization of a cubic cell](image1)

### 2.2. Recursive Binary Subdivision

The algorithm for constructing the hierarchical representation is based on a stepwise refinement of an initially given grid. Given an accuracy criterion, binary subdivision of the parent tetrahedron \( T_p \) occurs when the accuracy criterion, \( \text{Acc\_Thresh} \), is violated for any six edges of \( T_p \). The subdivision of \( T_p \) into two left and right tetrahedron, \( T_l \) and \( T_r \), occurs by the creation of a new node, \( M \), the middle point of the base edge \( E(= \overrightarrow{P_0P_2}) \) of maximum length, followed by initialization of \( M \) with the local properties, i.e., the field value, orientation, curvatures of an isosurface containing \( M \). Then, the violation of \( \text{Acc\_Thresh} \) is recursively evaluated for each \( T_l \) and \( T_r \) independently.

#### 2.2.1. Tetrahedral Primitives

In recursive binary-subdivision, only three tetrahedral primitives including mirror-symmetry, TYPE-I, TYPE-II and TYPE-III, as show in Fig. 4. They are cyclically generated at level \( 3N \), \( 3N+1 \), \( 3N+2 \), respectively, as shown in Fig. 5. Fig. 6 shows recursive definitions of \( T_l \) and \( T_r \) using the parent node \( (P_0, P_1, P_2, P_3) \) and \( M \) for TYPE-I, TYPE-II, TYPE-III. As Fig. 5 shows three successive subdivision of a parent tetrahedron \( T_p \) of TYPE-I at level \( 3N \), generate the same type of great-ground children of TYPE-I cyclically at level \( 3(N+1) \) with each edge length and its volume decreased by \( 2 \) and by \( 8 \), respectively.

As Fig. 4 shows, face shapes of TYPE-I, TYPE-II, TYPE-III are either an isosceles triangle or a right triangle. With the ratio of maximum to minimum edge length \( \sqrt{3} \) at TYPE-I, \( 2\sqrt{2}/\sqrt{3} \) at TYPE-II, \( 2 \) at TYPE-III, respectively. This binary tetrahedrization using the middle points thus sufficiently satisfies the equi-angular requirement. Another advantage of the binary tetrahedrization is that it provides a more continuous level of volume approximation, because a tree with fewer descendants has more levels of approximation for a given range of volume variation.
2.2.2. Orientation of Initial Tetrahedrization

Our initial tetrahedrization of a cube, which is equivalent to the CFK tetrahedrization, has following features. First, there is no alternating tetrahedrization required, since, all six faces of a InitCubicCell have diagonally consistent edges with adjacent faces (Fig. 3(a)). Second, the choice of orientations in initial tetrahedrization does not effect to the final tetrahedrization.

As Fig. 21 shows, there are four distinct orientations of the initial tetrahedrization of a cube, patterns (A), (B), (C) and (D). These orientations depends on the direction of diagonal edges in a cubic cell, shared by all six-root tetrahedra. However, as Fig. 21 shows, four diagonally different initial tetrahedrizations reach to the same tetrahedrization after 1 cycle of three successive subdivisions. Fig. 22 shows the adjacency of all four patterns in a InitCubicCell. Each pair of the same patterned cells of size 1/8 are diagonally positioned and all four pairs meet at the center of the InitCubicCell. Thus, all orientation effects are cancelled out at finer resolution.

2.2.3. Binary Subdivision Algorithm

Above steps for recursive binary subdivision of tetrahedra are summarized in the following pseudo code.

Procedure Divide_Tetrahedron(T_p, Acc_Thresh)
(* T_p : a binary region tree *)
(* Acc_Thresh : homogeneity criteria *)

begin

Step 1: (*Collect subdivision requests from neighbors for crack handling discontinuities *)

If NeighborRequire_for_Subdivision(T_p, Acc_Thresh)
then require for subdivision of T_p;

Step 2: (* Neighbors require for subdivision of T_p*)
Divide a parent tetrahedron T_p into T_l and T_r and process them independently

Step 2.1: (*Initialize T_l and T_r using the parent nodes of T_p and a middle point on the base edge *)
Divide_Tetrahedron(T_l, Acc_Thresh);
Divide_Tetrahedron(T_r, Acc_Thresh);

end

At each recursion, the volume of every tetrahedron decreases by 2, therefore the upper bound of recursion n max, assuming that minimum cell size is 1, is given as,

$$n_{\text{max}} \leq \log_2(\text{InitCubeSize}^3) \quad (1)$$

where InitCubeSize is the edge length of an initial grid element, i.e. InitCubicCell.

2.3. Crack Handling for Discontinuities

The major problem with adaptive subdivision techniques is that cracks, i.e., discontinuities, may arise if each tetrahedron is subdivided independently, as shown in Fig. 7.

When there is a large field variation near the initial grid element, a crack may be formed along the boundary between the grid elements. This crack is caused by the unilateral subdivision of a grid element on one side of the large field variation. In order to avoid cracks between adjacent tetrahedra, we have developed an algorithm that collects subdivision information from neighbors by recursively expanding the 3D region of reference until a sudden field change is observed.

In the binary subdivision, every tetrahedron is subdivided only at the middle point of its base edge, the insertion of the middle point in uences the subdivision of adjacent tetrahedra only along the base edge. For the base edge E of T_p,
we associate a 3D region of influence $RI(E)$ bounded by a group of tetrahedra sharing $E$, which is called a diamond in [GMPV02].

If $E_3$ is parallel to one of the X, Y and Z coordinate axes, then $RI(E_3)$ consists of eight tetrahedron from four adjacent cubic cells sharing $E_3$, as shown in Fig 8(c). Ten vertices $P_i (0 \leq i \leq 9)$ of $RI(E_3)$ can be computed from Eq. 4.

\[ \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_7 \\ P_8 \\ P_9 \end{bmatrix} = \begin{bmatrix} (P_1 + P_3)/2 \\ P_1 \\ P_2 \\ P_3 \\ (P_1 + P_3)/2 \\ P_1 \\ P_2 \\ (P_1 + P_3)/2 \\ P_1 \\ (P_1 + P_3)/2 \end{bmatrix} + L/2 \times \begin{bmatrix} -I_x \\ 0 \\ -I_x \\ -I_x \\ 0 \\ I_x \\ -I_x \\ 0 \\ I_x \\ 0 \end{bmatrix} \]

(4)

Then, we add another constraint on the subdivision of $T_p$ in $RI(E)$. That is, ”if any other one of tetrahedra in $RI(E)$ is bisected at $M$, then also bisect $T_p$ at $M$“.

Thus, if the given accuracy criteria is violated for any edge of any one of tetrahedra in $RI(E)$, then $T_p$ is subdivided into two tetrahedra $T_l$ and $T_r$ at $M$.

For each grid element $Cb(i,j,k)$, we process its root tetrahedra $Rt[ij]:0, ..., 5$, independently. For each tetrahedron $T_p$, we rst evaluate the accuracy achieved along its base edge $E$. If the accuracy along $E$ has not reached the given accuracy criteria, we immediately decide the subdivision of $T_p$ without further examining the neighbors. The reasons are:

1) From the constraint on $E$ with its region of influence $RI(E)$, all tetrahedra in $RI(E)$ are also subdivided at $M$, thus there are no cracks in $RI(E)$, even if eld discontinuities are observed inside of one of tetrahedron in $RI(E)$, and

2) The insertion of $M$ in oe only the subdivision within $RI(E)$, and does not in oe the subdivision of neighbors outside $RI(E)$. Otherwise, we postpone the decision until we are able to compare it from subdivision information collected from the neighbors of $Cb(i,j,k)$.

Next, we associate each of base edges of $T_l$ and $T_r$ with its region of influence $RI(E_i) (0 \leq i \leq 2)$, respectively, then evaluate whether the accuracy in $RI(E_i)$ has been reached at the given threshold. This evaluation process leads to the recursive division of the regions of influence for each base edge of $T_l$ and $T_r$ at successive levels.

If the accuracy reached in $RI(E(n))$ along the base edge $E(n)$ de ned at the nth level, does not satisfy the threshold, the request for a subdivision arises at level $n$, then requests for subdivision of all ancestor tetrahedra will be propagated to the root tetrahedra of $Cb(i,j,k)$. Otherwise, the recursive division of $RI(E(n))$ followed by the expansion of $RR(Cb(i,j,k))$ continues until the size of $RI(E(n))$ becomes 1(a minimum size cell). In this case, eld values
in the neighboring region bounded by $RR(Cb(i, j, k))$ is constant, so the region bounded by $Cb(i, j, k)$ can be sufficiently approximated with six root tetrahedra.

The volume of $RI(E)$ decreases by a factor of $2^3$ after three successive recursions, causing the base edge length reduction by a factor 2. Thus, a region of reference $RR(Cb(i, j, k))$, will be recursively expanded. Fig. 23 and Fig. 24 show neighboring regions considered in the subdivision of $Cb(i, j, k)$. Fig. 24 shows the 3D region of reference $RR(Cb(i, j, k))$ expanded by recursive definition of $RI(E(n))$ at each level. Fig. 23 shows the 2D projection of $RI(E)$ after three successive subdivisions of 6 root tetrahedra in a $Cb(i, j, k)$. This leads to the recursive expansion of the overall regions referenced for $Cb(i, j, k)$, denoted by $RR(Cb(i, j, k))$.

The projection of distance $d$, on either the X-Y, Y-Z and Z-X coordinate plane, from the boundary of $Cb(i, j, k)$ to the farthest region $RI(E)$ after the $i$th recursion is given as,

$$d = \sum_{i=1}^{k} \left( \frac{InitCubeSize}{(2^i)} \right) < InitCubeSize$$  \hspace{1cm} (5)

where $k = \lfloor n/3 \rfloor$ and $InitCubeSize$ is the edge length of an initial grid element.

Eq. 5 indicates that the upper bound $d_{max}$ is limited by $InitCubeSize$, as shown in Fig. 9. Thus, we can define the bounded region of reference $RR(Cb(i, j, k))$ of volume $8^{1/3}(1 + 6 + 8 \times (1/6)) \times (InitCubeSize)^3$ for each initial grid element $Cb(i, j, k)$, as shown in Fig. 24(f).

The upper bound of recursion $n_{max}$ is also given as,

$$n_{max} \leq \log_2 \left( \frac{1}{6} \right) (InitCubeSize)^3$$  \hspace{1cm} (6)

The local definition of a bounded region of reference allows each grid element to be subdivided independently. This enables parallel computations of tetrahedrization, with no cracks, in bounded time of $O(n)$ enables parallel computations of tetrahedrization, with no cracks, in bounded space of $O(\log n)$. Thus, a region of reference $E(n)$ can be subdivided into $E(n+1)$, denoted by $RR(Cb(i, j, k))$. Fig. 24 shows neighboring regions considered in the subdivision of $E(n)$.

**Procedure Neighbor_Require_for_Subdivision($T_p$,Acc_Thresh)**

(* $T_p$ : a binary region tree *)

(* Acc_Thresh: homogeneity criteria *)

begin

Step 0: If $CufAcc$ along BaseEdge $E$ of $T_p$ does not reach $Acc_Thresh$ then return a request for subdivision of $T_p$.

Step 1: Estimate the size of $T_p$

If $T_p$ is the smallest Tetrahedra then return $NeedSubdivision$

Step 2:Associate $E$ with $RI(E)$ of a group of tetrahedra $[T_p]$ sharing $E$.

Step 3: (* Recursive Expansion of $RR(Cb(i, j, k))$ *)

(* Evaluate $Acc_Thresh$ for $T_l$ and $T_r$ of each of $T_p$ sharing $E$ independently *)

for each $T_p$ in $RI(E)$

Step 3.1:Divided $T_p$ into $T_l$ and $T_r$ at a middle point of $E$.

Step 3.2: (* Recursively evaluate $Acc_Thresh$ for $T_l$ and $T_r$)

**CurDepth++**

Neighbor_Require_for_Subdivision($T_l$,Acc_Thresh)

**Neighbor_Require_for_Subdivision($T_r$,Acc_Thresh)**

Step 4: If any neighbor requires for subdivision then return for the subdivision of $T_p$.

Step 5: Return ($NeedDivision$).

end

**3. Accuracy Criterion**

Our accuracy criterion is given as the ratio of a curve of field value changes to its linear approximation.

We first consider field value changes along every edge $E_i$ of a tetrahedron in the following 2D space $S$, where the x axis is along $E_i$ and field values along $E_i$ are represented as heights in the direction of the y axis, as shown in Fig. 25. Field values changes of $C^1$ continuous volume data along $E_i$ will draws a curve rather than a line. Linear interpolation of field values inside a tetrahedron, which is conventionally used in many method [GMPV92] [TG00] is equivalent to draw a line between $P_l$ and $P_r$ in $S$.

Let $P_l$ and $P_r$ be end points of $E_i$ in $S$ and let $D_1$ be a 3D distance between $P_l$ and $P_r$ and $R_i$ be the arc length of a curve of field values along $E_i$. Such curves can be obtained with conventional curve interpolation techniques, e.g., B-spline interpolation. In our implementation, such curves are estimated using the both end nodes illustrated in Fig. 25.

A curve of field values between two points $P_l$ and $P_r$ is estimated as three B-spline segments, painted pink, green and red, using B-spline interpolation. The B-spline control points colored purple and yellow are generated according to osculating circles defined at $P_l$ and $P_r$, which are computed from curvatures of iso-surfaces estimated at $P_l$ and $P_r$, and an angle $\Theta$ specified at $Acc_Thr$. The centers $O_l$ and $O_r$ of the osculating circles are determined from the normals,
\[ \vec{n}_l = \frac{\vec{O}_l \vec{P}_l}{\| \vec{O}_l \vec{P}_l \|}, \quad \vec{n}_r = \frac{\vec{O}_r \vec{P}_r}{\| \vec{O}_r \vec{P}_r \|}, \quad \text{and the curvature,} \\
\[ k_1 = \frac{1}{\| \vec{O}_1 \vec{P}_1 \|}, \quad k_2 = \frac{1}{\| \vec{O}_r \vec{P}_r \|}, \quad \text{at} \ P_l \ \text{and} \ P_r \ \text{in the direction} \ \vec{P}_1 \vec{P}_2. \]

Our accuracy criterion is given as the ratio of a curve of field value changes to its linear approximation, and is applied to every edge \(E_i\) of all tetrahedra, That is,

\[ Acc_{\text{Thresh}} = \frac{m}{\pi} \sin \left( \frac{\pi}{m} \right) \]

where \(m\) specifies the accuracy which is equivalent to the \(m\)-hedron approximation of a unit circle, illustrated in Fig. 10. This criterion is equivalent to constraining angles between gradient vectors of adjacent tetrahedra, which also constrains an angle between adjacent triangles(patches) of isosurface approximation.

With the above accuracy criterion, the condition on subdivision of a given tetrahedra \(T_p\) is stated as follows:

\[ \forall i \quad \frac{R_i}{D_i} \leq Acc_{\text{Thresh}}, (0 \leq i \leq 5). \]

**Figure 10:** A \(m\)-hedron approximation of a unit circle, where \(R\) is an arc length and \(D\) is a chord length. \(Acc_{\text{Thresh}}\) is given as an \(m\)-hedron approximation of a unit circle, \(Acc_{\text{Thresh}} = \frac{\text{arc length}}{\text{chord length}} = \frac{\pi}{m} \sin(\frac{\pi}{m})\), which constrains an angle \(\theta\) between gradient vectors of adjacent tetrahedra as \(\theta \leq \pi - \frac{2\pi}{m}\), which also constrains an angle between adjacent triangles(patches) of isosurface approximation.

### 4. Experiments

We have implemented the tetrahedral adaptive grid algorithm in C++ programming language, VTK(the Visualization ToolKit) as graphic library and MPI library to run it on a PC cluster system. Our PC cluster system consists of 16 host computers, one CPU for job control, one CPU for memory control and 14 CPUs for parallel computation, with the Score Cluster System Software. Every host computer consists of dual Xeon 2.8 GHz CPU, 2GB of main memory and Myrinet 2000 NIC.

We applied the algorithm to two kinds of volume data. The one is Lobster of size 321 × 321 × 33. The another one is Human Foot of size 161 × 321 × 129, which is reconstructed from CT les of a female cadaver, as collected for the National Library of Medicine’s Visible Human Program. The initial cube size of Lobster and Human Foot is 32. Hence, the max decomposition level becomes 15. The accuracy criterion was specified as, \(Acc_{\text{Thresh}} = (20\sin \frac{1}{20})\), i.e., the 20HII-hedron approximation of a unit circle.

Second, we tested on Human foot. The Input data is shown in Fig. 15. And the result that applied the algorithm is shown in Fig. 16. Fig. 17 and Fig 18 show the results of hierarchical tetrahedrization projected on X-Y, Y-Z and Z-X planes at level 6, 9 and 12. Table. 1 shows results of compression rate at level 3, 6, 9, 12 and 15, respectively.

**Table 1:** Compression rate (lobster)

<table>
<thead>
<tr>
<th>Level</th>
<th>time(s)</th>
<th>Non-Adaptive</th>
<th>Adaptive</th>
<th>Compression rate(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0.044664</td>
<td>1,323</td>
<td>416</td>
<td>31.4</td>
</tr>
<tr>
<td>6</td>
<td>0.084752</td>
<td>8,405</td>
<td>1,983</td>
<td>23.60</td>
</tr>
<tr>
<td>9</td>
<td>0.394752</td>
<td>59,049</td>
<td>11,937</td>
<td>20.200</td>
</tr>
<tr>
<td>12</td>
<td>3.08826</td>
<td>440,657</td>
<td>42,475</td>
<td>9.3902</td>
</tr>
<tr>
<td>15</td>
<td>18.7092</td>
<td>3,400,353</td>
<td>298,892</td>
<td>8.7900</td>
</tr>
</tbody>
</table>

**Figure 11:** Input Data -Lobster-
Computation time V.S. the number of CPUs for constructing the adaptive grids of 6 and 9, respectively.

5. Conclusions

We have proposed a parallel algorithm of tetrahedral adaptive grid generation that automatically generate hierarchical tetrahedral representation of input volume data. The representation can be used as an accurate and efficient volume model. Such hierarchical tetrahedrization has the advantages of being intrinsic to the volume and of satisfying the arbitrarily specified absolute accuracy. We have also proposed a recursive search algorithm that collects subdivision information from neighbors to avoid cracks in the volume approximation. Then, from the boundary of the neighbors referred to, we defined a region of reference for each grid element. This local definition of bounded neighbors for each grid element allows each grid element to be subdivided independently. This enables parallel computation of hierarchical tetrahedrization with no cracks in bounded time and space. The method is general and can be applied to adaptive data compression of any volumetric data.

References


[CJ86] CHIEN C., J.K. AGGARWAL: Volume sur-


