EUROGRAPHICS Workshop on Multimedia (2004)
N. Correia, J. Jorge, T. Chambel, Z. Pan (Editors)

Tetrahedral Adaptive Grid for Parallel Hierarchical Tetrahedrization

Yasufumi Takama, Akinori Kimura† and Hiromi T.Tanaka

Computer Vision Laboratory, Graduate School of Information Science and Engineering,
Ritsumeikan University.
Noji-Higashi 1chome 1-1, Kusatsu, Shiga, 525-8577 Japan.
e-mail : {yasufumi, hiromi}@cv.ci.ritsumei.ac.jp, A.Kimura@cg.is.ritsumei.ac.jp

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, 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 gird 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 speci ed accuracy is required. Thus, researchers have been faced with the problem of constructing accuracy-based volume models that can be used ef ciently in various visual tasks. We address the problem of tetrahedra decomposition of input volume data. Our goal is to automatically construct a hierarchical tetrahedra representation of continuous smooth volume data. Our adaptive representation provides an accurate and ef cient 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 dif culty 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 dif cult.

The crack problem has approached by several methods. Mauback proposed the method [JM.95], which has been used in [TG00], performs a local subdivision rst and then

† CREST, Japan Science and Technology Agency

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

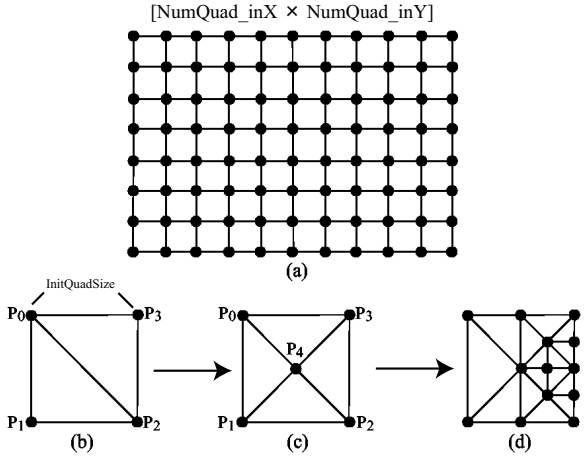
In this paper, we present an accuracy-based adaptive sampling technique to construct a multi-resolution model, we call a tetrahedral adaptive grid, for hierarchical tetrahedrization of C^1 continuous volume data. A tetrahedral adaptive grid is a straightforward extension to 3D of 2D adaptive mesh [TF93][Tan95], 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 speci ed level of view-invariant accuracy.

We have also developed a parallel algorithm that detects and preserves both C^0 and C^1 discontinuities of eld values, without the formation of cracks. This crack handling algorithm collects eld value discontinuity information by recursively expanding the neighborhood of adjacent tetrahedra until the discontinuities are observed. The boundary reached by this recursive expansion de nes the 3D region of reference for a grid element. This local de nition 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.

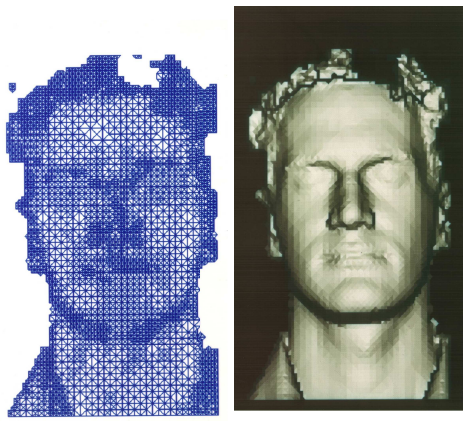
2. Tetrahedral Adaptive Grid Generation

We rst 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 brie y 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.



Step1: mesh initialization. (a) Black circles represent nodes; lines represent connections aligned with mutually orthogonal (x,y) coordinate lines of viewing plane. Step2: initial triangulation. (b) At each node, the surface shape is recovered with depth z, normal $\mathbf{n}$, principal curvatures k_1,k_2, and their directions $\vec{e}_1$, $\vec{e}_2$. The region bounded by each quadrangle is initially approximated by two triangular patches of four neighboring nodes. Step3:recursive subdivision. (c,d) According to the curvatures at both ends, the nodes are increased along the lines to approximate the regions of high curvature with ner triangular patches.

Figure 1: *Adaptive Mesh Generation [Tan95]*



(a)Adaptive mesh (b)Flat-shaded rendering of (a)

Figure 2: *Adaptive mesh model of a human face*

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 $InitCubeSize^3$. 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 eld properties observed at the bounding nodes, the root tetrahedra are recursively bisected independently in the region of rapid eld variation. This subdivision process is repeated until the entire volume is approximated with the given accuracy criterion, *Acc_Thresh*.

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

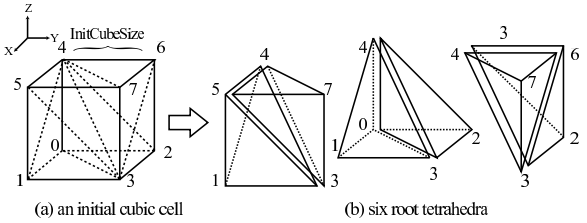


Figure 3: *Initial tetrahedrization of a cubic cell*

2.2. Recursive Binary Subdivision

The algorithm for constructing the hierarchical representation is based on a stepwise re nement of an initially given grid. Given an accuracy criterion, binary subdivision of the parent tetrahedron T_p occurs when the accuracy criterion, *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 $E(=\overrightarrow{P_1P_2})$ of maximum length, followed by initialization of M with the local properties, i.e., the eld value, orientation, curvatures of an isosurface containing M. Then, the violation of *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 de nitions 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

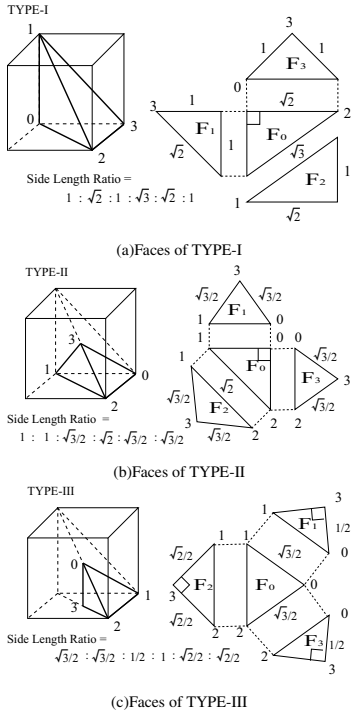


Figure 4: *Three tetrahedral primitives*

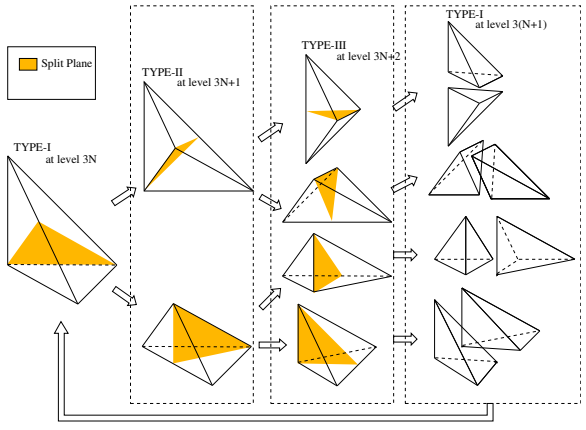


Figure 5: *Cyclic subdivision of a tetrahedron into TYPE-I, TYPE-II and TYPE-III primitives*

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 suf - ciently satis es 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.

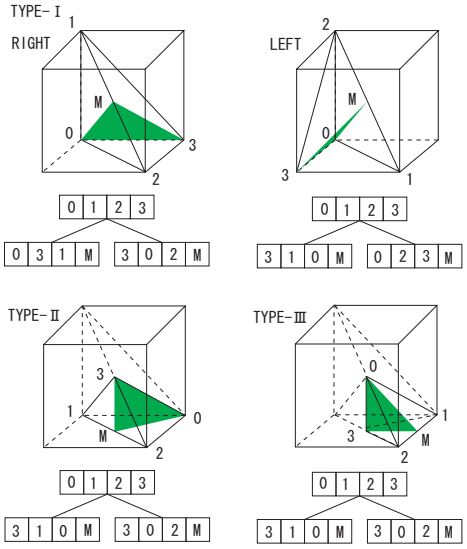


Figure 6: *Recursive definition of T_l and T_r for three primitives*

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 nal 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 InitCubic Cell. Thus, all orientation effects are cancelled out at ner 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 Neighbor_Require_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 *)*
Step 2.2: (Recursive Subdivision of T_l and T_r *)*
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,

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

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.

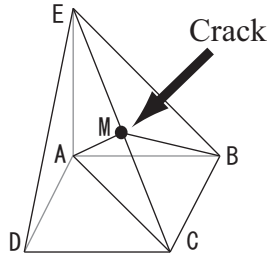


Figure 7: *The crack problem. The tetrahedron $T_{A,B,C,E}$ consists of four vertices A, B, C and E. The tetrahedron has been subdivided into two tetrahedra $T_{A,M,B,C}$ and $T_{A,M,B,E}$ at a middle point M. However, the tetrahedron $T_{A,C,D,E}$ has not been subdivided. This problem causes making holes when generating isosurfaces of input volume data.*

When there is a large eld 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 eld 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 eld 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 in uence $RI(E)$ bounded by a group of tetrahedra sharing E, which is called a diamond in [GMPV02].

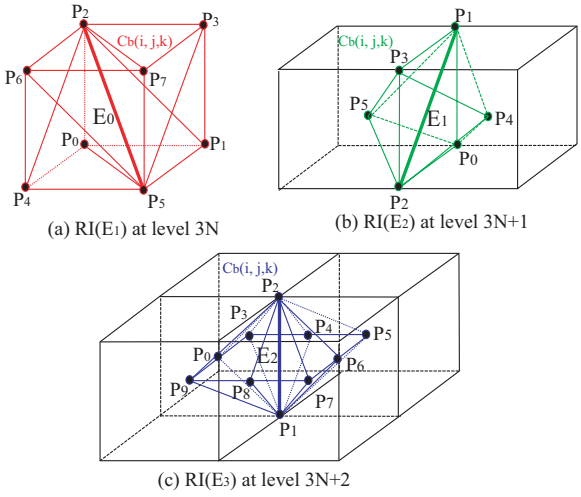


Figure 8: *Three types of 3D region of influence : $RI(E_i)$*

Let $I_x = (1,0,0)$, $I_y = (0,1,0)$ and $I_z(0,0,1)$ be x, y and z element of an unit vector I, respectively. And let L be an edge length of a cubic cell at current level n(=3n).

If E_1 is a diagonal edge inside a cubic cell at level n(=3N) then *RI(E_1)* is the cubic cell itself consisting of six tetrahedra sharing E_1, as shown in Fig 8(a). Eight vertices P_i $|(0 \leq i \leq 7)$ of $RI(E_1)$ can be computed from Eq. 2.

$$\begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_7 \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \\ P_1 \\ P_1 \\ P_2 \\ P_2 \\ P_1 \\ P_2 \end{bmatrix} + L \times \begin{bmatrix} -I_y \\ -I_z \\ 0 \\ I_x \\ -I_x \\ 0 \\ I_z \\ I_y \end{bmatrix} \quad (2)$$

If E_2 is a diagonal edge on a bounding face shared by adjacent cubic cell, then *RI(E_2)* consists of 4 tetrahedron sharing E_2, two from its own cubic cell and other two from the adjacent cell, as shown in Fig 8(b). Six vertices P_i $|(0 \leq i \leq 5)$ of $RI(E_2)$ can be computed from Eq. 3.

$$\begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \end{bmatrix} = \begin{bmatrix} P_1 \\ P_1 \\ P_2 \\ P_1 \\ (P_1+P_2)/2 \\ (P_1+P_2)/2 \end{bmatrix} + L \times \begin{bmatrix} -I_y \\ 0 \\ 0 \\ I_z \\ I_x/2 \\ -I_x/2 \end{bmatrix} \quad (3)$$

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.

$$\begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_7 \\ P_8 \\ P_9 \end{bmatrix} = \begin{bmatrix} (P_1+P_2)/2 \\ P_1 \\ P_2 \\ (P_1+P_2)/2 \\ (P_1+P_2)/2 \\ (P_1+P_2)/2 \\ (P_1+P_2)/2 \\ (P_1+P_2)/2 \\ (P_1+P_2)/2 \\ (P_1+P_2)/2 \end{bmatrix} + L/2 \times \begin{bmatrix} -I_x \\ 0 \\ 0 \\ -I_x - I_z \\ -I_z \\ I_x - I_z \\ -I_x \\ I_x + I_z \\ +I_z \\ -I_x + I_z \end{bmatrix} \quad (4)$$

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[i]i: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 in uence $RI(E)$, 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 uences only the subdivision within $RI(E)$, and does not in uence the subdivision of neighbors outside $RI(E)$. Otherwise, we postpone the decision until we are able to con rm 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 in uence $RI(E_i)$ $|(0 \leq i \leq 2)$, respectively, then evaluate whether the accuracy in $RI(E_i)$ has been reached at the given threshold. This evaluation process leads to the recursive de nition of the regions of in uence 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 s subdivision of *all* ancestor tetrahedra will be propagated to the root tetrahedra of $Cb(i,j,k)$. Otherwise, the recursive de nition 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 suf ciently 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 de nition 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 ith expansions is given as,

$$d = \sum_{i=1}^{k} (InitCubeSize/(2^i)) < InitCubeSize \qquad (5)$$

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 de ne the *bounded* region of reference $RR(Cb(i,j,k))$ of volume $8\frac{1}{3}(= 1+6+8*(1/6))*(InitCubeSize)^3$ 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,

$$n_{max} \leq \log_2 \frac{1}{6}(InitCubeSize)^3 \qquad (6)$$

The local de nition 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 $O(\log_2(InitCubeSize)^3)$, and in bounded space of $8\frac{1}{3}*(InitCubeSize)^3$.

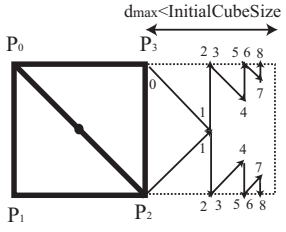


Figure 9: *Upper bound d_{max} ($\leq$ InitCubeSize)*

These steps are summarized in the following pseudo codes.

Procedure Neighbor_Require_for_Subdivision(T_p,Acc_Thresh)

```
(* Tp : a binary region tree *)
(* Acc_Thresh: homogeneity criteria *)
begin
  Step 0: If CurrAcc along BaseEdge E of Tp does not
          reach Acc_Thresh then
          return a request for subdivision of Tp.
  Step 1: Estimate the size of Tp
          If Tp is the smallest Tetrahedra then
          return !NeedSubdivision
  Step 2:Associate E with RI(E) of a group of
          tetrahedra {Tpi} sharing E.
  Step 3:(* Recursive Expansion of RR(Cb(i,j,k)) *)
          (* Evaluate Acc_Thresh for Tl and Tr of each of
                        Tpi ∈ RI(E) independently *)
          for each Tpi ∈ RI(E)
    Step 3.1:Divided Tpi into Tli and Tri at a middle point of E.
    Step 3.2:(* Recursively evaluate Acc_Thresh for Tli and Tri
        CurrDepth++
Neighbor_Require_for_Subdivision(Tli,Acc_Thresh)
Neighbor_Require_for_Subdivision(Tri,Acc_Thresh)
  Step 4:If any neighbor requires for subdivision
          then require for the subdivision of Tp.
  Step 5:Return (!NeedDivision).
end
```

3. Accuracy Criterion

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

We rst consider eld 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 eld 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 eld values inside a tetrahedron, which is conventionally used in many method [GMPV02] [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_i be a 3D distance between P_l and P_r and R_i be the arc length of a curve of eld 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 eld 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 de ned at P_l and P_r, which are computed from curvatures of iso-surfaces estimated at P_l and P_r, and an angle θ speci ed at *Acc_threth*. The centers O_l and O_r of the osculating circles are determined from the normals,

$\overrightarrow{n_l} = \overrightarrow{O_lP_l}/\|\overrightarrow{O_lP_l}\|$, $\overrightarrow{n_r} = \overrightarrow{O_rP_r}/\|\overrightarrow{O_rP_r}\|$, and the curvature, $k_1 = 1/\|\overrightarrow{O_lP_l}\|$, $k_2 = 1/\|\overrightarrow{O_rP_r}\|$, at P_l and P_r in the direction of $\overrightarrow{P_1P_2}$.

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

$$Acc\_Thresh = \frac{m}{\pi}\sin(\frac{\pi}{m}) \tag{7}$$

where m speci es 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:

$$\forall i \ \frac{R_i}{D_i} \leq Acc\_Thresh, (0 \leq i \leq 5). \tag{8}$$

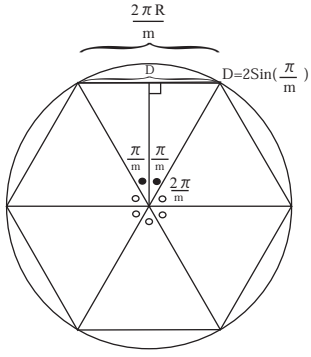


Figure 10: *A m-hedron approximation of a unit circle, where R is an arc length and D is a chord length. Acc_thresh is given as an m-hedron approximation of a unit circle, Acc_thresh = (arc length)/(chord length) = π / m sin(π/m), which constrains an angle θ between gradient vectors of adjacent tetrahedra as θ ≤ π − 2π/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 $321 \times 321 \times 33$. The another one is Human Foot of size $161 \times 321 \times 129$, 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 speci ed as, $Acc\_Thresh=(20sin\frac{1}{20})$, i.e., the 20Π-hedron approximation of a unit circle.

We rst tested the algorithm on Lobster data (Fig. 11). Fig. 12, Fig. 13 and Fig 14 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)*

Level	time(s)	Non-Adaptive	Adaptive	Compression rate(%)
3	0.044664	1,323	416	31.4
6	0.084752	8,405	1,983	23.60
9	0.394752	59,049	11,937	20.200
12	3.08826	440,657	42,475	9.3902
15	18.7092	3,400,353	298,892	8.79000

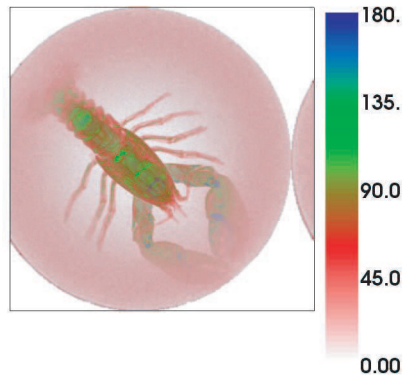


Figure 11: *Input Data -Lobster-*

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 plane at level 9 and 12. The compression rates of each level are 15.2063% and 10.3186%, respectively.

Performance of algorithm is evaluated on the PC cluster system by increasing the number of CPUs. Fig. 19 and Fig. 20 show evaluation results on Human foot comparing

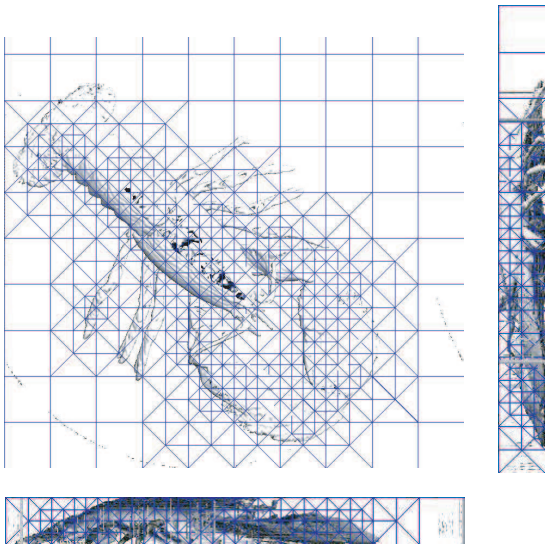


Figure 12: *the result of Lobster at Level 6*

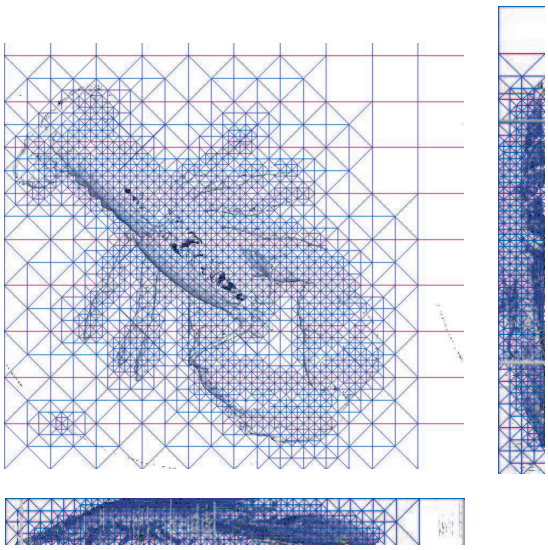


Figure 13: *the result of Lobster at Level 9*

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 ef cient volume model. Such hierarchical tetrahedrization has the advantages of being intrinsic to the volume and of satisfying the arbitrarily speci ed absolute accuracy. We have also proposed

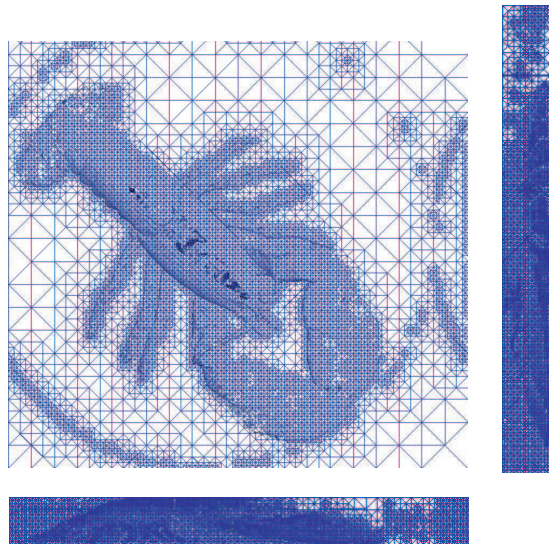


Figure 14: *the result of Lobster at Level 12*

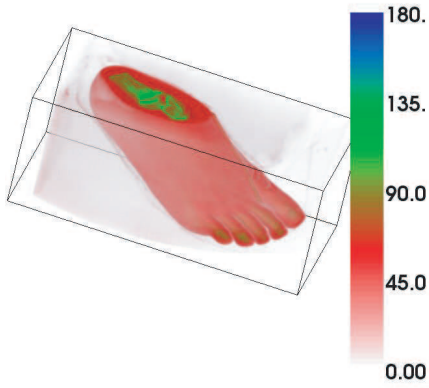


Figure 15: *the Input data of Human Foot*

a recursive search algorithm that collects subdivision information from neighbors to avoid cracks in the volume approximation. Then, from the boundary of the neighbors referenced, we de ned a region of reference for each grid element. This local de nition 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

[Bey95] BEY J.: Tetrahedral mesh ri nement. *Computing Vol.55*, No.13 (1995), pp.355–378.

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

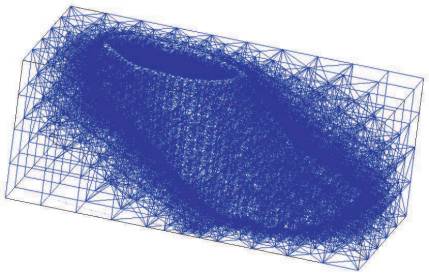


Figure 16: *the result of Human Foot*

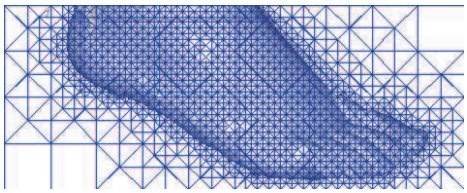


Figure 17: *the result of Human Foot at level 9*

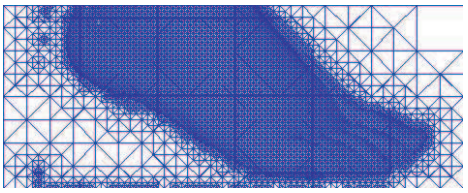


Figure 18: *the result of Human Foot at level 12*

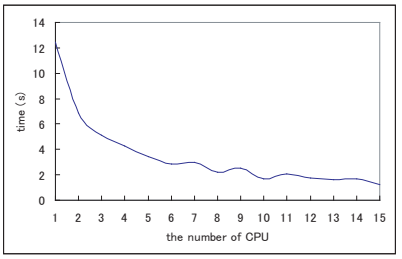


Figure 19: *Computing time V.S. # of CPUs at (level 6)*

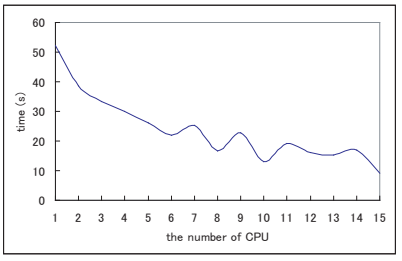


Figure 20: *Computing time V.S. # of CPUs at (level 9)*

face octrees for the representation of three-dimensional objects. *Computer Vision Graphics and Image Processing Vol.36* (1986), pp.100–113.

[GMPV02] GREGORSKI B., M.DUCHAINEAU, P.LINDSTROM, V.PASCUCCI: Interactive view-dependent rendering of large isosurfaces. *IEEE Visualization 2002* (2002), pp.475–482.

[JM.95] JM.MAUBACK: Local bisection re nement for n-simplicial grids generated by re ection. *SIAM Journal of Scientific Computing Vol.16*, No.1 (1995), pp.210–227.

[Kau01] KAUFMAN A.: State of the art in volume graphics. In *Volume Graphics*. Springer, 2001, pp. pp.3–28.

[NHR99] NIELSON G., HOLLIDAY D., ROXBOROUGH R.: Cracking the cracking problem with coons patches. *Proc.IEEE Visualization '99, San Francisco, CA* (1999).

[Nie97] NIELSON G.: *Tools for triangulations and tetrahedrizations int Scientific Visualization*. IEEE CS, 1997.

[Nie01] NIELSON G.: Volume modelling. In *Volume Graphics*. Springer, 2001, pp. pp.29–48.

[Tan95] TANAKA H.: Accuracy-based sampling and reconstruction with adaptive meshs for parallel hierarchical triangulation. *COMPUTER VISION AND IMAGE UNDERSTANDING Vol.61*, No.3 (1995), pp.335–350.

[TF93] TANAKA H., F.KISHINO: Adaptive mesh generation for surface reconstruction: Parallel hierarchical triangulation without cracks. *Proc.IEEE 10th International Conference on Pattern Recognition* (1993), pp.88–94.

[TG00] T.ROXBOROUGH, G.M.NIELSON: Tetrahedron based, least squares, progressive volume models with application to freehand ultrasound data. *IEEE 2000* (2000), pp.93–100.

[THJW99] TROTTS.I, HAMANN.B, JOY.K, WILEY.D: Simpli cation of tetrahedral meshes with error bounds. *IEEE Transactions of Visualization and Computer Graphics* (1999).

[ZCK97] ZHOU Y., CHEN B., KAUFMAN A.: Multiresolution tetrahedral framework for visualising regular volume data. *Proc.IEEE Visualization '97, Phoenix, AZ* (1997), pp.135–142.