面心立方格子上の等値面生成に関する高速化手法
－Interval-tree を用いた境界セル探索の高速化－

高橋 友和† 井手 一郎† 目加田 慶人‡ 村瀬 洋† 米倉 達広††

†名古屋大学大学院情報科学研究科 名古屋市千種区不老町
‡中京大学 生命システム工学部
E-mail: ttakahashi@murase.m.is.nagoya-u.ac.jp

あらまし 面心立方（Face-Centered Cubic）格子上のボリュームデータを対象とした等値面パッチ生成に関する高速化手法を提案する．本手法は，Interval-tree（区間木）と呼ばれるデータ構造を用いることで，ボリュームデータ内の境界セルを高速に探索し，それによって等値面生成全体の計算コストを削減するものである．メタボールによって生成したボリュームデータと面心立方格子上に再標本化された医用 CT データの２つのボリュームデータを対象とし，閾値を連続的に変化させながら等値面生成を繰り返す実験を行い，Interval-tree を使用する場合と使用しない場合での境界セル探索時間，ならびに全体の等値面生成時間を比較することにより，提案手法の有効性を確認した．

キーワード 面心立方格子，ボリュームデータ，等値面生成，高速化，Interval-tree，境界セル探索

A Speedup Method for Isosurfacing Volumetric Data Sampled with a Face-Centered Cubic Lattice
－An Approach using Interval-Trees to Speedup of Active-Cell Search－

T. Takahashi† I. Ide† Y. Mekada‡ H. Murase† and T. Yonekura††

†Graduate School of Information Science, Nagoya University, Japan
‡Life System and Technology, Cyukyo University, Japan
E-mail: ttakahashi@murase.m.is.nagoya-u.ac.jp

Abstract We propose a speedup method for isosurfacing volumetric data sampled with a Face-Centered Cubic (FCC) lattice in this report. This reduces the whole cost by using interval-trees for speedup of searching volumetric data for active-cells. We generated isosurfaces and measured the times of active-cell search and of Isosurfacing while changing thresholds sequentially. Two sets of volumetric data were prepared for the experiments. One of those was generated by metaball-functions and another was re-sampled from medical CT data. The experimental results demonstrated the effectiveness of our speedup method.

Keyword Face-Centered Cubic lattice, Volumetric data, Isosurfacing, Speedup, Interval-tree, Active-cell search

1. はじめに

連続な 3 次元のスカラ場において，ある等しい値を持つ点の集合が形成するいくつかの閉曲面のことを等値面（等スカラ面）と呼ぶ．等値面は，ある閾値を与えられたとき，その閾値より大きい値を持つ領域と小さい値を持つ領域を分割する境界面と考えることができる．一般にボリュームデータとは，連続な 3 次元のスカラ場を立方格子の各格子点位置において離散的に標本化したものであり，代表的なものとしては 3 次元 CT データが挙げられる．ボリュームデータから，等値面を 3 角形パッチの集合として近似的に抽出する手法は，スカラ場の持つ幾何学的な特徴の抽出や解析等に有用であり，例えば医用 CT データを対象とする場合には，医用画像処理技術と連携して，胃腸や気管支等の管腔形状を持つ臓器や骨格の形状解析や可視化等に用いられる．

Marching Cubes（MC）法[1]は，立方格子上に標本化されたボリュームデータから等値面をパッチモデルとして生成する手法であり，アルゴリズムが単純であるのに対し，比較的良好な等値面形状が得られるという理由から，最も幅広く利用され，これに対して近年になっても多くの関連手法が提案されている[2,3]．MC 法では，ボリュームデータをセルと呼ばれる小さな立方体領域で空間的に分割し，任意の閾値が与えられた

とき，その等値面と交差するセル（以後，境界セルと呼ぶ）を探索し，その内部に等値面を近似するような3 角形パッチ群を生成する．この等値面パッチ生成は各セルについて独立に処理される．一方，MC 法が立方格子上のボリュームデータを対象とするのに対して，等値面の近似精度や形状の滑らかさの向上を目的として，体心立方格子や面心立方格子（以後，FCC 格子と呼ぶ）をボリュームデータの標本化格子として用いる手法が提案されている[4-9]．これらの手法では，様々な形状のセルが等値面生成に用いられる．FCC 格子（図1）上のボリュームデータ（以後，FCC ボリュームデータと呼ぶ）からの等値面生成手法（以後，FCC 法と呼ぶ）[6]に関して，現在までに等値面形状の滑らかさに関する報告[6,8]，近似精度に関する報告[7,8]，医用CT データの可視化応用に関する報告[9]がなされ，それらに関する有効性が確認されている．

一方，特に大規模なボリュームデータを扱う場合や閾値の変化に伴う等値面の形状変化をリアルタイムに観察するような場合には，等値面生成処理の高速化が求められ，これに関する報告も数多くなされている[10-13]．処理の高速化の観点から，等値面生成は以下に挙げる 2 つの処理に大きく分けられる．

・ ボリュームデータ中の境界セル探索
・ 等値面パッチの各頂点の座標・法線ベクトル計算

前者の境界セル探索は，ある閾値が与えられたとき，ボリュームデータ中の全てのセルの集合から，境界セルの集合を抽出する処理である．これに関して， 1 つの境界セルから等値面に沿うようにその近傍の境界セルを次々に探索していく手法[10]や，各セルをセルの頂点が持つ値の最大値と最小値からなる区間とみなし，与えられた閾値が含まれる区間を探索する問題に帰着させる手法[11-13]が提案されている．後者の頂点計算は，各境界セルに関して，セルの各頂点座標と各頂点が持つ値，ならびに与えられた閾値から，等値面パッチの各頂点座標とその法線ベクトルを算出する処理である．これに関しては，境界セルと等値面の交差のパターンに対応した頂点計算のためのテーブルを予め用意しておく手法や，セルの各頂点位置における法線ベクトルを前処理として計算し，保持しておく手法，ハッシュ表を用いて頂点計算の重複を回避する手法が報告されている．等値面は空間中の閉曲面群であることから，一般的にセルの総数に対する境界セル数の割合は極端に小さくなる場合が多い．そのため，境界セル探索の高速化は，頂点計算の高速化と比較して，全体の処理の高速化に対する寄与がより高いと考えられる．しかし，現在までに FCC 法の等値面生成に関する境界セル探索の高速化に関する報告はない．

そこで本報告では，上記の中でも特に Interval-tree（区間木）と呼ばれるデータ構造を用いた境界セル探索の高速化[11]に着目し，FCC 法による等値面生成を Interval-tree を用いて高速化する手法を提案する．前述した境界セル探索，頂点計算に共通して，高速化を行う場合に様々なデータ構造を用いるため，それらを保持するための付加的なメモリコストが余計にかかる．このことから，計算コストとメモリコストはトレードオフの関係にあり，高速化を考える場合には両コストの増減のバランスを考慮する必要があると考える．これに関して，提案手法では以下に挙げる 2 つの点の実現を目指した．

・ Interval-tree を用いることによる，FCC 法の境界セル探索に関する計算コストの削減
・ FCC ボリュームデータの効率的なメモリ配置の実現による，Interval-tree の使用に伴う付加的なメモリコストの削減

ボリュームデータの効率的なメモリ配置とは，標本化空間とメモリ空間の位置的な対応付けが可能であるような配置を指す．通常の立方格子上のボリュームデータの場合，これは容易に行うことができるが，FCC ボリュームデータの場合，これを容易とするためには何らかの仕組みが必要であると考える．このことについて，FCC 法に関する報告[6-9]の中には明確な言及がなされていなかったが，提案手法と密接な関係があるため，本稿において実現を検討した．

以降，2 節において，等値面生成手法のうち，MC 法と FCC 法のアルゴリズムの概略と，境界セル探索の高速化に関連する手法について述べる．3 節において，提案手法のアルゴリズムとして，FCC 手法の高速化手法として，FCC ボリュームデータの効率的な配置の実現方法と，FCC 手法の境界セル探索に関する Interval-tree を用いた高速化手法について述べる．4 節では，提案手法の有効性を確認するために行った実験の方法，ならびに結果とその考察について述べ，5 節において本稿をまとめる．

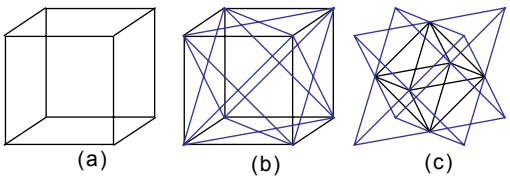


図 1．面心立方格子：(a)は通常の立方格子の隣接する 8 格子点が形成する立方体を示す．(b)，(c)は共に面心立方（FCC）格子であり，(a)の各正方形面の重心位置にもう 1 つの格子点を持つ．

2. 関連研究

2.1. MC 法

MC 法では，ボリュームデータを立方格子上の隣接する 8 つの格子点を頂点として持つ立方体セルに分割する(図 2). 本稿では便宜上, セルの各頂点について，閾値以上の値を持つ頂点を内部点（internal point)，閾値より小さい値を持つ頂点を外部点（external point）と呼ぶ．内部点と外部点を結ぶ線分は，必ず一度は等値面と交差すると考えられるから，あるセルが内部点と外部点を持つとき，そのセルは境界セルとなる．すべての境界セルに対して，内部点と外部点を分離するような等値面パッチ群を生成する．内部点と外部点の対を境界対と呼ぶ．等値面パッチの各頂点の座標と法線ベクトルは, 境界対を両端点とするセルの稜線上に，閾値とそれら 2 点の値，座標，法線ベクトルから線形補間を用いて算出される．この頂点計算処理は，セルの内部点，外部点の配置の組み合わせに対応して，頂点計算の方法を記述したテーブルを予め用意しておくことにより，テーブルマッチングを用いた高速化が図られる．それらの組み合わせは，全部で 256 通り存在するが，セルの回転，対称を考慮することによって 22 通りに統合される. MC 法の詳細については[1]に譲る.

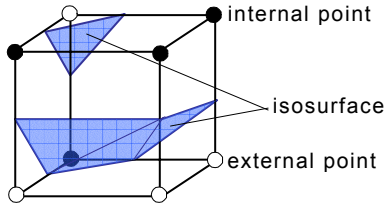


図 2. MC 法における立方体セル内の等値面パッチ生成

2.2. FCC 法

FCC ボリュームデータは，FCC 格子上の隣接する 6 つの格子点を頂点として持つ正 8 面体と，隣接する 4 つの格子点からなる正 4 面体の 2 つの多面体によって隙間なく分割することができる（図 1(c)）．このとき，両多面体の位置関係は，隣接する 3 つの格子点からなる正 3 角形面が必ず正 8 面体と正 4 面体によって共有されるような関係になっている．FCC 手法のセルは，1 つの正 8 面体とそれに隣接する 8 つの正 4 面体の 9 つの多面体群によって形成され，14 個の格子点を持つ（図 3）．ただし，境界セル判定の際には，中央の 8 面体の 6 頂点が持つ値のみが使用される．8 面体の 6 頂点に対する内部点，外部点の配置の組み合わせは全部で 64 通り存在し，セルの回転，対称，内外の反転を考慮することで 6 通りに統合されるため，MC 法と比較して，等値面パッチの頂点計算の高速化に用いるテーブルの作成が容易となる．等値面パッチの頂点は，8 面体と各 4 面体の内部にそれぞれ生成される．具体的には，セル内の各多面体について，多面体が境界対を持つ場合，境界対に挟まれる全ての稜線上に一時的な頂点を線形補間し，それらの点群の幾何重心位置に等値面パッチの頂点を生成する．FCC 手法の詳細については[6-9]に譲る.

2.3. 境界セル探索の高速化手法

境界セル探索の高速化には，主に以下の 2 つのアプローチが存在する.

- 等値面に沿って境界セルを探索するもの[10]
- セルの頂点が持つ値の最大値と最小値の情報を利用するもの[11-13]

前者は，境界セル同士が互いに隣接関係にあるという性質を利用したものである．初めに始点となる境界セルを探索し，その境界セルに隣接するセルの中からまだ未探索の境界セルを探索する処理を繰り返す．ただし，この場合，探索される境界セルは，1 つの閉じた等値面に対するもののみである．一般的に等値面は複数の閉曲面により形成されるため，ボリュームデータ中の全ての境界セルを探索するためには，一連の処理を閉曲面の数だけ繰り返す必要がある.

後者は，各セルをセルの頂点が持つ値の最大値と最小値からなる区間とみなし，与えられた閾値が含まれる区間を探索する問題に帰着させるものである．これに関して，通常の立方格子上のボリュームデータ内の立方体セルを対象とした場合には，ボリュームデータを空間的に 8 つに分割する処理を繰り返す 8 分木構造と呼ばれる階層構造を利用する手法[12]が提案されているが，FCC ボリュームデータへの適用はセル形状の違いから困難であると考える．一方，任意の値が含まれる区間の探索は，最小値と最大値を 2 つの軸とする 2 次元平面上に各区間をプロットすることにより，2 次元の探索問題とみなすことができ，この探索の高速化に関する報告がなされている[11,13]．中でも Interval-tree と呼ばれる 2 分木構造を用いた手法[11]は，理論的に最も高速であるとされている．このため，提案手法では，境界セル探索の高速化のために Interval-tree を用いた．Interval-tree の構築，ならびにある値が含まれる区間の探索のアルゴリズムを擬似コードの形式で以下に示す.

＜構築＞　区間 i は最小値 i.min と最大値 i.max を持つ．J 個の区間を要素とする集合 $S=\{i_j\}\ (j=0,1,2,...,J-1)$ から Interval-tree を構築する．Interval-tree の各ノード n は，代表値 n.value として持ち，n.value が含まれる区間の集合を区間の最小値について昇順ソートしたリスト n.min_sorted_list と，最大値について降順ソートしたリスト n.max_sorted_list の 2 つのリストをそれぞれ保持する．ここで，n.value は S 内の区間を全て含むような最小幅の区間を算出し，その最小値と最大値の中央値とする．また，n は左の子ノード n.left_node と右の子ノー

ド n.right_node を持つ．

n := root; $S := \{i_j\}, (j = 0,1,2,...,\mathrm{J}-1)$
make_interval_tree(S, n)
  n := new_node( );
  n.value := median(S);
  n.left_node := NULL; n.right_node := NULL;
  $S_{Boundary} := \varphi$; $S_{Upper} := \varphi$; $S_{Lower} := \varphi$;
  $j := 0$;
  while $(i_j \in S)$
    if $(i_j.\max < n.\mathrm{value})$ then $S_{Lower} := S_{Lower} \cup \{i_j\}$;
    else if $(i_j.\min \geq n.\mathrm{value})$ then $S_{Upper} := S_{Upper} \cup \{i_j\}$;
    else $S_{Boundary} := S_{Boundary} \cup \{i_j\}$;
    j++;
  n.min_sorted_list := sort_min_inc $(S_{Boundary})$;
  n.max_sorted_list := sort_max_dec $(S_{Boundary})$;
  if $(S_{Lower} \neq \varphi)$ then
    make_interval_tree$(S_{Lower}, n.\mathrm{left_node})$;
  if $(S_{Upper} \neq \varphi)$ then
    make_interval_tree$(S_{Upper}, n.\mathrm{right_node})$;

＜検索＞　任意の値 θ が与えられたとき，2 分木探索と各ノードが保持する 2 つのリスト内の線形走査を複合的に利用することにより，θ が含まれる区間の集合 S_θ を高速に検索する．

n := root; $S_\theta := \varphi$;
find_intervals (S_θ, θ, n)
  if $(\theta < n.\mathrm{value})$ then
    $j := 0$;
    while $(i_j \in n.\mathrm{min\_sorted\_list}$ & $i_j.\min < \theta)$
      $S_\theta := S_\theta \cup \{i_j\}$; j++;
    if $(n.\mathrm{left\_node} \neq \mathrm{NULL})$ then
      find_intervals$(S_\theta, \theta, n.\mathrm{left_node})$;
  else if $(\theta > n.\mathrm{value})$ *then*
    $j := 0$;
    while $(i_j \in n.\mathrm{max\_sorted\_list}$ & $i_j.\max \geq \theta)$
      $S_\theta := S_\theta \cup \{i_j\}$; j++;
    if $(n.\mathrm{right\_node} \neq \mathrm{NULL})$ then
      find_intervals $(S_\theta, \theta, n.\mathrm{right\_node})$;
  else
    $j := 0$;
    while $(i_j \in n.\mathrm{min\_sorted\_list})$
      $S_\theta := S_\theta \cup \{i_j\}$; j++;

Interval-tree の構築，検索に必要な情報は，区間の最大値と最小値のみであるが，これを境界セル探索に用いる場合，あるセルが探索されたとき，そのセルに対する等値面パッチ生成を行うためにそのセルの各頂点の位置や値の情報を取得する必要がある．これらの情報を全て Interval-tree 上に保持することはメモリコストの観点から好ましくない．MC 法の場合，以下の 2 つの性質を利用し，探索されたセルの再構築を行うことで，メモリコストの削減が可能としている[11].

- 標本化空間において，セルを代表する頂点の位置から，相対的に他の頂点の位置が決定可能である
- 標本化空間とメモリ空間の位置的な対応付けが容易に可能である

FCC 法の場合，前者の性質は満たしているものの，後者に関しては言及されていない．そこで本稿では，FCC 手法に対して Interval-tree を用いた高速化を行うにあたり，後者の性質を満たすような FCC ボリュームデータの効率的なメモリ配置の実現を検討し，付加的なメモリコストの削減を図った．

3. 提案手法のアルゴリズム

3.1. FCC ボリュームデータのメモリ配置

通常の立方格子上のボリュームデータが 3 次元配列に保持されているとし，配列の各要素を $G(i,j,k)$ ($i = \{0,1,..,\mathrm{I}-1\}$，$j = \{0,1,..,\mathrm{J}-1\}$，$k = \{0,1,..,K-1\}$) で表す．ボリュームデータが式(1)に示すようにメモリ空間内で連続して保持されていることを前提とする．ここで p は $G(i,j,k)$ が保持されているメモリ空間での位置を表す．標本化空間の 2 つの格子点の相対位置を $(\Delta_x, \Delta_y, \Delta_z)$ で表し，その 2 点の値が保持されているメモリ空間での相対位置を Δ_p で表すとき，式(2)，(3)を用いて，標本化空間とメモリ空間の位置的な対応付けが容易に行える．

$$p = \mathrm{IJ}k + \mathrm{I}j + i \quad (1)$$

$$\Delta_p = \mathrm{IJ}\frac{\Delta_z}{d} + \mathrm{I}\frac{\Delta_y}{h} + \frac{\Delta_x}{w} \quad (2)$$

$$\begin{pmatrix}\Delta_x\\ \Delta_y\\ \Delta_z\end{pmatrix} = \begin{pmatrix}(\Delta_p \% \mathrm{I})w\\ \left\lfloor \frac{\Delta_p}{\mathrm{I}} \right\rfloor h\\ \left\lfloor \frac{\Delta_p}{\mathrm{IJ}} \right\rfloor d\end{pmatrix} \quad (3)$$

ここで w，h，d をそれぞれ x 軸方向，y 軸方向，z 軸方向の標本化間隔とし，i 方向，j 方向，k 方向が，それぞれ x 軸，y 軸，z 軸と重なるものとする．

上記のような対応付けを容易に行うため，FCC ボリュームデータのメモリ配置を以下のように検討した．

＜標本化＞　データ形式として式(1)のようにメモリ空間内で連続して保持された 3 次元配列を用いる．配列中の任意の要素 $F_{CC}(i,j,k)$ を，連続なスカラ場 $f(x,y,z)$ から式(4)のように標本化する．

$$F_{CC}(i,j,k)=\begin{cases} f(iw,\ jh,\ kd) & k \text{ is even.} \\ f\left(\left(i+\frac{1}{2}\right)w,\ \left(j+\frac{1}{2}\right)h,\ kd\right) & k \text{ is odd.} \end{cases} \quad (4)$$

このとき，特に各標本化間隔の比が式(5)の等式を満たすとき，FCC ボリュームデータは完全に等方となる．

$$w:h:d=1:1:\sqrt{2}/2 \quad (5)$$

＜標本化空間とメモリ空間の対応付け＞　上記のように作成された FCC ボリュームデータを用いて，標本化空間とメモリ空間の位置的な対応付けを行う．FCC 格子上の 2 つの格子点の相対位置 $(\Delta_x,\Delta_y,\Delta_z)$ に対応するメモリ空間での相対位置は，基準となる一方の格子点の値を保持する要素の k 方向の添え字の偶奇に依存して変化する．基準となる要素の k 方向の添え字が偶数のときのメモリ上での相対位置を Δ_{p_even}，奇数のときを Δ_{p_odd} とし，式(6)のように Δ_z に対応する k 方向の添え字の差を Δ_k とすると，標本化空間とメモリ空間の対応付けは，式(7)～(9)によって実現できる．

$$\Delta_k=\frac{\Delta_z}{d}=\left\lfloor\frac{\Delta_{p\_even}}{\mathrm{IJ}}\right\rfloor=\left\lfloor\frac{\Delta_{p\_odd}}{\mathrm{IJ}}\right\rfloor \quad (6)$$

$$\Delta_{p\_even}=\begin{cases} \mathrm{IJ}\frac{\Delta_z}{d}+\mathrm{I}\frac{\Delta_y}{h}+\frac{\Delta_x}{w} & \Delta_k \text{ is even.} \\ \mathrm{IJ}\frac{\Delta_z}{d}+\mathrm{I}\left(\frac{\Delta_y}{h}-\frac{1}{2}\right)+\frac{\Delta_x}{w}-\frac{1}{2} & \Delta_k \text{ is } odd. \end{cases} \quad (7)$$

$$\Delta_{p\_odd}=\begin{cases} \mathrm{IJ}\frac{\Delta_z}{d}+\mathrm{I}\frac{\Delta_y}{h}+\frac{\Delta_x}{w} & \Delta_k \text{ is even.} \\ \mathrm{IJ}\frac{\Delta_z}{d}+\mathrm{I}\left(\frac{\Delta_y}{h}+\frac{1}{2}\right)+\frac{\Delta_x}{w}+\frac{1}{2} & \Delta_k \text{ is } odd. \end{cases} \quad (8)$$

$$\begin{pmatrix}\Delta_x\\ \Delta_y\\ \Delta_z\end{pmatrix}=\begin{cases} \begin{cases} (\Delta_{p\_even}\%\,\mathrm{I})w, & (\Delta_{p\_odd}\%\,\mathrm{I})w & \Delta_k \text{ is even.} \\ \left(\Delta_{p\_even}\%\,\mathrm{I}+\frac{1}{2}\right)w, & \left(\Delta_{p\_odd}\%\,\mathrm{I}-\frac{1}{2}\right)w & \Delta_k \text{ is odd.} \end{cases} \\ \begin{cases} \left\lfloor\frac{\Delta_{p\_even}}{\mathrm{I}}\right\rfloor h, & \left\lfloor\frac{\Delta_{p\_odd}}{\mathrm{I}}\right\rfloor h & \Delta_k \text{ is even.} \\ \left(\left\lfloor\frac{\Delta_{p\_even}}{\mathrm{I}}\right\rfloor+\frac{1}{2}\right)h, & \left(\left\lfloor\frac{\Delta_{p\_odd}}{\mathrm{I}}\right\rfloor-\frac{1}{2}\right)h & \Delta_k \text{ is odd.} \end{cases} \\ \left\lfloor\frac{\Delta_{p\_even}}{\mathrm{IJ}}\right\rfloor d, & \left\lfloor\frac{\Delta_{p\_odd}}{\mathrm{IJ}}\right\rfloor d \end{cases} \quad (9)$$

3.2. Interval-tree を用いた FCC 法の高速化

＜Interval-tree の構築＞　FCC ボリュームデータ中の各セルについて中央の 8 面体のみに注目し，その 6 つの頂点が持つ値の最大値と最小値を算出する．ここで，最大値と最小値が等しい場合，そのセルは境界セルになり得ないセルとして，探索対象から外すことにより探索時間の削減を図る．セルを代表する頂点の位置を予め 1 つ決定しておき，各セルの代表点に対応する，メモリ空間でのボリュームデータの先頭位置からの相対位置を，そのセルの識別子とする．ボリュームデータの先頭位置は，3 次元配列中の $(i,j,k)=(0,0,0)$ 要素に対応するため．セルの識別子は式(7)で算出される．各セルの識別子，最大値，最小値の情報から，Interval-tree を構築する．この Interval-tree の構築は前処理として 1 度だけ行われる．任意の閾値 θ が与えられたとき，同一の Interval-tree を用いて，以降の処理を行う．

＜境界セルの探索＞　Interval-tree の検索操作により，閾値 θ に対する境界セルの集合 S_θ を求める．S_θ に属するすべてのセルについて以降の処理を行う．ここで，メモリ空間における不連続なアドレス参照に伴うキャッシュのミスヒットを減少させるために，S_θ をセルの識別子について昇順ソートしておく．

＜セルの再構築＞　標本化空間において，セルの代表点を基準とした他の 13 頂点の相対位置を予め算出しておき，それらの値とセルの識別子から，式(7)，(8)を用いて各頂点の値を取得し，識別子に対応するセルの再構築を行う．

＜等値面生成＞　再構成されたセルに対して，セル中央の 8 面体の各頂点の内部点，外部点の配置の組み合わせから，テーブルマッチングにより生成される等値面パッチ群のパターンを判別し，等値面生成に必要な各頂点の座標，各頂点における法線ベクトルを算出する．詳細については[6-9]に譲る．

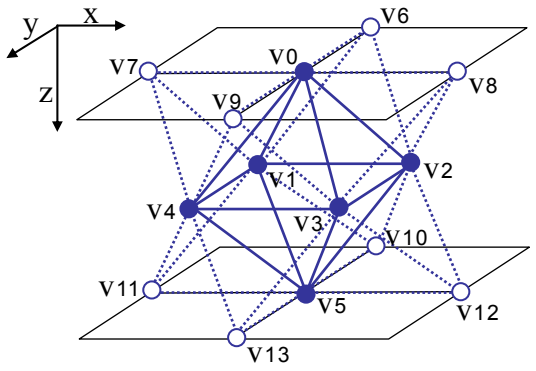


図 3．FCC 法のセル：1 つの正 8 面体と 8 つの正 4 面体からなる．V0 ～ V13 の 14 個の頂点から構成される．中央の 8 面体の 6 つの頂点 V0 ～ V5 の値のみが境界セル判定に用いられる．

4. 実験と考察

4.1. 実験方法

200 個のメタボールをランダムに発生させ，FCC 格子上に標本化した人工ボリュームデータ (サイズ

$(\mathrm{I},\mathrm{J},\mathrm{K})=(128,128,181)$，標本化間隔比 $w:h:d=1:1:\sqrt{2}/2$，セル総数 2841804) と，CT データを FCC 格子上に再標本化したもの（$(\mathrm{I},\mathrm{J},\mathrm{K})=(128,128,244)$，標本化間隔 $(w,h,d)=(2.5,2.5,2.0)[mm]$，セル総数 3841992) を対象に等値面生成実験を行った．ここで，等値面パッチの頂点計算の計算コスト削減を目的に，各格子点位置での法線ベクトルを予め FCC ボリュームデータと同様な方法で計算し，作成した法線ベクトルデータを使用した．また，実験中の使用メモリ量削減を目的にボリュームデータの要素の最小値と最大値がそれぞれ 0，65535 になるように，各要素の値を 2 バイト符号無し整数で表現したものを用いた．法線ベクトルデータについては各要素の x，y，z をそれぞれ 2 バイト整数で表現した．

実験では，様々な閾値 θ を 0.0～65535.0 の浮動小数点数で与え，そのときの等値面生成全体でかかった時間を計測し，Interval-tree を使用する場合（提案手法）と使用しない場合（従来手法）で比較した．提案手法の場合には，Interval-tree の構築時間，θ に対する境界セル数，境界セル探索時間も計測した．実験では CPU: Intel Xeon 3.0GHz, L2 キャッシュ容量: 512KB, 主記憶容量: 2.0GB 相当の計算機を使用した．

4.2. 結果と考察

人工データに対する Interval-Tree の構築時間は 1,312msec，CT データでは 1,610msec であった．与えられた θ における等値面生成時間を提案手法（with Interval-tree）と従来手法（without Interval-tree）で比較した結果を各データについて図 4,5 に示す．提案手法について，境界セル数に対する探索時間（active-cell search）と等値面生成時間（isosurfacing）を各データについて図 6,7 に示す．また，本実験において生成された等値面の一例を図 8 に示す．図 8 に関して，(a)～(f)は θ をある一定の間隔で変化させたときの等値面形状の変化を示す．また，(g)～(i)は CT データからの等値面生成の一例であり，(g)は骨格領域の境界形状，(h)は空気領域と人体組織領域との間の境界形状，(i)は最大の境界セル数を与える θ に対する等値面形状をそれぞれ示す．(h)，(i)については内部の等値面形状を可視化する目的で半透明表示を行った．

＜計算コストに関する考察＞ 提案手法に関して，Interval-Tree の構築は前処理として行うため，一度構築してディスク等に保存することで，次回以降は構築する必要はない．提案手法は Interval-tree の使用により，等値面生成全体の計算コストを大幅に削減することができており（図 4,5），その計算コストは境界セル数にほぼ比例して変化する(図 6,7). 境界セル探索の計算コストに関して，ボリュームデータ中の全セル数を n，境界セル数を b とすると，従来手法は，線形走査による境界セル探索を行う場合と同様であるとみなせるため，その最悪の計算コストは $O(n)$ で表される．それに対し，提案手法の場合，Interval-tree を用いることにより，2 分木探索と線形走査を複合的に用いるため，最悪の計算コストは $O(b+\log n)$ で表される．一般に $b<<n$ が十分に仮定できるため，他のボリュームデータに対しても，本実験結果と同様な大幅な計算コストの削減が実現できると考えられる．

＜メモリコストに関する考察＞ Interval-tree は，各ノードに区間情報を異なる基準でソートした 2 つのリストを保持することと，異なるノード間で区間の重複がないことから，区間の総数を n，各区間の情報量を i とすると，そのメモリコストは概ね $2ni$ で表される．そのため，i を小さくすることが，メモリコストの削減につながる．提案手法に関して，各セルに対応する区間情報は，頂点の値の最大値，最小値，そして，セルの代表点位置での値を保持するメモリ空間での位置のみであり，FCC ボリュームデータの効率的なメモリ配置の実現により，境界セルの高速な探索に用いる Interval-tree を保持する付加的なメモリコストを最小限に抑えることができた．

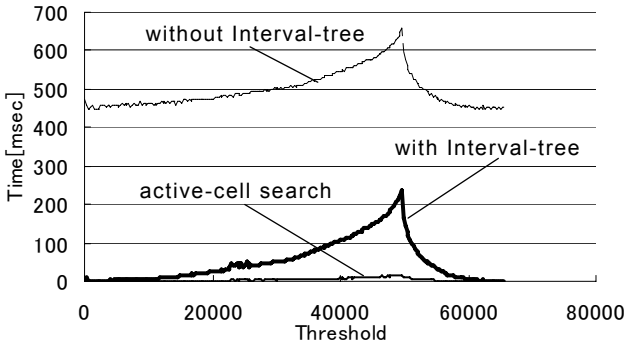


図 4. 人工データに対する等値面生成時間：提案手法の生成時間(with Interval-tree)には境界セル探索時間(active-cell search)が含まれる．

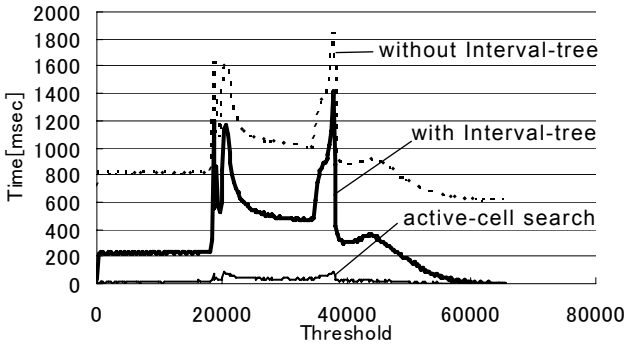


図 5. CT データに対する等値面生成時間：提案手法の生成時間(with Interval-tree)には境界セル探索時間(active-cell search)が含まれる．

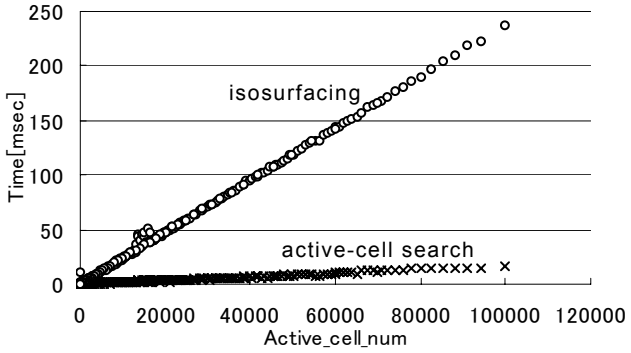


図 6. 人工データに対する境界セル数に対する提案手法の境界セル探索時間と等値面生成時間：生成時間(Isosurfacing)には探索時間(active-cell search)にはが含まれる.

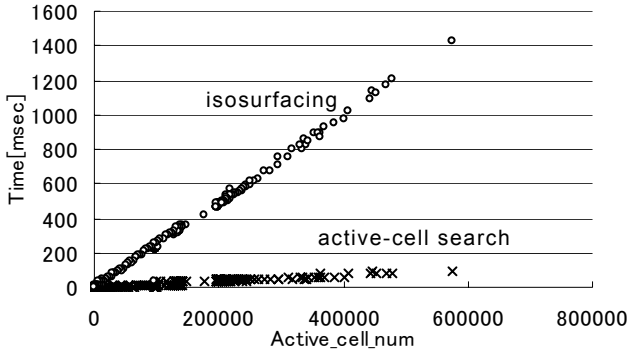


図 7. CT データに対する境界セル数に対する提案手法の境界セル探索時間と等値面生成時間：生成時間(Isosurfacing)には探索時間(active-cell search)にはが含まれる.

5. おわりに

FCC格子上のボリュームデータを対象とした等値面パッチ生成に関する高速化手法を提案した．提案手法は，計算コストとメモリコストに関する以下の 2 つの有用な性質を持つ.

- Interval-tree を用いた境界セル探索に要する計算コストの削減
- FCC ボリュームデータの効率てきなメモリ配置の実現による，高速化に要する付加的なメモリコストの削減

メタボールによって人工的に作成したボリュームデータと 3 次元 CT データの２つのボリュームデータを対象とした実験により，提案手法の有効性を確認した．今後は，他の境界セル探索の高速化手法との比較，等値面パッチの頂点計算の高速化を検討したい.

謝辞 日頃より熱心御討論いただく村瀬研諸氏に感謝する．本研究の一部は日本学術振興会科研費，21 世紀 COE プログラム，厚生労働省がん研究助成金によった.

文 献

[1] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," Computer Graphics, 21(4): 163-169, Jul. 1987.

[2] Gregory M. Nielson, "Dual marching cubes," Proc. of IEEE VIS2004, 489-496, Oct. 2004.

[3] M. Bertram, "Volume refinement fairing isosurfaces," Proc. of IEEE VIS2004, 449-456, Oct. 2004.

[4] G. M. Treece, R. W. Prager, and A. H. Gee, "Regularised marching tetrahedra: improved isosurface extraction," Computers and Graphics, 23(4): 583-598, 1999.

[5] H. Carr, T. Theusl, and T. Moller, "Isosurface on optimal regular samples," IEEE TCVG VisSym 2003, 39-48, May 2003.

[6] T. Takahashi, Y. Mekada, H. Murase, T. Yonekura, "High quality isosurface construction from volumetric data sampled with a face-centered cubic lattice," Proc. of ACCV2004, 2: 872-877, Jan. 2004.

[7] 高橋，目加田，村瀬，米倉，"面心立方格子データから生成された等値面の誤差評価，"VC-GＣＡＤ合同シンポジウム，2004 年 6 月.

[8] 高橋，目加田，村瀬，米倉，"面心立方格子上のボリュームデータからの等値面パッチ生成手法の性能評価，"画像電子学会， Vol.33, No.4-B, pp.547-554, 2004 年 8 月.

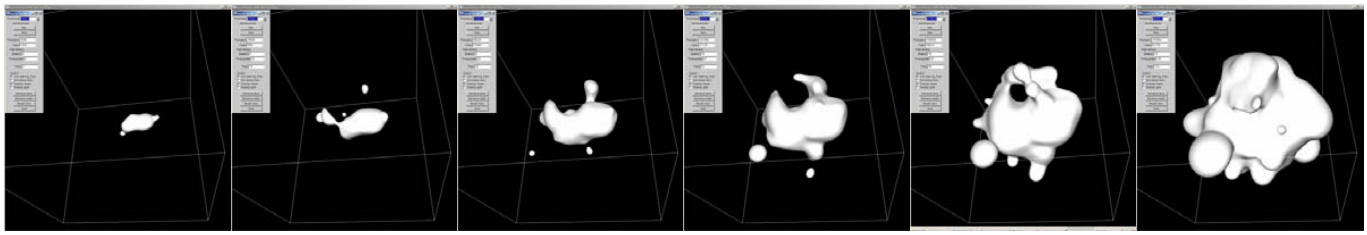
[9] T. Takahashi, Y. Mekada, H. Murase, T. Yonekura, "High Quality Isosurface Generation from Volumetric Data and Its Application to Visualization of Medical CT Data," Proc. of ICPR 2004, 3: 734-737, Aug. 2004.

[10] C. L. Bajaj, V. Pascucci, D. R. Schikore, "Fast Isocontouring for Improved Interactivity," Proc. of Symposium on Volume Visualization, 39-46, Oct. 1996.

[11] P. Cignoni, P. Marino, C. Montani, E. Puppo, R. Scopigno, "Speeding Up Isosurface Extraction Using Interval Trees," IEEE Trans. on Visualization & Computer Graphics, 3(2): 158-170, Apr. 1997.

[12] J. Wilhelms, A. V. Gelder, "Octrees for faster isosurface generation," ACM Trans. On Graphics. 11(3): 201-227, July 1992.

[13] H. Shen, C. D. Hansan, Y. Livnat, C. R. Johnson, "Isosurfacing in Span Space with Utmost Efficiency," Proc. of Visualizasion'96, 1: 287-294, Nov. 1996.



(a) θ =7500.0 (9,196)　(b) θ =15000.0 (30,144)　(c) θ =22500.0 (71,396)　(d) θ =30000.0 (107,000)　(e) θ =37500.0 (195,168)　(f) θ =45000.0 (316,480)

(g) θ =40000.0 (650,100)　(h) θ =32000.0 (1,002,792)　(i) θ =37888.0 (3,037,672)

図 8．等値面生成結果の一例：(a)～(f)は人工データ，(g)～(i)は CT データからの等値面．θ は閾値，括弧内の数値は 3 角形パッチ数．