社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2006－CVIM－154（21）
2006／5／18

ボリュームデータの動的アダプティブグリッド生成

山下 裕礼[†]　　高間 康文[†]　　山口 哲[†]
フイン クアン フイ ヴィエト[†]　　田中 弘美[†]

近年，穿刺は医術の練習として利用されている手順のひとつである．医師にとって手術をする際の触覚は重要である．そこで，本稿では穿刺シミュレーションの開発を目的とし，四面体によって表現された 3 次元物体に対してリアルタイムでの穿刺シミュレーションをするために Triangular Half-Face Data Structure というデータ構造を提案する．そのデータ構造は面に対して四面体の外側に法線を引き，その法線の反時計回りに格子点の向きを決めてやることで，隣接する四面体間で共有する面に対し，正面と逆面を設定することで隣接する四面体をすばやく検索しリアルタイムで穿刺シミュレーションする手法を提案する．

Dyanamic Adaptive Grid Generation of Volume Data

Hironori Yamashita ,[†] Yasufumi Takama ,[†]
Satoshi Yamaguchi ,[†] HUYNH QUANG HUY VIET [†]
and Hiromi T.Tanaka[†]

The insertion of needle is one of the procedures utilized in clinical practice.The force feedback from tool on anatomic structure is necessary for doctors.For real time maintaining the mesh conformity in tetrahedral meshes, it is necessary to have an effective data structure for referring to adjacent tetrahedral mesh elements. In this paper, we develop a data structure called the triabgular haif-face for real time simulation of needle inseation on model of 3D material represented by a tetrahedral mesh. Using the data structure, each face of tetrahedron has revercivle face which conformity tetrahedral has.

1. 研 究 背 景

近年，医用ボリュームデータを用いた手術シミュレーションシステムの開発が注目されている．その中でも穿刺手術は医療において，若手の医師が臨床医療に入る前の練習の一つとして用いられるものである．そして，医師にとって手術をする際の触覚は重要である．そこで，医術練習を目的とした穿刺シミュレーションは今まで多くの提案がなされてきた．Dimaio らは直接計測できない針にかかる接触力を求める方法を提案した[1]．また，Nienhuys らは 2 次元や 3 次元での弾性物体に対する穿刺シミュレーションの変形を計算する方法を提案した[2]．そして，彼らは高性能化するためにアダプティブメッシュを用いた．アダプティブメッシュは，2 等分生成のアルゴリズム[3][4] を用いることによ針の近傍では局所的にメッシュは細かく分割される[7]．さらに，アダプティブメッシュは三角形の最長辺を分割し，また整合性を維持するために分割された三角形に隣接する三角形を再帰的に分割することにより生成される．

四面体のメッシュの整合性をリアルタイムで維持するためには，隣接すつ四面体の面要素を参照する効果的なデータ構造が必要である．Half-Face Data Structure は隣接する四面体の面要素へ検索に適しているとして紹介されている[6]．しかしながら，Nienhuys らの方法では穿刺シミュレーションに対してリアルタイムでの整合性を維持するための適したデータ構造については述べられていない．

そこで，本稿ではアダプティブグリッドによって表現された 3 次元物体に対してリアルタイムでの穿刺シミュレーションをするために Triangular Half-Face Data Structure というデータ構造を提案し，シミュレーションによりその有効性を示した．

2. アダプティブグリッド

2.1 アダプティブグリッド生成法

アダプティブグリッド[7] は，四面体により適応的に

† 立命館大学 理工学研究科
Ritsumeikan University

空間分割されたボリューム表現を生成する手法である．アダプティブグリッドは，田中らが提案した平面分割を行う適応的格子生成法を３次元空間分割に拡張したものであるといえる．アダプティブグリッドにより，入力ボリュームデータを局所特徴の変化に適応的に分割生成された一様な小四面体集合の階層構造として表現することが可能となる．

2.2 再帰的四面体分割法

2.2.1 初期部分空間メッシュの生成

再帰的四面体分割法では，まず入力されたボリュームデータに対して，図 1 のように初期部分空間集合を生成する．

一般のボリュームデータは，X，Y，Z 方向に任意のサイズのデータである． アダプティブグリッドでは，以下の分割処理が立方体を前提としているため，ボリューム空間を任意のサイズの部分立方体空間にあらかじめ分割する．(図 1 (a))．これを初期立方格子メッシュと呼ぶ（また各々の立方体を，その要素と呼ぶこととする）．

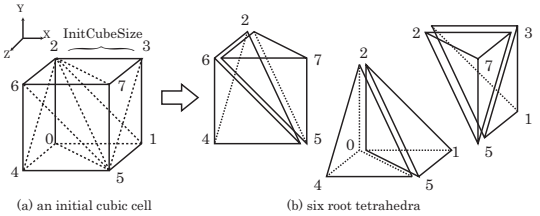

図 1 初期部分空間メッシュの生成

初期立方格子の各要素は，図 1 (b) のように６つのルート四面体に分割される．これら６つの四面体おいて鏡面対象は同じ形状とみなすと，全て同じ形状である．各ルート四面体は，特徴の変化に基づいて与えられた一様性基準を満たすまで，同時に独立に再帰的に２分割される．部分空間に分割することにより，各要素ごとに独立して処理を行うことが可能である．処理量は初期立方格子サイズから決定される分割深度により決定されるため，初期格子サイズを減少させることにより，処理を並列化し処理速度の高速化が期待できる．

2.2.2 再帰的四面体分割

再帰的四面体分割では四面体を再帰的に２分割していく（図２）．ボリューム空間が立方体であることを前提とすると，その立方体セルは６つの四面体に分割する．これら６つの四面体は，鏡面対称な全て同形状のセルとなる．この形状を直方四面体（図 2 (at level 3N)）と呼ぶことにする．この直方四面体を長い辺の

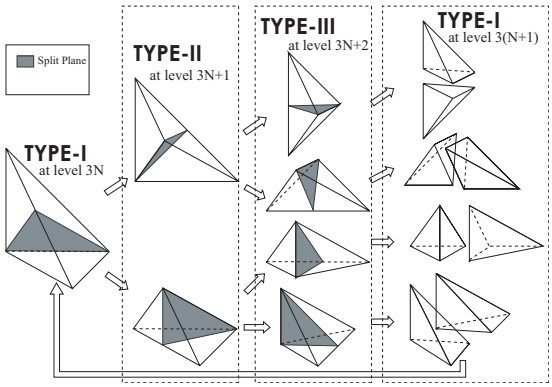

図 2 再帰的四面体分割法

中点により２分割を繰り返すと，３回２分割した時点で元の直方四面体の全ての辺を 1/2 した直方四面体（図 2 (at level 3N)）が生成される．その２分割の過程では，四面体 （直方四面体），四面体 （図 2 (at level 3(N+1))），四面体 （図 2 (at level 3(N+2))），四面体 （図 2 (at level 3N)）の順に生成されることになるが，これ以外の形状が生成されることはない．これらの形状は全ての面が直角三角形，２等辺三角形であり，統一された形状をしている．これらのことから，アダプティブグリッドは四面体の最も長い辺で分割するという簡単なアルゴリズムと生成される四面体の形状が特定の形状に限定され，いびつな形状が生成されないという点で特徴を持ったアルゴリズムであるといえる．

2.2.3 ク ラック

適応的分割における大きな問題は，それぞれの四面体が独立に分割された時に、隣接する格子要素間でしばしば異なった格子分割が行われることにより，格子表現から得られる近似多角形にクラックが発生するということである．クラックは大きくフィールド値が変化している面上で格子要素の一方だけが分割されることによって引き起こされるものである．図 3 において，四面体 ABCD が辺 AB 上の中点 F 二つの四面体 AFDC と四面体 AFDB に分割される時，辺 AB を共有し四面体 ABCD に隣接する四面体 ABCE が分割されていないと不整合が生じる．すなわち，隣接する四面体間（四面体 ABCE と四面体 AFDC 間，四面体 ABCE と四面体 AFDB 間）で近似精度に差が生じることになり．クラックが発生する．このクラック問題が発生すると，等値面を抽出した時に穴が生じたりする原因となる．

再帰的２分割において，全ての四面体は，基辺の中点を用いて分割される．中点を挿入することは，基辺

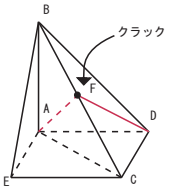

図 3　クラック問題

を共有する隣り合う四面体だけの分割に影響をおよぼすということである．本稿ではボリュームデータにアダプティブグリッドを用いて適応的四面体格子表現をしていて，アダプティブグリッドにおいてこのクラックが発生する可能性は３つのパターンがある．その影響を及ぼす範囲を図 4 で示す．まず，図 4 (a) では辺 E0 の中点に新しい格子点が生成された場合のその影響領域はＰ０からＰ７の８つの頂点で作られる６つの四面体に及び，６つの四面体の内のどれか一つが分割された場合は残りの５つの四面体も同じように分割をしなければクラックが発生してしまう．同様に図 4 (b)，図 4 (c) では，その影響は４つの四面体と８つの四面体に及ぶ。

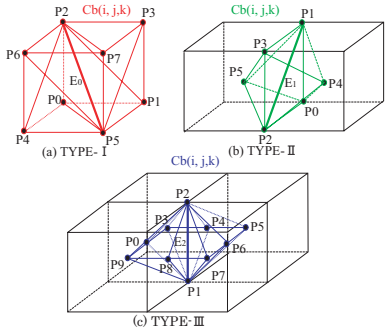

図 4　クラックの影響領域

3.　アダプティブグリッド動的生成

3.1　研究の流れ

本手法のアルゴリズムを説明する．

まず、入力ボリュームデータに対しアダプティブグリッドを用いて，適応的四面体格子表現する．

そして，適応的四面体格子表現された入力ボリュームデータに対してプローブを侵入させる．まず，プローブが四面体に侵入したかどうかの衝突判定を行う．その後，衝突したと判断された四面体に対して分割を行う．

四面体を分割する際に前節で述べたように各々の四面体を独立に分割を行うとクラックを発生する可能性があるので，これを回避するために分割される四面体に隣接する四面体を検索し同様に分割を行う．クラックの発生がなくなるまでこれを再帰的に繰り返す．

クラックの発生の可能性がなくなったら描画を行う．その過程を図 5 に示す．

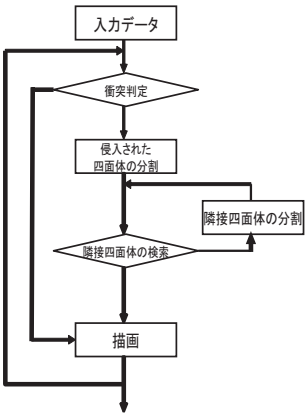

図 5　アルゴリズム

3.2　データ構造

本稿で用いるデータ構造は Triangler Half-Face Data Structure（以下 THFDS）と呼ぶ本研究室で独自のデータ構造を使う．そのデータ構造を図 6 と表 1 に表示する．表 1 に示されているように THFDS は，Point，Edge，Face，Tetrahedron，の 4 つのデータ構造から成り立っている．特に本稿では 3 次元での隣接関係を調べるために, 面に対して注目をおいている．

Point のデータ構造には，

(1)　格子点の (x,y,z) の座標の情報が保持されている．

Edge のデータ構造には，

(1)　線のを構成する両端点の２てんの Point のデータ構造番号が保持されている．

Face のデータ構造には，(注：実際は図 6 の全面（f0～f3）に対して成り立つが，理解しやすいように f0 に注目して書く)

(1)　f0 を構成する 3 点の格子点の Point のデータ構造番号が図 6 に示すような順番に保持されている．(P0 → P2 → P1)

(2)　f0 が構成している Tetrahedron(T0) のデータ構造番号が保持されている．

(3)　f0 が分割されているかどうかの情報を保持している．

(4)　分割で f0 を生成する親の Face のデータ構造番号と f0 が分割で新たに生成する２つの子の Face のデータ構造番号を保持している．

(5)　f0 に対して図 6 に示すような順番と逆向きの

順番 (P0 → P1 → P2) で格子点が保持されている面の Face のデータ構造番号を保持している．

Tetrahedron のデータ構造には，

(1) 四面体を構成する 4 面の Face のデータ構造番号を保持している．(保持する順番は後述する．)

(2) 初期の四面体をレベル 0 とし，分割される度にレベルが 1 つづつ増えていく．

(3) この四面体が分割されているかどうかの情報を保持している．

(4) この四面体がノードとして左か右かの情報を保持している．

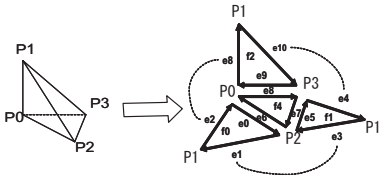

図 6 Triangler Half-Face Data Structure の図

表 1 Triangler Half-Face Data Structure の表

	Triangler Half-Face Data Structure
Point	1. 格子点 (x, y, z)
Edge	1. 辺を構成する Point (2 点)
Face	1. 面を構成する Point (3 点) 2. この面が構成する Tetrahedron 3. 分割判定 flag 4. 親情報と子情報 5. 逆向きの Face
Tetrahedron	1. 四面体を構成する Face（4 面） 2. 分割レベル 3. 分割判定 flag 4. 木構造での左か右か (left or right)

3.3 プローブと四面体の衝突判定

四面体を分割する際に，3 次元デバイスによって操作されるプローブの先端が四面体内部に侵入した場合にその四面体の分割を行うので，四面体内部に侵入したかどうかを調べる衝突判定が必要である．そこで，まず侵入される四面体の体積を求める．四面体の体積は図 7 のようにある場合 $\vec{OA} = (a1, a2, a3)$, $\vec{OB} = (b1, b2, b3)$, $\vec{OC} = (c1, c2, c3)$ とすると，

$$V = \frac{1}{6}|a1b2c3 + a2b3c1 + a3b1c2 - a1b3c2 - a3b2c1 - a2b1c3| \quad (1)$$

の公式で求めることができる．

さらに図 8 のようにプローブの先端（点 P）と四面体の各面の 3 点とを結び，新しい四面体を 4 つ生成する．新しく生成された 4 つの四面体のそれぞれの体積は式 (1) の方法で求めることができる．そして，それぞれ求められた体積の総和と分割される四面体の体積が等しくない時には，プローブは四面体の外部に存在する．逆に，体積の総和と分割される四面体の体積が等しくなる時に，プローブは四面体の内部に存在している．よって，プローブと各面で生成される 4 つの四面体の体積の総和と分割される四面体の体積が等しくなった場合にプローブの先端が侵入したと判定する．

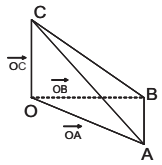

図 7 TYPE RIGHT の四面体

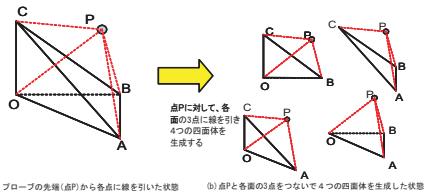

図 8 プローブの先端 (点 P) と各 4 面とで 4 つの四面体を生成

3.4 新しい格子点の生成と四面体の分割

分割される四面体にプローブが侵入した場合，その四面体を分割を行う必要がある．四面体を分割する場合は，図 9 (a) のように四面体を構成する 6 辺の中で最長辺の中点に新しい格子点を生成する．そこで，本稿では四面体の最長辺を検索する時に図 10 のように四面体の形状のパターンは 4 パターンあるが，どのパターンも | 0 | 1 | 2 | 3 | という配列で格子点番号が保持されている．この構造では四面体の最長辺はどのパターンも | 0 | 1 | 2 | 3 | という格子点番号間で 1 と 2 の間に存在している．本手法ではこの性質を利用し，分割を行う際に 1 と 2 の線分の中点に新しい格子点を生成し，分割をする度に最長辺を見つけ出す手間を省略した．

そして，その新しく生成された格子点を利用し図 9 (b) のように 2 つの四面体に分割する．

また，新しい格子点をMとした場合では，分割の際には図 10 のような形で格子点の配列をするようにする．そうすることにより，分割 Level がどの Level になっても，上記の方法で新しい格子点を生成することができる．

3.5 データ構造の更新と追加

3.4 節で示したように，四面体が分割され，新しい

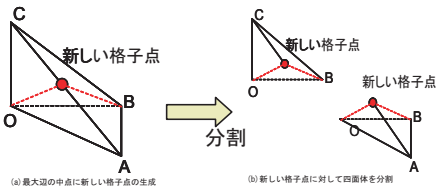

図 9　新しい格子点の生成と四面体の分割

四面体が 2 つ生成された場合には表 1 のデータ構造の更新と追加をしなければならない。

Point のデータ構造
新しく生成された格子点を追加.

Edge のデータ構造
新しく生成された辺を追加.

Face のデータ構造
分割された面に分割 flag と子情報を更新.
新しく生成された面を追加.

Tetrahedral のデータ構造
分割された四面体に分割 flag を更新.
新しく生成された四面体を追加.

表 2　THFDS の更新

	Triangler Half-Face Data Structure
Point	格子点 (x, y, z)
Edge	始点, 終点
Face	1. 面を構成する Point (3 点) 2. この面を構成する Tetrahedron 3. 分割判定 flag　→（分割済みに更新する） 4. 親情報と子情報 →（子情報を更新する） 5. 逆向きの Face
Tetrahedron	1. 四面体を構成する Face（4 面） 2. 分割レベル 3. 分割判定 flag →（分割済みに更新する） 4. 木構造での左か右か (left or right)

3.6　THFDS による隣接四面体の検索

プローブの侵入により動的に四面体が分割していく場合，クラックが発生する可能性がある．後述するクラックの回避のために隣接する四面体の検索を行う必要がある．本手法では分割される四面体に隣接する四面体を検索する方法として THFDS を使用した隣接関係の検索を利用する．前述したデータ構造をそれぞれに保持させているので，これらのデータ構造を利用して隣接する四面体の検索を行う．

ここで，注意すべきなのは THFDS を用いる時に重要なのが面のデータ構造に含まれる逆面情報（表 1 参照）であるということである．この逆面情報というのは前述したように，格子点の配列順番により逆面かどうかを判断しなければならない．そこで，四面体が分割されて，面が新しく生成される度に生成された面の逆面を既存する面集合全体から探し出していたので計算時間が膨大になってしまい．リアルタイムで処理するということができなくなってしまう．そこで，本稿では逆面の設定を面集合全体から探し出すという方法を用いず，効率のより逆面設定を提案する．

まず，本稿で用いているアダプティブグリッドを使用してボリュームデータを適応的四面体構造表現をした場合，四面体の形状は図 10 の鏡面形状を含んだ 4 パターンに分けることができる．それぞれ，4 つの格子点配列によって表現 (図 10 (親)) されているが，分割された時に生成する二つの子の格子点配列は TYPE によってそれぞれ違う (図 10 (子))．これでは，新しく生成された四面体のどの部分がどの位置にあたるのかがわからない．四面体を構成する面が四面体のどの位置に存在するのかがわからなければ逆面を設定するのに不都合である．

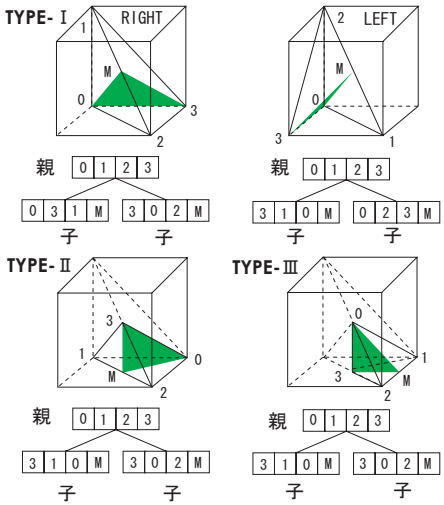

図 10　アダプティブグリッドにおける四面体 TYPE

よってどのパターンでも新しく生成される四面体の面の位置に統一性がとれるように図 11 のように四面体の格子配列から Face データ構造を生成する．図 11 では TYPE　-RIGHT の例をあげている。図 11 のアルファベットで書かれたのがすべての四面体 TYPE での面のデータ構造の生成パターンである．

上記のようにすることにより，まず図 12 のような 3 つの四面体（T1，T2，T3）と隣接する四面体 T0(図

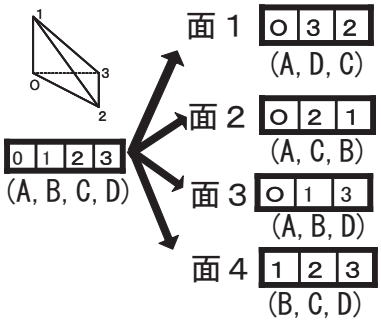

図 11 格子点情報から面情報を生成

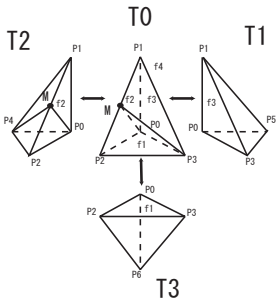

図 12 効率のより逆面設定

10 の TYPE　-RIGHT) が存在する．T0 において面の格子点配列情報は図 11 のように面が構成される．(図 11 面 1 が図 12 f1 に対応) そして，T0 が辺 P1P2 の中点に新しい格子点 M が生成され分割された場合に新しい四面体が 2 つ生成される．本稿ではその分割時に逆面を設定する手法を行っている．まず，T0 が分割されると，図 13 のように分割される．

まず図 13 (Ta) と (Tb) 両方の面 3 は TO が分割されたことにより新しく生成される面であるので，T0 と T1 の面 3 がお互い逆面同士であるので，それを設定する．

次に，図 13 (Ta) の面 2 は T0 が分割された時に影響を受けない面である．図 12 の f3 面がそれにあたる．よって，分割された T0 の f3 面 (P0 → P1 → P3) の逆面情報 (図 12 の T1 の f3 面 (P0 → P3 → P1)) を，そのまま図 13 (Ta) の面 2 の逆面として設定してやればよい．同じように図 13 (Tb) の面 2 も親の四面体である T0 の f0 面の逆面 (T1 の f0 面 (P0 → P2 → P3)) を逆面とせっていすればよい．

そして，図 13 (Ta) の面 1 は T0 が分割された時に T0 の f2 が分割されて生成された面である．この面の逆面の設定は，まず図 12 で T0 が分割されると後述のクラックの回避のために，M を共有する四面体が同時に分割されるので図 12 T2 も分割される．ということは，図 13 (Ta) の面 1 の逆面は，親の面である TO の f2 の逆面が分割されたどちらかになる．(図 12 の (P0 → M → P2) か (P0 → P1 → M)) よって，分割された T0 の f2 面 (P0 → P2 → P1) の逆面 (図 12 の T2 の f2 面 (P0 → P1 → P2)) の子のどちらかが図 13 (Ta) の面 1 の逆面となる．

これらをまとめたものを図 14 に示す．

この手法をとることにより，分割時に逆面を設定することができるので新しい面が生成された時に既存の面全てを新しい面の逆面かどうかを比較する必要がなくなり，計算量の大幅な削減が可能となる．

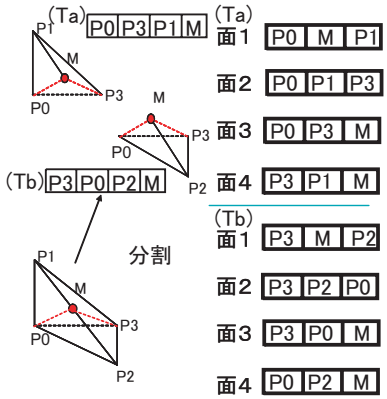

図 13 分割時の新しい面情報の生成

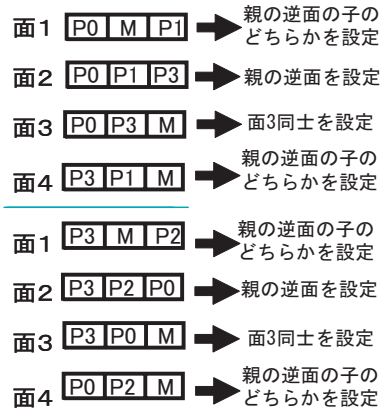

図 14 逆面の設定の図

これにより Face のデータ構造には逆面情報が保持されているので，それぞれの面の逆面を知ることができる．さらに，逆面のデータ構造で構成する四面体の Tetrahedron のデータ構造番号を得ることができる．よって，逆面の構成している四面体を知ることができる．これにより分割された四面体に隣接する四面体を THFDS をすることができる．

3.7 再帰的検索よりクラック回避

プローブの侵入により，分割された四面体に隣接する四面体は上記の方法で検索することができる．よって，その手法を使い隣接する四面体を検索をし再帰的に分割をしてやることによりクラックを回避する．

図 15 は 3 次元では視覚的に理解しにくいので，3 次元物体を真横から見たと仮定し，2 次元での説明を行う．まず，図 15 (1) のように，四面体に対してプローブが侵入したとすると，侵入された四面体に注目をし分割を行う．(図 15 (2)) さらに，小さな四面体を生成をするために，プローブが侵入した四面体を分割を行う．(図 15 (3a)) そうすると，図のように四面体間での不整合生じクラックを発生させてしまう．

これを，回避するために本手法では四面体に対してプローブが侵入したとすると，侵入された四面体に注目し分割を行い．(図 15 (2)) その時に，四面体の最長辺に新しい格子点が生成されているので，最長辺を共有している隣接する四面体を検索し分割する必要がある．最長辺を共有している四面体を検索するには，前述の図 11 によりどの面がどの位置にいるかは判定できる．それにより，最長辺を持つ面情報は図 11 図の面 2 と面 3 になるので，THFDS で二つの面の隣接する四面体を検索することにより最長辺を共有する隣接する四面体を検索する．そして，その四面体に対して注目をしてやり同じように再帰的に隣接する四面体を検索を行い分割することでクラックを回避する．(図 15 (3b)

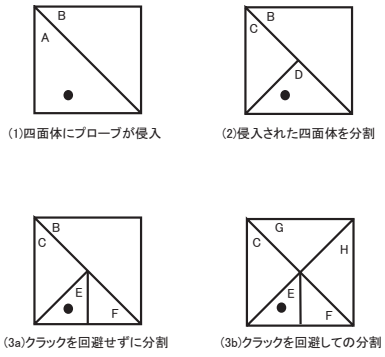

図 15 クラックが発生する状況

より詳しく再帰的検索の部分を述べると最長辺を共有している面は図 10 のどの TYPE でも必ず図 11 面 2 と面 3 になる．そこで，まず面 2 の方を注目をして (図 18 ( ))，その逆面の状況を調べる．(図 18 ( ))

逆面がないというのは，注目している面が一番外側にいるなどの時であり，その場合は面 3 の方に注目をしにいく (図 18 (a))．また，逆面がすでに分割されているというのは逆面が構成している四面体がすでに同じレベルまで分割されているということであり，その場合は面 3 の方に注目をしにいく (図 18 (b))，その逆に未分割は逆面が構成している四面体がまだそのレベルまで分割されてないことを示し，その場合はその四面体を注目をして分割する (図 18 (c))．

同様の事を面 3 にもしてやり，これを分割された四面体の最長辺を共有する隣接する四面体がすべて分割されるまで再帰的に行う．

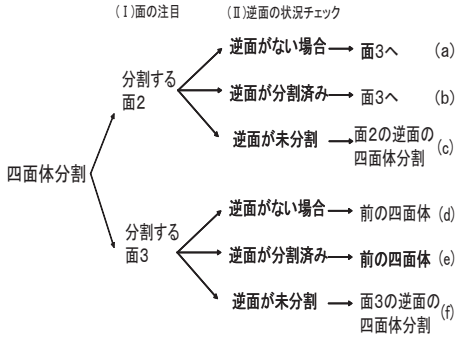

図 16 分割時に分割される面

4. 検 証 実 験

4.1 実 験 条 件

3 章で示したボリュームデータの動的アダプティブグリッド生成アルゴリズムを実装した．図 17 で示すように立方体格子のボリュームデータに対してプローブを侵入させて，そのままボリュームデータを貫通するまでプローブを前進させる．また，比較実験として THFDS を用いずに，四面体の最大辺を共有する四面体をすべての四面体から探し出してクラックを生じずに動的にアダプティブグリッド生成する手法との計算スピードの比較を行う。実験環境として PC 性能は Pentium　の CPU3.4GH ｚのメモリ 1.0GB を使用する．また 3 次元デバイスには PHANToM Desktop (Sensable 社) をしよう

4.2 実 験 結 果

以下の表 3 には，全検索の方法と RHFDS での方法の比較結果を示す．示す結果には，分割レベル，新しく作られた四面体の数 (個)，その時の既存の四面体の数 (個)，時間 (S) の 4 つを示す．

4.3 考　　察

本手法で行った場合，リアルタイムでアダプティブグリッドを生成することができた (図 18). 次に比較

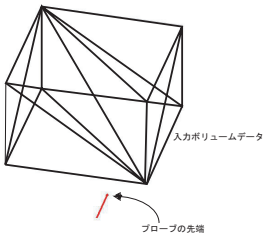

図 17 実験

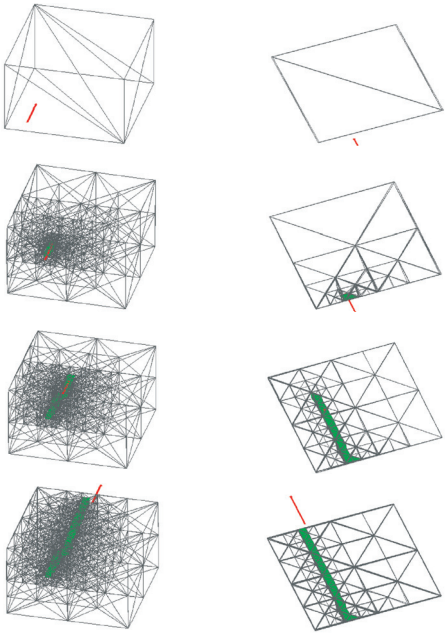

図 18 分割時に分割される面

表 3 全検索と THFDS との比較結果

			全検索	RHFDS
分割レベル	既存の数	生成された数	処理時間	処理時間
5	約 300	約 10	0.031	0.031
10	約 4000	約 120	0.031	0.031
15	約 11000	約 220	0.094	0.063
20	約 24000	約 630	1.047	0.112
30	約 35000	約 1050	2.203	0.803
50	約 59000	約 1900	8.704	0.910

実験では，全検索の方法では，四面体の数が少ない時には THFDS とはそれほど顕著な差は表れなかったが，四面体の数が増えていくにつれ，その計算時間が増えていった．分割レベルが 20 を超えると一度の処理に 1.047 秒と一秒を超えてしまう．こうなうると、リアルタイムでの処理に不都合が生じてしまう．逆に THFDS のほうでは四面体の数が増えてもそれ程の差はでなかったが，新しく生成される四面体が多いと計算時間がかかってしまう．しかし，分割レベルが 50 というのは実際では必要ないレベルであるので，この時に 0.910 秒という処理時間は実際の動的アダプティブグリッド生成には問題ないと考えられる．

5. ま と め

アダプティブグリッドにより適応的四面体格子表現されたボリュームデータを，THFDS を用いてクラックを発生させることなくリアルタイムで動的に分割する手法を提案した．今後は CT や MRI で得られたボリュームデータを用いて動的分割を行う予定である．さらに，四面体の点や辺に質量や応力を与えることで物理法則に基づいた手術シミュレーションの実現を目指す．また，反力により力のかえってくる 3 次元デバイスである PHANToM を用いて力を返して行きたい．

参 考 文 献

1) S.P.DiMaio and S.E.Salcudean, ”Needle Insertion Modeling and Simulation”, IEEE Int. Conf. on Robotics and Automation, 2002.
2) H.-W.Nienhuys,AF van der Stappen”A Comutationaltechnique for interactive needle insertions in 3D nonlinear material, Proc. of The IEEE International Conference on Robotics and Automation(2004),pp.2061-2067.
3) Maubach J.M., ”Local bisection refinement for n-simplicail grids generated by reflection”, SIAM. J. Sci. Compt., vol.16, pp.210–27, 1995.
4) H.T Tanaka and F.Kishino,”Adaptive mesh generation for surface reconstruction:Parallel hierarchical triangulation without cracks,”Proc. IEEE 10th International Conference on Pattern Recognition,pp.88–94,1993.
5) MC Rivera and P. Inostroza, ”A discussion on mixed(longest side midpoint insertion) delaunay techniques for the triangulation for the triangulation refinement problem,” Proceedings 4th International Meshing Roundtable. Albuquerque,pp.335–346,1995.
6) H.Lopes and G.Tavares, ”Structural operators for modeling 3-manifolds”, In Proc. of the 4th ACM SIGGRAPH Symposium on Solid Modeling and Its Applications, ACM Press, pp.10–18, 1997.
7) Y.Takama, A.Kimura, H.T.Tanaka:”Tetrahedral Adaptive Grid for Parallel Hierarchical Tetrahedrization”, EUROGRAPHICS Workshop on Multimedia, pp125-133, 2004.