PC グラフィックスハードウェアを利用した 6 面体メッシュ細分割モデリング手法

藤森智行　宝田洋佑　鈴木宏正　木村文彦
東京大学工学系研究科
E-mail: {fujimori, takarada, suzuki, kimura}@cim.pe.u-tokyo.ac.jp

近年、活発に研究されているメッシュの細分割手法は、6 面体メッシュにも適用の範囲を広げてきている。本研究では、6 面体メッシュ細分割を用いて、物体内部を操作可能なモデリング手法を実現するために、2 つの手法を提案する。最初に、6 面体メッシュ細分割を PC 用グラフィックスハードウェア上のシェーダとして実現する手法を提案する。次に、細分割結果を表示用のボリュームデータに変換し、ボリュームレンダリングを用いて表示を行う手法を提案する。これらの手法は、ベースメッシュの頂点に対する編集と細分割結果の表示のインタラクティビティを向上することを目的とする。また、手法の実装および実験を行って有用性を確かめるとともに、モデリングシステムの試作を行った。

Volumetric Subdivision Modeling on PC Graphics Hardware

TOMOYUKI FUJIMORI　YOSUKE TAKARADA
HIROMASA SUZUKI　FUMIHIKO KIMURA
Graduate School of Engineering, The University of Tokyo
E-mail: {fujimori, takarada, suzuki, kimura}@cim.pe.u-tokyo.ac.jp

The scope of the subdivision surface scheme is extended also in hexahederal meshes. In this paper, we propose two new methods for applying the volumetric subdivision scheme to modeling. The first method realizes multi-linear cell averaging (MLCA) subdivision using shaders on PC graphics hadrdware. The second method converts subdivided hexahedral meshes into regular volumetric data so that we can use volume rendering techniques for displaying subdivision results. We implement these two methods and a prototype modeling system to show interactive volume modeling can be performed on PC graphics hardware.

1　はじめに

近年、活発に研究されているメッシュの細分割手法は、6 面体メッシュにも適用の範囲を広げてきている [1, 2]。これらの 3 次元空間に拡張された細分割手法は、一般に Volumetric Subdivision 手法と呼ばれる。本研究では、この手法をベースとして、物体内部を操作可能なモデリング手法をインタラクティブに実現するための要素技術を提案することを目的とする。ここで、物体内部を操作可能なモデリング手法とは、細分割を適用する前のベースメッシュの頂点を移動・変更することによって、物体表面のみならず、物体内部の構造も決定可能なモデリング手法である。

Volumetric Subdivision 手法は、一般の細分割手法と良く似た特徴を持つ。そのうち物体内部の操作を行うようなモデリングで使用することを考えると、以下の特徴が注目される。まず、境界を陽に定義可能であるという特徴がある。これによって、物体表面と物体内部を同一の形式で表現することができる。また、個々の頂点の影響範囲が比較的小さい、多重解像度表現を行うことができるという特徴も、モデリングに役立つと考えられる。

逆に欠点としては、3 次元空間の細分割を行うため、データ量の増大が著しく、計算コストが大きいという問題が挙げられる。また、細分割の結果として得られる 6 面体メッシュの表示手法が確立されていないという問題がある。

本論文では、これらの問題を PC 用グラフィックスハードウェア (以下、ハードウェア) を用いて解決する手法を示す。前者の問題に関しては、Volumetric Subdivision 手法の特徴に着目して、処理の負荷を CPU とハードウェアに分散して細分割を行う手法を提案する。後者の問題に関しては、細分割の結果として得られた 6 面体メッシュをボリュームデータに変換し、これに対してボリュームレンダリングを行う手法を提案する。また、以上の手法をハードウェア上に実装し、本手法の有効性を確かめると同時に、これらを用いたモデリングシステムを試作する。

以下、2節で既存の Volumetric Subdivision 手法とモデリングで使用した場合の問題について示す。この問題を解決するために、3節でハードウェアを援用した細分割手法を提案し、4節で 6 面体メッシュをボリュームデータに変換する手法を提案する。5節で実験及び結果について、6節で結論を述べる。

2 Volumetric Subdivision 手法

4 辺形メッシュの代表的な細分割手法である Catmull-Clark 細分割手法を、3 次元空間に拡張した Volumetric Subdivision 手法の代表的なものとして、MacCracken らの 3 次元格子の細分割手法 [1]、Bajaj らの 6 面体メッシュ細分割手法 [2] がある。

どちらの手法を用いても同一の細分割結果を得られるが、Bajaj らの手法は MacCracken らの手法に比べて細分割ルールが簡単であるという特徴を持ち、ハードウェア実装との親和性が高い。この事から、本研究では Bajaj らの手法を採用した。彼らが提案した手法は、一般的な細分割における Averaging 手法 [3] を拡張した MLCA (Multi-Linear Cell Averaging) 手法である。

以降では MLCA 手法とモデリングに用いる場合の問題点を示す。

2.1 MLCA (Multi-Linear Cell Averaging) 手法

MLCA 手法は、細分割処理を以下の 2 段階の処理に分割する。

1. *Bi-linear Subdivision*

2. *Averaging Operation*

処理 1. の *Bi-linear Subdivision* は、単純な線形平均を用いて、6 面体の細分割を行う処理である。

図 1 に示すような 6 面体 $\boldsymbol{ABCDEFGH}$ が存在する時、細分割によって挿入される頂点は式 (1) のように決定される。ここで、$\boldsymbol{EP}$ は稜線上に挿入される頂点、$\boldsymbol{FP}$ は面上に挿入される頂点、$\boldsymbol{CP}$ は 6 面体内部に挿入される頂点である。

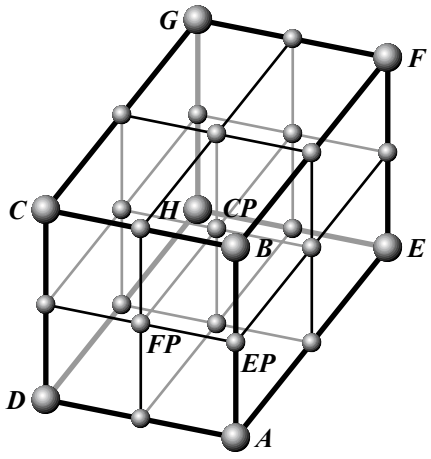


図 1: *Bi-linear Subdivision*

$$
\begin{aligned}
\boldsymbol{EP} &= \frac{\boldsymbol{A}+\boldsymbol{B}}{2} \\
\boldsymbol{FP} &= \frac{\boldsymbol{A}+\boldsymbol{B}+\boldsymbol{C}+\boldsymbol{D}}{4} \\
\boldsymbol{CP} &= \frac{\boldsymbol{A}+\boldsymbol{B}+\cdots+\boldsymbol{G}+\boldsymbol{H}}{8}
\end{aligned}
\tag{1}
$$

次に、処理 2. の *Averaging Operation* について説明する。*Averaging Operation* は、処理 1. で得られたそれぞれの頂点の座標を、頂点を共有する 6 面体の重心の平均で置き換える処理である。

図 2 にこの処理の擬似コード表現を示す。最初のループで 6 面体の重心と、頂点が何個の 6 面体に共有されているかを調べる。次のループで頂点の座標を初期化し、最後のループで頂点の座標を 6 面体の重心の平均に設定している。

```
cells ← 6 面体メッシュ
for all c ∈ cells do
  for all v ∈ c do {6 面体の各頂点について}
    c.center ← c.center + v/8
  end for
  for all v ∈ c do {6 面体の各頂点について}
    v.count ← v.count + 1
  end for
end for
for all v ∈ cells do {全ての頂点について}
  v ← 0
end for
for all c ∈ cells do
  for all v ∈ c do
    v ← v + c.center/v.count
  end for
end for
```

図 2: *Averaging Operation* の擬似コード表現

以上が、MLCA 手法の概要であるが、実際の細分割処理では境界に対する処理を行う必要がある。一般の細分割では、曲面の外周および内周に存在する頂点と稜線に特別な細分割ルール (マスク) を適用するが、6 面体メッシュ細分割では、頂点と稜線に加えて、曲面もまた境界にすることができる。

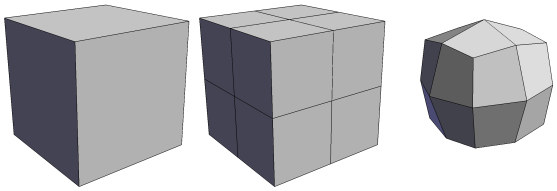
境界の処理は、以下の手順で行われる。

1. MLCA 手法を用いた 6 面体メッシュ細分割

2. 境界となる曲面に対して通常の Catmull-Clark 細分割手法を適用後、処理 1. によって得られた同じ頂点を上書き

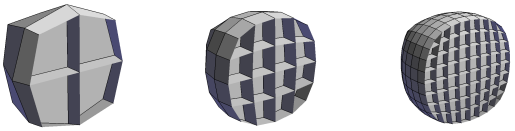
図 3, 4 に、立方体に対して MLCA 手法を適用した例を示す。ここで、立方体の表面は境界として設定されている。図 3 (A) が初期メッシュを表しており、(B), (C) はそれぞれ、*Bi-linear Subdivision* と *Averaging Operation* を適用した結果を表している。図 4 は、

細分割された 6 面体メッシュの内部の様子を、クリップ面を用いて表示したものである。



(A) (B) (C)

図 3: 立方体に対する MLCA 手法の適用



1 回細分割 2 回細分割 3 回細分割

図 4: 複数回の MLCA 手法の適用

2.2 MLCA 手法の問題点

本節では、モデリングに MLCA 手法を応用する場合の問題点を考察する。

第 1 の問題として、 MLCA 手法自体は単純な手法であるが、 6 面体の数が指数関数的に増加する (細分割 1 回あたり、 6 面体の数が 8 倍) ため、複数回の細分割を行おうとすると計算コストが大きくなってしまうという問題がある。本研究では、次節で示すように 6 面体同士がレギュラーに接続している場合の細分割処理は、ハードウェアとの親和性が高いことに着目して、ハードウェアを援用した細分割という解決策を提案する。

第 2 の問題として、表示方法が確立されていないという問題がある。一般的なポリゴンレンダリング手法で表示することはできるが、この場合、内部構造を視覚化することが難しい。本研究では、この問題に対して、ボリュームレンダリングによる表示という解決案を示す。これによって、面数の指数関数的な増加 (細分割 1 回あたり、面数が約 8 倍) による表示速度の低下を抑えるとともに、内部構造の表示や等値面表示といったボリュームレンダリングの機能を用いることができる。

3 ハードウェアを援用した細分割

Loop 細分割手法を始めとする細分割手法のハードウェア実装が活発に研究されている [4, 5]。これらの研究は、 CPU - ハードウェア間の帯域を消費せずに、滑らかな曲面を表示するという目的で共通している。

これに対して、本研究では表示を最終の目的とせず、 6 面体メッシュに特化して、細分割の負荷を CPU とハードウェアに効率良く分散するシステムを提案する。図 5 にシステムの概要を示す。本システムは、浮動小数点バッファを利用可能な次世代のハードウェア [6] を用いて、細分割をフラグメント単位の演算として処理するものである。

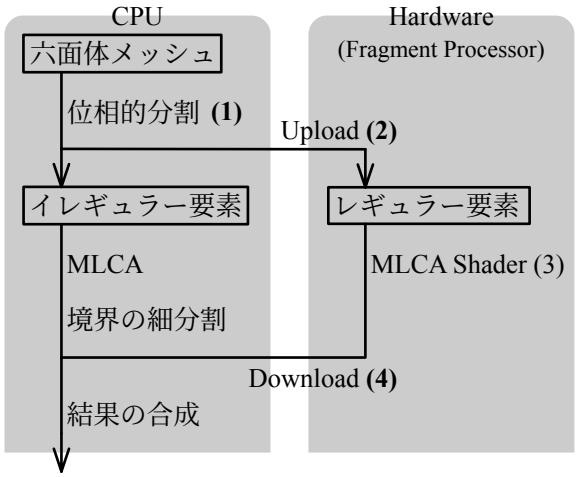


図 5: ハードウェアを援用した細分割システムの概要

システムは 6 面体メッシュを入力として与えられ、これを図 5 (1) のフェイズで、 CPU で処理する領域とハードウェアで処理する領域に分割する。ハードウェアで処理する領域は、境界部分を除いて、レギュラーに接続した 6 面体の集合であり、これをレギュラー要素と呼ぶことにする。レギュラーに接続した 6 面体とは、各頂点が 8 個の 6 面体に共有され、各稜線が 4 個の 6 面体に共有されるような 6 面体であるとする。

レギュラー要素は格子空間の変形によって表現することができるので、これを浮動小数点フォーマットの 3 次元テクスチャに格納する。それぞれのテクセルが頂点に対応し、テクセルの RGB 値は頂点の座標 (x, y, z) を表現する。テクセル同士の隣接関係は、レギュラー要素内の頂点同士の隣接関係を表現する。

図 5 (2) のフェイズでは、この 3 次元テクスチャを CPU 側からハードウェアにアップロードする。これを入力として (3) の MLCA シェーダ (3.1節に後述) をハードウェア上で実行し、細分割処理を行う。その後、 (4) のフェイズで、処理結果を画像としてハードウェアから CPU 側にダウンロードする。

ハードウェア上での細分割処理と平行して、 CPU 側ではイレギュラーな頂点と境界に対する細分割処理を行う。最終的な処理結果はこれらをマージすることで得られる。

3.1 MLCA シェーダ

本節では、2.1節で紹介した MLCA 手法を、ハードウェア上にシェーダとして実現するための方法を示

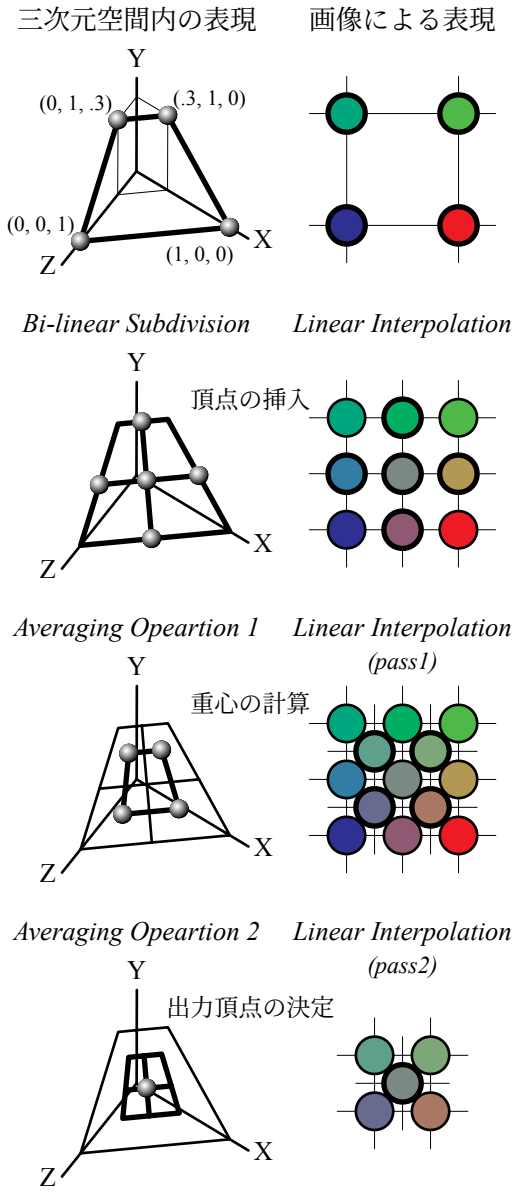


図 6: MLCA シェーダの 4 辺形への適用

す。基本的なアイディアは、画像の線形補間フィルタを複数回適用して、テクセル色で表現された頂点座標の細分割処理を行うというものである。図 6 に、4 辺形に MLCA シェーダを適用した、2 次元の場合の例を示す。

MLCA 手法の第 1 段階の処理である *Bi-linear Subdivision* を実現するために、アップサンプリング処理を行う。アップサンプリング処理は、入力の 3 次元テクスチャのサイズが $l \times m \times n$ である時、出力結果のサイズが $(2l-1) \times (2m-1) \times (2n-1)$ になるようにハードウェアの出力バッファサイズを設定して行う。アップサンプリング時のフィルタとして線形補間フィルタを使用すれば、*Bi-linear Subdivision* と等価な結果が得られる。

MLCA 手法の第 2 段階の処理である *Averaging Operation* を実現するためには、任意の頂点を共有する 8 個の 6 面体の重心を求め、その平均を求める必要がある。本システムでは、この処理をマルチパスに分解する。最初のパスで 6 面体の重心を計算し、ハードウェアのワークバッファに保存する。次のパスでこのワークバッファを読み込み、重心の平均を計算し、最終的な頂点を求める。どちらのパスの処理も、線形補間フィルタを適用することで実現することができる。

4 細分割された 6 面体メッシュの表示

本研究では、細分割された 6 面体メッシュの表示法として、6 面体メッシュから表示用ボリュームデータを作成し、これに対してボリュームレンダリングを行うシステムを提案する。

細分割された 6 面体メッシュの表示法のなかで、内部構造を表現可能な、最も自然かつ単純な手法は、Ray Casting 法であると思われる。Ray Casting 法では、描画面の各画素から視線方向に伸びる光線に沿って積分を行うが、本研究で提案するシステムはこれを離散化して近似することになる。

本システムの利点は、既存のボリュームレンダリング手法を適用可能である点である。これにより、インタラクティブな速度で内部構造も含めた表示を行うことができる。以降では、細分割された 6 面体メッシュを、ハードウェアを用いてサンプリングし、表示用ボリュームデータを作成する手法について述べる。

4.1 表示用ボリュームデータの作成

本手法では、表示用ボリュームデータとして、直交した格子上に要素が等間隔に配置されたボリュームデータを扱う。このようなボリュームデータは断面画像の重ね合わせとして表現することができる。即ち、必要な枚数の断面画像を作成することができれば、ボリュームデータを作成することができると言える。本節では、主に、ハードウェアを用いて断面画像を作成する手法について述べる。

まず、断面画像の定式化を行う。平面 P と物体 B が与えられている時、平面 P による物体 B の断面は、P 上の点のうち B の内部に存在する点の集合である。本手法における断面画像の作成は以下の手順で行う。

1. 離散化した P 上の点 X について、物体 B の内部に存在するかどうかを判定
2. 点 X が物体 B の内部に存在すれば、近傍の値から X の値を決定

この手順をシェーダとしてハードウェア上に実現する。シェーダは、ポリゴンとして表現される 6 面体メッシュを入力として、断面画像を出力する。

図 7 に示すように、描画面から距離 z 離れた場所に平面 P の位置を設定するとき、手順 1. は Z 値の比較演算によって実現することができる。描画面から視線

方向に伸びる光線と交差する、任意のポリゴン p 上の点を X_p とする。この点 X_p の描画面からの距離を z_p とする時、平面 P 上の点 X に近接するポリゴンを次のように求める。

- $z_{p_0} < z$ かつ $|z_{p_0} - z|$ を最小化するようなポリゴン p_0 は、平面 P よりも描画面に近いポリゴンのなかで、点 X に最も近接している。
- $z_{p_1} > z$ かつ $|z_{p_1} - z|$ を最小化するようなポリゴン p_1 は、平面 P よりも描画面から遠いポリゴンのなかで、点 X に最も近接している。

点 X に近接するポリゴン p_0, p_1 が求まれば、これらのポリゴンを面に持つ 6 面体の内部に点 X が存在する。この 6 面体が、手順 1. の物体 B に当たる。

次に、手順 2. で点 X の値を決定する。ここでは単純に、ポリゴン p_0, p_1 と光線の交差点 X_{p_0}, X_{p_1} を用いて、式 (2) に示すような線形補間を行う。

$$X = \frac{|z_{p_1} - z| X_{p_0} + |z_{p_0} - z| X_{p_1}}{z_{p_1} - z_{p_0}} \qquad (2)$$

この線形補間は、点 X を内部に持つ 6 面体の正確な線形補間ではない。しかし、充分細かい細分割が行われていれば、表示においては誤差は無視できるぐらいに小さくなると考えられる。また、点 X の値を精密に決定したい場合には、速度を犠牲にして、3 重線形補間カーネルやスプラインフィルタカーネルを適用することができる。

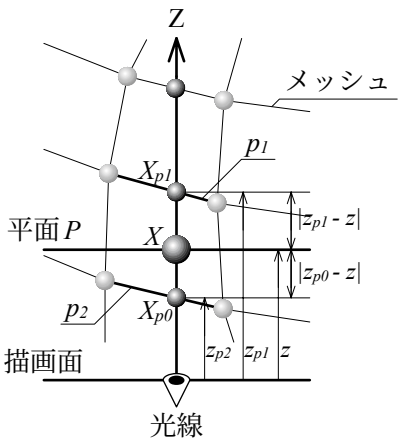


図 7: 断面画像の作成

5 実験および結果

NVIDIA GeForce3 Ti 500 を搭載した Pentium Ⅲ, Main Memory 512MB の PC を用いて実験を行った。実験で使用した GeForce3 Ti 500 では、浮動小数点バッファを使用することができないので、NVIDIA 社が提供している次世代ハードウェアのエミュレーションモードを使用して、モデリングシステムの試作を行った。ただし、計算時間の計測には、整数演算で近似した実装を用いている。

試作したモデリングシステムには、6 面体の頂点を直接編集するという低レベルのインタフェースを実装した。後述するように、高解像度の表示用ボリュームデータの作成には時間がかかるという問題があり、インタラクティビティを確保するために、多重解像度での表示を実装する必要があった。

図 8 に本手法を用いて、4 回細分割を行った結果を示す。右下の図では、閾値を用いて、特定の範囲の値を持つボクセルを除外している。ボリュームレンダリングを用いているため、内部構造が把握できるようになっている。

図 9 に、本研究で提案したハードウェアを援用した細分割手法の計算時間をまとめたグラフを示す。ベースメッシュの 6 面体数にもよるが、1 秒程度で 4 回細分割を行うことができた。MLCA のソフトウェア実装で同様の実験をした場合、1 分程度かかったので、本手法によって大幅に高速化できたと言える。

図 10 に、表示用ボリュームデータの作成に要した計算時間をまとめたグラフを示す。6 面体数とボリュームデータの解像度の双方によって、計算時間が増大している。細分割回数が少ない場合は、数秒以内でボリュームデータを作成することができたが、細分割回数が多くなってくると、数 10 秒から 1 分程度かかった。

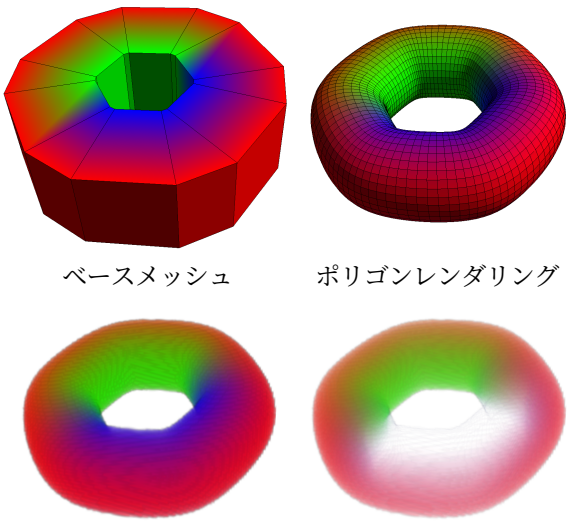


図 8: 本手法の適用結果: 上段: 錐台, 下段: トーラス

6 結論

PC 用グラフィックスハードウェアの機能を利用して、物体内部を操作可能なモデリング技術を実現する

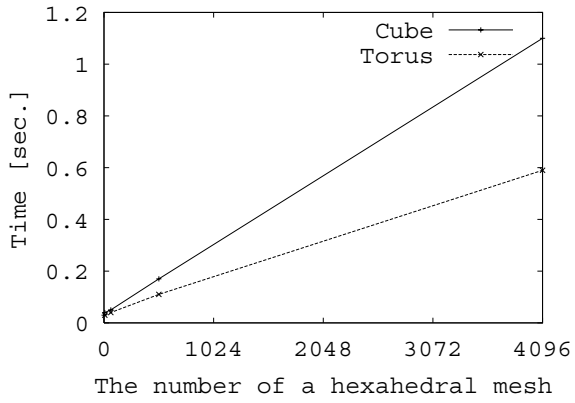


図 9: ハードウェアを援用した細分割手法の計算時間

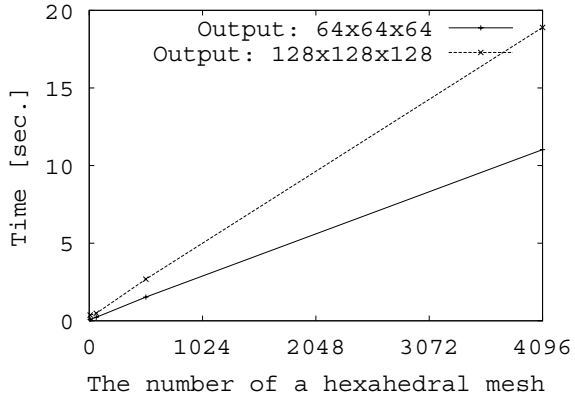


図 10: ボリュームデータの作成に要した計算時間

ための要素技術を提案した。

第一に提案した手法は、6 面体メッシュ細分割をハードウェアを援用して行う手法であり、これにより細分割処理速度の改善が見られた。第二に提案した手法は、細分割の結果として得られる 6 面体メッシュを、ハードウェアを用いてボリュームデータに変換する手法である。このボリュームデータを用いて表示を行うことで、内部構造も含めたインタラクティブな表示を行えるようになった。

以上より、本研究では、6 面体メッシュ細分割をモデリングに応用する際の速度的な問題・表示の問題に対するひとつの解決案を示すことができたと思われる。

しかし、表示用ボリュームデータ作成にまだ時間がかかっており、速度的な改善が今後の課題である。また、モデリングの問題としては、頂点単位の編集操作よりも高レベルな編集操作なども考えていきたい。

参考文献

[1] Ron MacCracken and Kenneth I. Joy. Free-form deformations with lattices of arbitrary topology. In *Proceedings of ACM SIGGRAPH 1996*, pp. 181–188. ACM Press / ACM SIGGRAPH / Addison Wesley Longman, August 1996.

[2] Chandrajit Bajaj, Scott Schaefer, Joe Warren, and Guoliang Xu. A subdivision scheme for hexahedral meshes. *The Visual Computer*, Vol. 18, No. 5/6, pp. 343–356, 2002.

[3] Joe Warren and Henrik Weimer. Averaging schmes for polyhedral meshes. In Diane D. Cerra, editor, *Subdivision Methods for Geometric Design: A Constructive Approach*, chapter 7, pp. 198–238. Morgan Kaufmann Publishers, 2002.

[4] Stephan Bischoff, Leif P. Kobbelt, and Hans-Peter Seidel. Towards hardware implementation of loop subdivision. In *2000 SIGGRAPH / Eurographics Workshop on Graphics Hardware*, pp. 41–50. ACM SIGGRAPH / Eurographics / ACM Press, August 2000.

[5] M. Bóo, M. Amor, M. Doggett, J. Hirche, and W. Strasser. Hardware support for adaptive subdivision surface rendering. In *2001 SIGGRAPH / Eurographics Workshop on Graphics Hardware*, pp. 33–40. ACM SIGGRAPH / Eurographics / ACM Press, August 2001.

[6] Marc Olano, Chas Boyd, Bill Mark, Michael McCool, Jason L. Mitchell, and Randi Rost. *State of the Art in Hardware Shading*. SIGGRAPH 2002 Cource 17, July 2002.