マルチコアプロセッサ向けに並列化した Deformed Marching Cubes法とその評価

江坂 理† 藤本 敬介† 中山 泰一†
†電気通信大学情報工学科

1 はじめに

陰関数からポリゴンメッシュを生成する手法としては，Marching Cubes 法（MC 法）がよく知られている．MC 法では，空間を格子状に区切り，各格子点に対して零値面との内外判定を行うことでメッシュを生成する．そのため，格子の幅よりも薄い部位が欠損する場合がある．格子の間隔を狭くすれば精度は上がるが，格子の間隔を $1/n$ 倍にすると，格子の数は n^3 倍になり，それに伴って計算時間も n^3 倍になる．この問題に対しては，格子点の位置を陰関数の形状に応じて移動させる Deformed Marching Cubes 法（DMC 法）[1] が有効である．DMC 法は処理どうしの独立性が高く，並列化による計算速度の向上が見込める．そこで本研究では，DMC 法をマルチコアプロセッサ向けに並列化し，その性能を評価する．

2 関連研究

2.1 Marching Cubes 法

Lorenson らは，ボリュームデータからメッシュを生成する手法として，MC 法を提案した [2]．MC 法において，ボリュームデータの代わりに陰関数を用いると，陰関数の零値面をメッシュによって表せる．その手順としては，空間を格子状に区切り，各格子に対して次の 1.〜3. の処理を行う．

1. 8 個ある頂点のそれぞれについて，その点における陰関数の値の符号から，その点が零値面の外にあるか内にあるかを判定する．
2. 12 本ある辺のそれぞれについて，その辺の両端が零値面を挟んで異なる側にあるかを判定し，そうであればその辺と零値面が交わる点を線形補間によって求める．
3. 1. の内外判定で決まるパターンに応じて 2. で求めた点を接続し，メッシュを生成する．8 個の頂点が零値面の外か内かの 2 通りの状態を持つので，パターンは 2^8 通りあるが，幾何学的に対象なものを同一視すれば，図 1 の 15 パターンに帰着する．

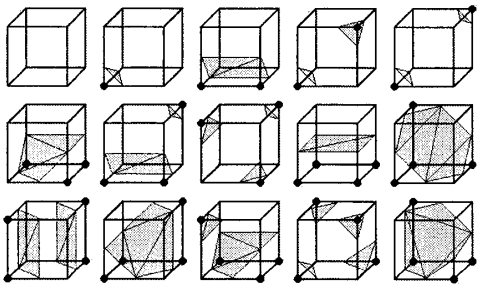

図 1: メッシュの生成パターン

2.2 Deformed Marching Cubes 法

DMC 法は，MC 法に格子点の移動という操作を加えることで，薄い部位の再現性を向上させた．DMC 法では，格子点の周辺に格子状にサンプル点を設け，零値面の内部にあるサンプル点の平均座標を求め，そこに格子点を移動する（図 2）．これにより，格子内に 1 つの部位しかない場合においては，サンプル点の間隔よりも厚い部位の非欠損性が保証される．格子点間の順序関係を壊さない程度の移動なので，移動後は MC 法と同様の処理を行えば良い．

3 設計

格子点を移動すると，その点を共有する 8 個の格子への処理に影響が出るが，点の移動先が一意に決まることから，各格子への処理は独立に行える．そこで，空間全体を多数の小さいブロックに分割して，各プロセッサコアに周期的に割り振る．これによって，一度に処

Evaluation of Deformed Marching Cubes algorithm paralellized for multi-core processors
Masaru ESAKA†, Keisuke FUJIMOTO†and Yasuichi NAKAYAMA
†Department of Computer Science, The University of Electro-Communications
182-8585, Chofu, Japan

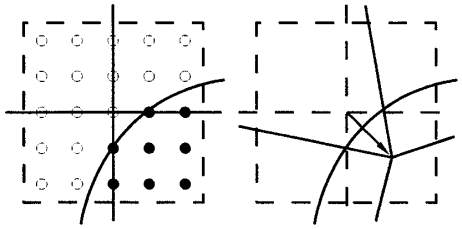

図 2: 格子点の移動

理するブロックが小さくなり，各コアは格子点の移動先などを保存するのに十分なメモリを確保できる．また，どのコアも空間的にばらばらな場所を処理することになり，負荷が分散されることが期待できる．

ブロックの処理と格子点の周辺の探索は，区間演算 [3] を用いることで効率化を図る．区間演算とは，被演算区間に属する任意の数どうしの演算結果を包含する区間を求める操作である．この場合，被演算区間はブロックや格子点の周辺，演算は陰関数の値を求めることである．区間演算によって得られる区間が 0 を含まないならば，被演算区間に関数の零点がないことが分かる．これによって，処理する必要がないブロックや，移動させる必要がない格子点を，ある程度発見できる．

4 実験

マルチコアプロセッサとして，PlayStation3 に搭載されている Cell Broadband Engine を使用した．格子の数は 512^3，格子の間隔は 1.0，格子点の周辺に設けるサンプル点の間隔は格子の間隔の 1/6，ブロックの大きさは格子の数にして 15^3 個分に設定した．また，関数 $f(x,y,z)$ を，

$$\begin{aligned} f(x,y,z) &= \max\{f_0, f_1, y\} \\ f_0 &= x^2+y^2+z^2-200^2 \\ f_1 &= 199.5^2-x^2-y^2-z^2 \end{aligned}$$

と定義した．以上の条件の下で $f(x,y,z)$ の零値面を表すメッシュの生成を行った．使用するコアの数が 1 から 6 までの場合の実行時間を表 1，図 3 に，6 コア使用時におけるコアごとの実行時間を表 2 に，それぞれ示す．

コアを 6 個使用したとき，速度向上率は理想値（= 6）の 92%であった．このとき，最も負荷の軽いコア（6 番）の実行時間が，最も負荷の重いコア（3 番）の実行時間の 87%であり，この負荷の偏りが，理想値よりも性能を悪くしていると考えられる．

表 1: 1〜6 個のコアを使用時の実行時間と速度向上率

使用コア数	1	2	3	4	5	6
実行時間/s	3.51	1.76	1.19	0.904	0.811	0.635
速度向上率	1	1.99	2.95	3.88	4.33	5.53

表 2: 6 コア使用時におけるコアごとの実行時間

コアの番号	1	2	3	4	5	6
実行時間/s	0.574	0.594	0.628	0.613	0.566	0.549

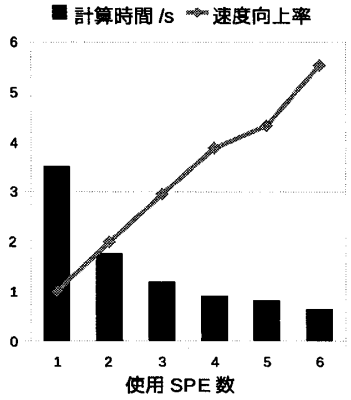

図 3: 1〜6 個のコアを使用時の実行時間と速度向上率

5 まとめ

本研究では，DMC 法をマルチコアプロセッサ向けに並列化し，評価実験を行った．実験の結果，コアを 6 個使用したとき，速度向上率は理想値（= 6）の 92%が得られた．今回は，静的な負荷分散を行ったので，まだ負荷に偏りがある．動的な負荷分散を実現することを今後の課題とする．

参考文献

[1] 藤本敬介，守屋俊夫，中山泰一: 格子形状の変形による Marching Cubes 法の細部表現能力の向上，情報処理学会論文誌，Vol.49, No.2, pp.1031-1040(2008).

[2] Lorensen, W. E. and Cline, H. E.: Marching cubes: A high resolution 3D surface construction algorithm, Computer Graphics, Vol.21, No.4, pp.163-170(1987).

[3] 大石進一: 精度保証付き数値計算，コロナ社，2000.