グラフィクスとCAD 86－5
（1997. 8. 21）

ポリゴン頂点の検索処理の不要な高速等値面生成手法

伊藤 貴之* 山口 泰** 小山田 耕二*

* 日本アイ・ビー・エム (株) 東京基礎研究所 ** 東京大学教養学部

* {itot, koyamada }@trl.ibm.co.jp ** yama@graco.u-tokyo.ac.jp

概要:

等値面のポリゴン頂点の大半は、複数のポリゴンに共有されている。そのため、等値面のポリゴンの生成過程では、そのポリゴン頂点がすでに生成された隣接ポリゴンと共有されるものなのかどうかを検索する処理が必要である。この処理は、ポリゴン生成の過程において、大きな計算量を占めている。

本報告では、ポリゴン頂点が接する格子辺を共有する格子を同時に処理することで、ポリゴン頂点の検索を不要にした、効率的な等値面生成手法を提案する。筆者らの実装では、本手法によって約 20 パーセントの処理速度の向上を実現することができた。

キーワード: 可視化、等値面、ポリゴン頂点。

Fast Isosurface Generation without Polygon-vertex Identification

Takayuki ITOH* Yasushi YAMAGUCHI** Koji KOYAMADA*

* IBM Research, Tokyo Research Laboratory

** The College of Arts and Sciences, The University of Tokyo

Abstract:

Most of polygon-vertices of an isosurface are shared by several polygons. Therefore, the polygon-vertex identification process which searches for the polygon-vertex generated at the same position is necessary in isosurfacing methods. The identification process generally occupies the largest part of the computational time in constructing polygons.

This paper proposes an efficient isosurfacing method which does not require the polygon identification process, since the method processes all cells adjacent to a cell-edge which a polygon-vertex lies at the same time. In the authors' implementation, the method is about 20 percent faster than the conventional isosurfacing methods.

Key Words: Visualization, Isosurface, Polygon-vertex.

1 はじめに

医療測定結果や科学技術計算結果として得られるボリューム・データから、スカラ場を可視化する手段として、等値面生成がある。等値面はボリューム・データ中の $S(x,y,z)-C=0$ を満たす点の集合であり、理論的には曲面であるが、ポリゴンの集合で近似表現されることが多い。

等値面生成の典型的な手順は、以下の通りである。

処理 1: ボリューム・データ中の各格子について、格子点 N_n のもつスカラ値 C_n と、与えられた定数 C の大小比較をする。$C_n > C$ である格子点と $C_n < C$ である格子点の両方が存在するとき、当該格子は等値面と交差する。

処理 2: 等値面と交差する格子について、その交差部分をポリゴンで近似する [1, 2]。

処理 3: 各ポリゴン頂点における座標値、および必要があれば法線ベクトルを算出する。

一般的に、等値面と交差する格子はボリューム・データ中のごく一部である。このことから近年では、等値面と交差しない格子の多くに対する処理を省略する、高速な等値面生成手法 [3, 4, 5, 6, 7, 8, 9, 10] が多く報告された。これらの手法によって、上記の **処理 1** に要する時間は非常に小さくなった。表 1 は、筆者らの実装において、すべての格子を処理する手法と、多くの非交差格子を処理から省略する高速な手法 [10] の、処理時間を比較したものである。この測定結果は、四面体格子で構成される非構造ボリュームを用いて、スカラ値の異なる 20 枚の等値面を生成するのに要した処理時間である。ここで、N_c と N_n はボリューム中の格子数と格子点数、N_t と N_v は 20 枚の等値面のポリゴン数とポリゴン頂点数を表したものである。T_1, T_2, T_3 はそれぞれ **処理 1**、**処理 2**、**処理 3** に要する時間を示したものである。これらの結果から、筆者らの研究成果では T_1 を非常に小さくすることに成功しており、さらに等値面生成を高速化するには T_2 や T_3 を小さくする別の手法が必要であることがわかる。

表 1: 等値面生成中の処理時間

ボリューム	1	1	2	2
手法	旧	新	旧	新
N_c	61680		346644	
N_n	11624		62107	
N_t	80995		135358	
N_v	43158		71358	
T_1 (sec.)	8.30	0.09	42.23	0.57
T_2 (sec.)	3.80	3.45	6.25	5.88
T_3 (sec.)	0.76	0.75	1.14	1.15
T_{total} (sec.)	12.86	4.29	49.62	7.60

等値面のポリゴン頂点の大半は、複数のポリゴンに共有されている。そのため、**処理 2** の過程で生成されるポリゴンについて、そのポリゴン頂点がすでに生成された隣接ポリゴンと共有されるものなのかどうかを検索し、すでにポリゴン頂点が生成されていればそれを共有するように実装されるのが一般的である [11]。筆者らの従来の実装では、ポリゴン頂点検索にハッシュ・テーブルを用いており、この検索に **処理 2** の多くの時間を費やしていることがわかっている。ポリゴン頂点を共有しないように実装すること自体は可能であるが、その場合にはポリゴン頂点数は約 6 倍となる。この時、たとえ T_2 が小さくなっても、逆に T_3 が 6 倍になってしまう。さらに、等値面のメモリ使用総量が 3 倍程度に増加する。よってポリゴン頂点の共有は、等値面生成の効率化には不可欠であることがわかる。

本報告では、ハッシュ・テーブルなどを利用したポリゴン頂点の検索処理を用いずに、ポリゴン頂点の共有を実現する、効率的な等値面生成手法を提案する。本手法では、交差格子内部におけるポリゴンを生成した後、ポリゴン頂点が存在するそれぞれの格子辺について、格子辺を共有する格子を隣接順に探索し、探索したそれぞれの格子内部のポリゴンに同一のポリゴン頂点を登録する。この処理によって、ひとつのポリゴン頂点を共有するすべてのポリゴンが同時に処理されるので、そのポリゴン頂点を後になって検索する必要がない。よって、本手法ではポリゴン頂点の検索処理が不要である。

2 従来の等値面生成手法

2.1 等値面のポリゴン頂点の検索

等値面のポリゴン生成の典型的な手法である Marching Cube 法 [2] では、すべてのポリゴン頂点は格子辺上につくられる。ポリゴン頂点を検索する典型的な手法 [11] では、ポリゴン頂点と、ポリゴン頂点が接する格子辺の組を、ハッシュ・テーブルを用いて検索する。ここで、一般的にボリューム・データは格子の集合および格子点の集合で表現され、格子辺の集合は保持されていないことが多い。よって文献 [11] では、格子辺は両端の格子点を用いて表現されている。

図 1 は、ポリゴン頂点の検索処理の一例を示したものである。この例では、ポリゴン P_1 が最初に生成されたときに、そのポリゴン頂点 a, b, c, d をハッシュ・テーブ

ルに登録している。この時、例えば a は格子点 (n_1, n_2) と組になって登録されているように、ポリゴン頂点はそれぞれの格子辺の両端の格子点と組になって登録される。続いてポリゴン P_2 が生成される時に、P_2 のポリゴン頂点が接する格子辺がすでにハッシュ・テーブルに登録されているかを検索する。この時、格子点の組 (n_1, n_2) がすでに登録されているので、(n_1, n_2) と一緒に登録されているポリゴン頂点 b が検索され、P_2 のポリゴン頂点として用いられる。

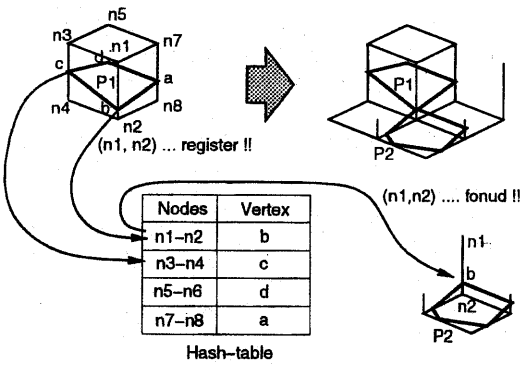


図 1: ポリゴン頂点の検索処理

2.2 自己増殖的な等値面生成手法

等値面に交差する格子が 1 つでも抽出されれば、その格子に隣接する交差格子を再帰的に探索することで、等値面を効率的に生成することができる [1, 12, 13]。筆者らはこのような手法を、「自己増殖的な等値面生成手法」と呼んでいる。

筆者らの実装では、幅優先探索によって隣接交差格子を処理している。まず、あらかじめ抽出された交差格子を FIFO に登録する。続いて、FIFO から格子を 1 個ずつ抽出し、格子内部のポリゴンを生成するとともに、隣接交差格子の中でまだ FIFO に登録されていないものがあれば登録する。の処理を、FIFO が空になるまで反復することで、等値面が生成される。

図 2 は、筆者らの実装を示したものである。ポリゴン P_1 が最初に生成され、隣接する 4 個の交差格子が FIFO に登録されている。続いて、これらの格子が FIFO から抽出され、ポリゴン P_2, P_3, P_4, P_5 が生成されている。さらに、ポリゴン P_2 の生成の際に登録された隣接交差格子が抽出され、ポリゴン P_6, P_7, P_8 が生成されている。

自己増殖的な等値面生成手法では、あらかじめ最低 1

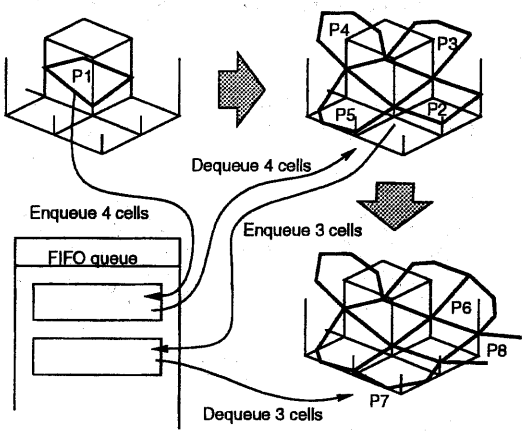


図 2: 自己増殖的な等値面生成手法

個の交差格子が抽出されている必要がある。特に、等値面が複数の非連結な部位で構成される場合には、それぞれの部位における交差格子が抽出されている必要があり、その自動抽出は大きな研究課題であった。

2.3 細線化手法を利用した自己増殖的な等値面生成手法

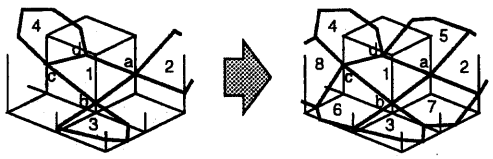
筆者らは、複数の非連結な部位で構成される等値面においても、それぞれの部位における交差格子を自動抽出し、自己増殖的に等値面を生成する、効率的な手法を提案している [9, 10]。この手法では、ボリューム中の極大点および極小点をすべて抽出し、細線化手法によって極大点および極小点を連結する骨格を生成する。この骨格には、骨格を構成する最低 1 個の格子が任意の等値面と交差する、という特徴がある。よって、骨格を構成する格子と等値面との交差判定によって最低 1 個の交差格子が抽出され、その交差格子を出発点にして自己増殖的な等値面生成を呼び出すことで、多くの非交差格子との処理を省略して等値面を生成することができる。

ここで、ボリュームを構成する格子数を n とすると、細線化手法によって生成される骨格の格子数は平均で $O(n^{1/3})$ であり、等値面と交差する格子数は平均で $O(n^{2/3})$ である。よってこの手法は、等値面生成に要する処理時間が $O(n)$ より小さくなることが期待できる。そのためこの手法は、格子数の非常に多い大規模なボリュームにおいても、処理時間の増加を抑えることができる。

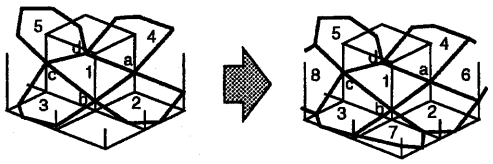
3 格子辺に着目した等値面生成手法

3.1 概略

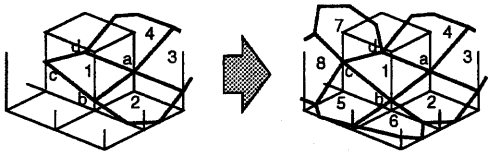
まず、従来の等値面生成手法における、ポリゴン頂点の検索処理の必要性について、もう一度考察する。



(a) Polygon generation in a straightforward algorithm



(b) Polygon generation in a plopagation algorithm



(c) Polygon generation in the new algorithm

図 3: 等値面生成手法におけるポリゴンの生成順

図 3 (a) のポリゴンにつけられた番号は、旧来の等値面生成手法におけるポリゴン生成順の例である。ポリゴン 1 とポリゴン頂点 a, b, c, d が最初に生成されている。続いてポリゴン 2 が生成されたとき、ポリゴン頂点 a はハッシュ・テーブルから検索されている。同様にポリゴン頂点 b も、ポリゴン 3 が生成されるときに検索されている。同様にして、生成されたポリゴン頂点が等値面生成中のいつ再び必要になるかわからないので、旧来の手法ではポリゴン頂点の検索処理が必要である。

図 3 (b) のポリゴンにつけられた番号は、従来の自己増殖的な等値面生成手法におけるポリゴン生成順の例である。ポリゴン 1 とポリゴン頂点 a, b, c, d が最初に生成されている。続いてポリゴン 1 に隣接するポリゴン 2, 3, 4, 5 が生成される。この手法では隣接順にポリゴンを処理するものの、一度処理されたポリゴン頂点はいつ再び必要になるかわからない。例えば、ポリゴン頂点 b を共有する ポリゴン 1, 2, 3, 7 のうち、ポリゴン 7 は別のポリゴン 4, 5, 6 のあとに処理されるので、ポリゴン頂点 b を再び検索する必要がある。よって、従来の自己増殖的な等値面生成手法においても、ポリゴン頂点の検索処理は必要である。

図 3 (c) のポリゴンにつけられた番号は、本報告で提案する新しい自己増殖的な等値面生成手法におけるポリゴン生成順の例である。この例では、ポリゴン 1 が最初に生成され、そのポリゴン頂点を共有するポリゴンが続いて生成されている。まず、ポリゴン頂点 a が接する格子辺に着目し、この格子辺を共有する格子を探索し、ポリゴン頂点 a を共有するポリゴン 2, 3, 4 が一気に生成され、a が登録される。同様に、ポリゴン 1 のポリゴン頂点 b, c, d を共有するポリゴンについても、b, c, d が一気に登録される。このように本手法では、ひとつのポリゴン頂点を共有するすべてのポリゴンに、同時にそのポリゴン頂点が登録されるので、再びそのポリゴン頂点が検索される必要がない。

図 4 は、本報告で提案する手法の擬似コードである。等値面に交差する格子 C_i に対して、Marching Cubes 法 [2] によってポリゴン頂点が接する C_i の格子辺 $\{E_1, E_2, ...\}$ を抽出する。格子辺上におけるスカラ値は一般的には線形補間されるので、格子辺上にはポリゴン頂点は 2 点以上存在することはない。そこで、抽出されたそれぞれの格子辺 E_n について、E_n 上にすでにポリゴン頂点が生成されているかどうか確認し、ポリゴン頂点が生成されていない場合にのみ、ポリゴン頂点 V_n を新規に生成し、ポリゴン P_i に登録する。続いて、新規に生成された V_n が接する格子辺 E_n を共有する格子 $\{C_1, C_2, ...\}$ を隣接順に探索し、それぞれの格子に登録されているポリゴン P_j に V_n を登録する。格子 C_j の内部にポリゴン P_j がまだ生成されていない場合には、Marching Cubes 法を用いてポリゴン P_j を生成し、格子 C_j に登録する。

本手法において、大半の交差格子は、擬似コードの for-loop (3) に示す探索処理ではじめて処理され、格子内部にポリゴンが生成される。続いて、擬似コードの for-loop (1) に示す反復処理で再び同じ格子が処理され、ポリゴン頂点がすべて登録される。よって本手法は、同一の交差格子を複数回処理することになるが、それでもハッシュ・テーブルを用いたポリゴン頂点の探索処理が不要なので、従来の手法よりも高速に等値面を生成できる。

筆者らの実装では、細線化手法を利用した手法 [10] を利用して等値面との交差格子を抽出している。この手法によって抽出された格子はあらかじめ FIFO に登録され、それ以外の交差格子は擬似コードの for-loop (3) に示す探索処理の過程で FIFO に登録される。擬似コードの for-loop (1) は、FIFO が空になるまで格子を抽出する反復処理に置きかえられる。

```
void Isosurfacing() {
  /* for-loop (1) */
  for (each intersected cell C_i) {
    if(polygon P_i in C_i is not constructed) {
      Construct P_i in C_i;
    }
    /* for-loop (2) */
    for(each intersected edge E_n) {
      if(a polygon-vertex V_n on E_n
        is not registered into P_i in C_i ) {
        Allocate V_n on E_n;
        Register V_n into P_i in C_i;
        /* for-loop (3) */
        for(each cell C_j which shares E_n) {
          if(P_j in C_j is not constructed) {
            Construct P_j in C_j;
          }
          Register V_n into P_j in C_j;
        } /* for(each C_j) */
      } /* if(there is not V_n) */
    } /* for(each E_n) */
  } /* for(each C_i) */

  for(each polygon-vertex V) {
    Calculate position and normal vector;
  }
} /* end Isosurfacing() */
```

図 4: 格子辺に着目した等値面生成手法の擬似コード

3.2 データ構造と実装方法

本報告では、四面体格子で構成される非構造ボリュームに対するデータ構造と実装方法を示す。

本実装では、格子点は座標値とスカラ値をもち、格子は格子点と隣接格子へのポインタ、および格子内部に生成されるポリゴンへのポインタ、さらに FIFO に登録されたか否かを示すフラグをもつ。ポリゴンはポリゴン頂点へのポインタと、ポリゴン頂点が接する格子辺を表すマスク値をもつ。ポリゴン頂点は座標値と法線ベクトルをもつ。図 5 は、これらのデータ構造を擬似コードで示したものである。

本実装では、等値面を生成する前に、まず格子の変数 "already-visited" および "polygon" をクリアする。図 4 における for-loop (3) の過程で、格子 C_j がはじめて処理された時に、C_j の内部にポリゴン P_j を生成し、格子の変数 "polygon" に P_j を記録する。また、C_j の変数 "already-visited" に True が代入され、C_j は FIFO に登録される。すでに変数が True になっていた場合には、C_j は再び FIFO に登録されない。さらに、生成されたポリゴン P_j に対して、Marching Cubes 法を用いてポリゴン頂点が接する格子辺を特定し、ポリゴンの変数 "action-value" にそれを記録する。

```
typedef struct _NODE {
  double position[3];
  double scalar-value;
} NODE;

typedef struct _CELL {
  NODE *nodes[4];
  CELL *adjacent-cells[4];
  POLYGON *polygon;
  char already-visited;
} CELL;

typedef struct _POLYGON {
  VERTEX *vertex[4];
  int action-value;
} POLYGON;

typedef struct _VERTEX {
  double position[3];
  double normal-vector[3];
} VERTEX;
```

図 5: 本手法を実装するデータ構造の例

図 4 における for-loop (1) の過程で、FIFO から格子 C_i が抽出されると、まず C_i の内部のポリゴン P_i の変数 "vertex" と "action-value" を確認する。P_i に生成されるべきポリゴン頂点があれば、図 4 における for-loop (2) の過程において、ポリゴン頂点 V_n を生成して変数 "vertex" に記録すると同時に、生成されたポリゴン頂点が接する格子辺の両端の格子点を、ポリゴン頂点の変数 "node" に記録する。

続いて、図 4 における for-loop (3) の過程で、新しく生成されたポリゴン頂点 V_n が接する格子辺を共有する格子 C_j を隣接順に順次探索する。この時に、C_j の内部のポリゴン P_j の変数 "vertex" に、新しく生成されたポリゴン頂点 V_n を登録する。

4 実行例

本手法を IBM PowerStation RS/6000 (Model 560) で実装し、処理速度を測定した結果を示す。本章で示す結果は、有限要素法などの数値解析によって得られた、四面体格子で構成される 4 個の非構造ボリュームを用いて測定したものである。

表 2 は、スカラ値を変えながら 20 枚の等値面を生成した結果を示したものである。ここで、N_c および N_n

はボリュームの格子数および格子点数を示し、N_t および N_v は 20 枚の等値面のポリゴン数およびポリゴン頂点数を示している。T_1 は従来の自己増殖的な等値面生成手法における処理時間、T_2 は本手法による処理時間を示している。T_{p1} および T_{p2} は、両手法において、ポリゴンの生成に要した処理時間を示している。

表 2: 等値面生成に要する処理時間

ボリューム	1	2	3	4
N_c	61680	346644	458664	557868
N_n	11624	62107	80468	97473
N_t	80995	135398	494480	1164616
N_v	43158	71358	251506	588796
T_{p1} (sec.)	3.45	5.88	21.35	49.80
T_{p2} (sec.)	2.53	4.01	15.72	36.20
T_1 (sec.)	4.29	7.60	26.65	60.81
T_2 (sec.)	3.35	5.52	20.61	46.80

この結果から、本手法ではポリゴン生成処理において約 25 パーセント、等値面生成処理全体においても約 20 パーセントの処理速度向上を実現したことがわかる。

5 むすび

本報告では、等値面に交差する格子辺を共有する格子を隣接順に続けて処理することで、等値面のポリゴン頂点を共有するすべてのポリゴンを同時に処理する手法を提案した。本手法では、等値面と交差する格子を複数回処理するものの、ポリゴン頂点の検索処理が不要であるので、従来の等値面生成手法よりも効率的である。

本報告では四面体格子で構成するボリュームを対象にして実装したが、原理的には六面体格子で構成されるボリュームにも実装が可能である。そこで今後の課題として、六面体格子で構成されるボリュームに対して本手法を実装し、処理時間を測定することがあげられる。

参考文献

[1] Wyvill G., McPheeters C., and Wyvill B., *Data Structure for Soft Objects,* The Visual Computer, Vol. 2, No. 4, pp. 227-234, 1986.

[2] Lorensen W. E., and Cline H. E., *Marching Cubes: A High Resolution 3D Surface Construction Algorithm,* Computer Graphics, Vol. 21, No. 4, pp. 163-169, 1987.

[3] Giles M., and Haimes R., *Advanced Interactive Visualization for CFD,* Computer Systems in Engineering, Vol. 1, No. 1, pp. 51-62, 1990.

[4] Gallagher R. S., *Span Filtering: An Optimization Scheme for Volume Visualization of Large Finite Element Models,* Proceedings of IEEE Visualization '91, pp. 68-74, 1991.

[5] Livnat Y., Shen H., and Johnson C. R., *A Near Optimal Isosurface Extraction Algorithm Using the Span Space,* IEEE Transactions on Visualization and Computer Graphics, Vol. 2, No. 1, pp. 73-84, 1996.

[6] Shen H., Hansen C. D., Livnat Y., and Johnson C. R., *Isosurfacing in Span Space with Utmost Efficiency (ISSUE),* Proceedings of IEEE Visualization '96, pp. 287-294, 1996.

[7] Welhelms J., and Gelder A. Van, *Octrees for Fast Isosurface Generation,* ACM Transactions on Graphics, Vol. 11, No. 3, pp. 201-227, 1992.

[8] Silver D., and Zabusky N. J., *Quantifying Visualization for Reduced Modeling in Nonlinear Science: Extracting Structures from Data Sets,* Journal of Visual Communication and Image Representation, Vol. 4, No. 1, pp. 46-61, 1993.

[9] Itoh T., and Koyamada K., *Automatic Isosurface Propagation by Using an Extrema Graph and Sorted Boundary Cell Lists,* IEEE Transactions on Visualization and Computer Graphics, Vol. 1, No. 4, pp. 319-327, 1995.

[10] Itoh T., Yamaguchi Y., and Koyamada K., *Volume Thinning for Automatic Isosurface Propagation,* Proceedings of IEEE Visualization '96, pp. 313-320, 1996.

[11] Doi A., and Koide A., *An Efficient Method of Triangulating Equi-valued Surfaces by Using Tetrahedral Cells,* IEICE Transactions, Vol. E74, No. 1, pp. 214-224, 1991.

[12] Bloomenthal J., *Polygonization of Implicit Surfaces,* Computer Aided Geometric Design, Vol. 5, No. 4, pp. 341-355, 1988.

[13] Speray D., and Kennon S., *Volume Probe: Interactive Data Exploration on Arbitrary Grids,* Computer Graphics, Vol. 24, No. 5, pp. 5-12, 1990.