境界表現モデルから Octree 表現への一変換手法*

小堀 研一**・石黒 和也***・久津輪敏郎**

An Accurate Conversion Method from Boundary Representations to Octree Data*

Ken-ichi KOBORI**, Kazuya ISHIGURO*** and Toshiro KUTSUWA**

In general, B-Reps model is used for three dimensional model creation. In recent years, the octree data structure gets into the spotlight as the computer technology makes rapid progress. Although several conversion algorithms of B-Reps into the octree data structure were proposed, former methods could not apply to B-Reps models with a concave shape. This paper describes a new conversion approach of a concave polyhedron into the octree data structure model. The proposed method consists of four processes, dividing a concave polygon into several convex polygons, determining whether octants intersect polygons, searching inner octants and calculating the volume occupied by a B-Reps object. Several experimental results show that the method is effective for the concave polyhedron and high conversion accuracy is obtained.

1. はじめに

現在, 三次元 CAD における形状モデルとして物体分割モデルである B-Reps (境界表現) が一般に用いられている. 物体分割モデルのソリッドモデルは形状の構成要素である面, 稜線, 頂点などに着目した表現であり, 形状表現に優れたモデルである. しかし, 幾何情報と位相情報を持つ複雑なデータ構造を扱うため, ソリッドモデルの基本機能である集合演算を行うときに多くの計算量を必要とする問題がある. 一方, 空間を分割して物体を表現する空間分割モデルが数値解析や CT に代表される医療分野で発展してきた. これは精度に少し問題はあるが, データ構造が単純であるため, 物体同士の干渉が容易に行える利点がある. したがってこれら両者を相互変換して, 互いの利点を生かしたハイブリッド型の形状モデルを実現すれば, 必要な操作に応じてモデルを使い分けることが可能となると考えられる. そこで本論文では, 物体分割モデルの多面体 B-Reps から空間分割モデルの Octree データ構造へ変換する手法について提案する.

2. Octree データ構造

空間分割モデルの代表的なものとしては Voxel があるが, 精度をあげるためには膨大なメモリ量を必要とし現実的でないため, 本論文では Octree データ構造¹⁾を採用した. これは Voxel のように空間を一定の大きさの立方体で表現するのではなく, Fig. 1 (図では二次元の Quadtree 表現) に示すように, 物体の複雑さに応じて可変の大きさの立方体 (以後, Octant という) に

Fig. 1 An object represented using a quadtree data structure

* 原稿受付 1994年3月22日

** 大阪工業大学 電子工学科 Department of Electronic Engineering, Osaka Institute of Technology; 5-16-1 Ohmiya, Asahi ward, Osaka city, Osaka 535, JAPAN

*** 大阪工業大学 電気工学専攻 Department of Electrical Engineering, Osaka Institute of Technology; ditto

Key Words: octree, B-Reps, spatial-partitioning, geometric model, CAD.

分割して表現するものである。したがって Fig. 1 の Quadrant が Octant に対応する。形状表現能力は Voxel と同じであるが、メモリ効率が優れているという特長がある。

3. 従来手法の概要

これまで、B-Reps から Octree データ構造への変換に関する研究^{2)~4)}がいくつか提案されている。扱っている B-Reps 形状は多面体であり、一般には次の二つの処理から構成されている。1 番目の処理は B-Reps と交差する Octant を決定する交差判定処理である。2 番目の処理はそれらの多角形と非交差の Octant を B-Reps 形状の内側または外側の Octant に分類する内外判定処理である。内外判定処理では、非交差 Octant が B-Reps のすべての多角形に対して内側であれば内側 Octant、一つでも外側となる多角形が存在すれば外側 Octant と判定するものである。これらの交差判定処理と内外判定処理を用いて、あらかじめユーザが定める Octree を作成するレベル（以後、最大分割レベルという）まで階層的にデータを構築する。しかしこれには問題点がある。最初の問題点は、変換対象の B-Reps が凹形状の場合には変換できないということである。一般に B-Reps で表現している物体は凹形状であるので、変換する前処理として、Fig. 2 に示すように凹 B-Reps を凸 B-Reps に分割しておく必要があった。これには二つの原因が考えられる。一つは交差判定処理において起こるもので、他方は内外判定処理の際に起こるものである。交差判定処理を行うときには B-Reps を構成する多角形と Octant との交差判定を X, Y, Z 平面に投影

Fig. 2 A concave B-Reps object divided into three convex B-Reps objects

(a) Positional relations between an octant and a concave polygon in three dimensional space

(b) Projection on X, Y and Z plane

Fig. 3 Determining whether an octant intersects a polygon

し、それぞれの平面上で二次元の交差判定を行う。しかし多角形が凹形状の場合、Fig. 3 (a) に示すように、多角形と Octant が実際には交差していない場合でも、Fig. 3 (b) のように X, Y, Z 平面に投影するとすべての平面に対して Octant は交差と判定されてしまう。

また、内外判定を行うときに、Fig. 4 (a) のように凸 B-Reps において、判定対象の Octant が B-Reps を構成するすべての多角形の内部に存在する場合、内側 Octant、一つ以上の多角形に対して外側である場合、外側 Octant と判定できる。しかし、Fig. 4 (b) のような凹 B-Reps の場合、内側に存在する Octant でも外側 Octant と判定される。したがって Fig. 4 (b) のような凹 B-Reps の場合は、Fig. 4 (c) のように三つの凸 B-Reps A, B, C に事前に分割し、判定対象の Octant をそれぞれの B-Reps に対して内外判定を行う。すなわち変換対象が凹 B-Reps である場合、まず凸 B-Reps に分割し、分割された B-Reps 群と Octant との内外判定をほかの B-Reps との位置関係を考慮に入れて行う必要があった。

ところで、Octree データ構造のほかの問題点として、複雑な形状の表現力が乏しいという欠点が存在する。例えば B-Reps で表現された球を Octree データ構造で表現する場合、階層作成レベルである最大分割レベルが 3 の場合、Fig. 5 (a) のようになる。最大分割レベルを 4 にしても Fig. 5 (b) のように球としての表現が不十分である。なお Fig. 5 は二次元で表現している。4. 以下にこれらの問題を解決する手法を提案する。

(a) In case of a convex B-Reps object

(b) In case of a concave B-Reps object

(c) An octant intersects B-Reps A and B

Fig. 4 Determining whether an octant lies inside a B-Reps object

(a) In case of level 3

(b) In case of level 4

Fig. 5 A sphere represented by using octants

4. 手法の概要

本手法は Fig. 6 に示すように大きく分けて四つの処理から構成されている。

4.1 凸多角形分割処理

変換対象の B-Reps が Fig. 7 (a) のように凹多角形をもつ場合、その多角形を Fig. 7 (b) のようにすべて凸多角形に分割する。凹形状である多角形は内角が 180° 以上の頂点（以後、鈍角点という）を一つ以上持っている。そのような頂点すべての内角を 180° 以下に分割することで、凹形状の多角形を凸形状にする。鈍角点の一つの

Fig. 6 General flow of the method

(a) A concave B-Reps object with two concave polygons

(b) A concave B-Reps object constituted by convex polygons

Fig. 7 Division of concave polygons

場合と二つ以上存在する場合に分けてこの処理を述べる。

(1) 鈍角点の一つのみ持つ多角形の分割

鈍角点 $V1$ の内角を二等分するような方向ベクトル V を求める。鈍角点 $V1$ から、その V と交差する凹多

角形の1本の稜線の両端点 e_1, e_2 へ向かうベクトルをそれぞれ E_1, E_2 とする. V と E_1 および E_2 との内積のうち大きい方を構成しているベクトルの端点と V を結んで稜線として現在の凹多角形を二つの多角形にする. 分割した二つの多角形が凸形状であれば処理を終了し, どちらか一方が凹形状であればその多角形に対して同様の処理を繰り返す.

(2) 鈍角点を二つ以上有する多角形の分割

Fig. 8 (a) のように任意の鈍角点 V の内角を二等分するような方向ベクトル V を求める. V からほかのすべての鈍角点へ向かうベクトルのなかで, 多角形の稜線と交差せず, ベクトル V との内積が最も大きいベクトルを構成する鈍角点を選ぶ. Fig. 8 (a) の場合, 鈍角点 e_2 が選ばれる. その点と V とを結んで稜線として, 現在の凹多角形を Fig. 8 (b) のように二つの多角形に分割する. この処理を分割した多角形がすべて凸形状になるまで繰り返す.

Fig. 8 Division of a concave polygon into convex polygons

以上の (1), (2) の処理を対象となる B-Reps 形状を構成する凹多角形すべてに行い, 凸多角形だけで構成される B-Reps 形状にする. これにより X, Y, Z 平面に投影して交差判定を行うときの問題を解消した.

4.2 交差判定処理

つぎに, 凸多角形1枚1枚に対して Octant との交差判定を行う. 交差判定処理は bbox, 3Dclipping および projection という三つの処理を順に行い, 最終的に Octant を交差または非交差 Octant に分類する. Fig. 9 にこの交差判定処理の手順を示す.

(1) bbox 手続き

B-Reps を構成する多角形を含む最小直方体 (bbox) と Octant との交差判定を行う. Octant と bbox が共

Fig. 9 Flow chart of intersection check

Fig. 10 Intersection check between an octant and edges of a polygon

通部分をもつ場合, 交差の可能性のある Octant (以後未判定 Octant という), 共通部分がなければ, 非交差 Octant と分類する.

(2) 3Dclipping 手続き

bbox 手続きにより未判定と分類された Octant に対して, 3Dclipping 手続きを行う. この手続きは Fig. 10 に示すように多角形を構成する全稜線と Octant との交差状態を得ることにより, 多角形と Octant との交差を判定する. これには Cohen-Sutherland の 3Dclipping^{5), 6)} を利用している. 3Dclipping により多角形の稜線と交差している Octant を交差 Octant, それ以外を未判定 Octant として分類する.

(3) projection 手続き

3Dclipping 手続きの後、未判定 Octant に対して行う処理が projection 手続きである。この projection 手続きでは、2Dclipping^{5), 6)}を用いて、Fig. 11 に示すように X, Y, Z 平面に投影された Octant に対応する正方形（以後二次元 Octant という）と B-Reps に対応する二次元多角形の交差・非交差を判定する。それによって Octant と B-Reps を構成する多角形の交差状態を調べ、判定対象の Octant は交差または非交差に分類される。すなわち X, Y, Z 平面において一つでも二次元 Octant と二次元多角形が非交差であれば、判定対象の Octant は非交差と判定される。2Dclipping は四角形と線分との位置関係を判定する処理で、四角形の辺を延長して二次元平面を九つの領域に分け、線分の端点がどの領域にあるかによって (i) 線分全部もしくは端点が四角形の内部にある、(ii) 線分全部が外部にある、(iii) 線分の一部が四角形の内部にある可能性がある、の三つの状態に容易に分類できるものである。この 2Dclipping を交差判定に应用する。

X, Y, Z 各々の平面に投影した二次元 Octant と B-Reps を構成する多角形の稜線を投影した二次元稜線との交差は、Fig. 12 のように三つの状態に分類される。

Fig. 12 (a) は多角形の二次元稜線が明らかに二次元 Octant に交差している完全交差、Fig. 12 (b) は二次元稜線が明らかに二次元 Octant に交差していない完全非交差、および 2Dclipping だけでは判定できない Fig. 12 (c) の交差かどうか、判定不可能の状態である。したがって Fig. 12 (c) の状態の Octant は次の処理によって判定する。

Fig. 11 Determining whether an octant intersects a convex polygon

2Dclipping 手続きで判定できないこの状態は、Fig. 13 に示すように二次元 Octant に対して二次元に射影された多角形の稜線が傾斜している場合のみである。そこで二次元 Octant の 4 頂点と二次元の稜線との距離を用いて判定を行う。まず Fig. 13 に示すように二次元 Octant の 4 頂点と二次元の稜線を含む無限直線との符号付き距離を求める。二次元 Octant の頂点のうちの 1 点でも二次元の Octant の中心の距離の符号と異なる符号であれば、この二次元の Octant と二次元の稜線は交差である。Fig. 13 (b) のように二次元の Octant の中心と頂点四つの符号がすべて同じであれば、非交差と判定される。

ただし、上記の projection 手続きでは Fig. 14 (a) のように、Octant が多角形を貫いている場合、Fig. 14 (b) に示すように X, Y, Z の一つ以上の平面への投影で二次元多角形が完全に二次元 Octant を包含してしまう。この場合、二次元多角形すべての稜線が Fig. 12 (b) の状態と判定されるため、三次元的には交差しているにもかかわらず、非交差と判定されてしまう。そこで、こ

Fig. 12 Three relations between edges and two-dimensional octant

Fig. 13 Calculation of distances between an edge and four vertices

(a) Positional relations between an octant and a polygon

(b) Intersection check at the state of (a)

Fig. 14 An octant passing through a polygon

れを回避するため二次元多角形の内部に二次元 Octant の中心が存在するかどうかという処理を projection 手続きに付加した. これにより, Fig. 14 の状態でも二次元 Octant の中心が多角形内部に存在するため, 交差と判定できる.

以上の三つの手続きを行うことにより, Octant を交差あるいは非交差に分類する.

4.3 内側 Octant 探索処理

交差判定処理を終了した時点で, Fig. 15 のように交差 Octant 群で閉じられた内部空間と外部に非交差 Octant が存在する. そこで B-Reps の内部に相当する Octant を内側 Octant として検出する処理を行う. この処理により, すべての Octant が交差, 内側そして外側 Octant に分類される. 最初に交差 Octant のすぐ内側の非交差 Octant から内側 Octant を探索するための Start Octant を選び出す.

この Start Octant の決定は交差 Octant に対して B-Reps の内部に内側 Octant が存在するということを利用して次のように行う. B-Reps を構成する任意の多角形一枚を取り出し, その重心 G を求める. その G を含む交差 Octant 一つを選ぶ. 次にその多角形の法線ベクトルの X, Y, Z の成分方向のうち, 最大成分の方向を取り出す. Fig. 16 に示すようにその最大成分方向とは反対方向に, 重心 G を含む Octant から内側 Octant を探索していく. 最大成分方向を用いるのはできるだけ探索回数を少なくして, 内側 Octant で

Fig. 15 Classification of octants

Fig. 16 Search for start octant

ある Start Octant を決定するためである. Fig. 16 では 2 回の探索で Start Octant を決定した様子を示している.

決定された Start Octant からすべての内側 Octant を探索する. この Octant の上下左右前後六方向を探索していく. 探索した Octant が非交差 Octant であれば, その Octant の六方向を探索していく. 探索中に子 Octant をもつ Octant に当たると, その子 Octant についても探索をする. 同様に内側 Octant または子 Octant をもたない交差 Octant に当たればその Octant のその方向の探索は終了する.

以上のようにして交差 Octant の内部にあるすべての非交差 Octant を内側 Octant に決定する.

4.4 交差 Octant の体積割合計算

従来 Octree データ構造では, 最大分割レベルにおける交差 Octant を内側 (または外側) Octant とすることで, 内側と外側の二つの状態だけに分類されていた. その結果, 形状が複雑な場合, 最大分割レベルが低い Octree データ構造では, 形状の表現精度に問題があった. 分割回数を増やすことにより表現力を補うことは可能であるが, そのためにデータ量が増大し, 処理速度が

低下するといった問題点が残されていた。そこで筆者らは、最大分割レベルでの交差 Octant の状態をそのまま表現することにより、分割回数を増やすことなく複雑な形状を表現できる手法^{7),8)}を開発した。

まず、最大分割レベルにおける交差 Octant の 8 頂点と、交差している多角形との符号を考慮した距離を求める。Octant に多角形が複数枚交差している場合は、各頂点における距離をそれらの多角形を含む平面ごとに求め、その距離の中で一番大きいものを各頂点にもたせる。Fig. 17 の多角形二枚が Octant に交差している場合を例にとって二次元で説明する。Fig. 17 (b), (c) から Octant の頂点 1 では $k_1 > k_2$ であるので多角形 1 の距離 k_1 を採用し、頂点 3 では $l_1 > l_2$ なので多角形 1 の距離 l_1 を採用する。この例ではその距離は Fig. 17 (d) のようになる。

つぎに、各頂点での距離を用いて Octant の稜線上の交点を算出する。頂点 1 と 3 の距離は異符号なので頂点 1 と 3 を結ぶ稜線上に交点が存在し、その交点は k_1 と l_1 の内分点として求められる。

Fig. 17 Calculation of the volume occupied by a B-Reps object

同様に交点 B も頂点 2 と 4 の距離 m_2, n_2 から求めることができる。

以上のようにして求めた交点すべてを結ぶことで、Octant 内の近似 B-Reps 形状を構成する多角形を生成する。そして Marching Cubes 法^{9),10)} による分類方法を用いて交差 Octant を 15 通りに分類し、その分類からその Octant に含まれる B-Reps を表現する。したがって Fig. 5 (b) のような Octant の最大分割レベル 4 の Octant も Fig. 18 のように正確に表現される。

Fig. 18 A sphere represented by proposed method

5. 実験結果

数種類の B-Reps 形状を対象として (1) 従来法, (2) 本論文で提案した交差, 内外判定処理を用いた本手法 1, および, (3) 本手法 1 に 4.4 の体積割合計算法を加えた本手法 2 の三つの方法について変換実験を行った。ここで取り上げた従来法は、少ない計算量で高速に凸多面体を Octree データ構造に変換できる登尾らの手法^{11),12)}を用いた。変換対象の B-Reps としては Table 1 のような 3 種類を取り上げた。また、変換後の表現精度は B-Reps の真の体積と変換後の Octant の体積との相対誤差によって評価した。なお、使用したワークステーションはオムロンデータゼネラル社の AV350 (cpu 性能：約 28MIPS) である。

Table 2 から、同じ最大分割レベルで従来法よりも本手法 2の方が大幅に体積精度が向上していることがわかる。例えばレベル 5 の場合、従来法では体積精度の相対誤差は 22.3% であるのに対し、本手法 2 では相対誤差は 0.2% となっている。すなわち、真値に近い値が得られるといえる。また Table 2 から、従来法のレベル 7 においては処理速度 23.68s, 体積精度相対誤差 5.1% であるが、本手法 2 のレベル 5 においては処理速度 15.1s, 相対誤差 0.2% となっており、処理速度では約 1/2 でありながら相対誤差は約 1/25 という高い表現精度が得られている。またメモリ容量については、従来法では Octant の個数に対して最低 1bit 必要であるのに対して、本手法では最低 8bit 要するが、Octant の数がレベル 5 で 10321 と少ないので、メモリ容量も

Table 1 Three B-Reps objects to be converted in the experiment

	Type of polygons	Number of polygons	Type of B-Reps objects
Sphere	凸	128	凸
Suspension shape	凸	34	凹
Star shape	凹・凸	12	凹

Table 2 B-Reps shape : Statistics for execution of the method (Sphere)

method \ Level		5	6	7
Former method	Processing time (in seconds)	4.57	10.40	23.68
	Volume (Relative error)	75031 (2.2E-1)	67924 (1.1E-1)	64524 (5.1E-1)
	Number of octants	6601	26057	102537
Proposed method 1	Processing time (in seconds)	11.61	50.25	226.43
	Volume (Relative error)	79731 (3.0E-1)	72366 (1.8E-1)	69270 (1.3E-1)
	Number of octants	10321	46737	242785
Proposed method 2	Processing time (in seconds)	15.1	67.3	358.7
	Volume (Relative error)	61225 (2.0E-3)	61314 (6.2E-4)	61341 (1.8E-4)
	Number of octants	10321	46737	242785

Table 3 B-Reps shape : Statistics for execution of the method (Suspension shape)

method \ Level		5	6	7
Proposed method 1	Processing time (in seconds)	7.13	31.35	152.22
	Volume (Relative error)	4059351 (4.6E-1)	3611368 (3.0E-1)	3349092 (1.9E-1)
	Number of octants	5561	26249	135233
Proposed method 2	Processing time (in seconds)	7.85	34.12	167.11
	Volume (Relative error)	2914171 (5.0E-2)	2847119 (2.4E-2)	2798344 (7.0E-3)
	Number of octants	5561	26249	135233

Table 4 B-Reps shape : Statistics for execution of the method (Star shape)

method \ Level		5	6	7
Proposed method 1	Processing time (in seconds)	6.65	39.54	257.72
	Volume (Relative error)	13776 (8.5E-1)	10680 (4.4E-1)	9448 (2.7E-1)
	Number of octants	4977	29937	203665
Proposed method 2	Processing time (in seconds)	7.51	43.28	290.15
	Volume (Relative error)	8847 (3.0E-3)	8835 (1.7E-3)	8829 (1.0E-3)
	Number of octants	4977	29937	203665

(a) Proposed method 1

(b) Proposed method 2

Fig. 19 Sphere

(a) Proposed method 1

(b) Proposed method 2

Fig. 20 Suspension shape

(a) Proposed method 1

(b) Proposed method 2

Fig. 21 Star shape

$$\frac{\text{本手法 2}}{\text{従来法}} = \frac{8 \text{ bit} \times 10321 \text{ 個}}{1 \text{ bit} \times 102537 \text{ 個}} = 0.81$$

となり、従来法のレベル7に比較しても0.81倍と本手法2の有効性が明らかである。

Table 3, および4は変換対象の凹B-Reps形状で

あるため、従来法では変換できないので、本手法 1 と 2 について実験を行った。これらの表から体積割合計算法を付加した本手法 2 は、本手法 1 よりも体積の相対誤差が大幅に向上していることがわかる。しかも本手法 2 は本手法 1 よりもわずかに約 10 数 % の処理時間の増加にとどまっている。したがって、体積割合計算法による体積精度の向上は有効であると考えられる。Fig. 19 (a), 20 (a), 21 (a) は本手法 1 を、Fig. 19 (b), 20 (b), 21 (b) は本手法 2 を用いて変換した結果である。Fig. 19~21 から明らかなように、体積割合計算手法を用いた本手法 2 の方が体積精度が向上し、もとの B-Reps 形状に近い Octree で表現されていることがわかる。

6. おわりに

従来、変換対象の B-Reps が凹形状の場合、その凹形状を凸形状に分割し、それぞれの凸 B-Reps に対して交差判定および内外判定を行い、Octree データに変換し、その複数の Octree データの接続関係を考慮して処理を行う必要があった。

本論文では提案した凸多角形分割処理、交差判定処理、および内側 Octant 探索処理を用いて B-Reps 形状を Octree データに変換する新しい手法を提案した。これらの三つの処理を導入したことで、あらかじめ凹 B-Reps を凸分割して複数の凸 B-Reps に対する交差判定を行わなければならないといった問題点を解決した。また Octree の最大分割レベルが低い場合、Octree データ構造には物体の表現力が低いという問題があった。これについても体積割合計算法を用いて、交差状態の Octant を Marching Cubes 法と同様に交差状態を 15 通りに分類し、その Octant を交差の状態のまま表現することにより、もとの B-Reps に対する表現精度が高まった。以上のことから B-Reps との近似精度が高く、効率的なメモリ容量での Octree データ構造の変換が可能となった。

今後は逆に Octree データ構造表現された形状を B-Reps 形状に変換する手法を開発して相互変換を実現して、両者の長所を生かした形状モデラへと発展させる予定である。

参 考 文 献

- 1) F. Foley, A. van Dam, S. Feiner and J. Hughes: Computer Graphics Principles and Practice, Addison-Wesley Publishing Company, pp. 533~562 (1990)
- 2) W. R. Franklin and V. Akman: Building an Octree from a Set of Parallelepipeds; IEEE Computer Graphics and Applications, Vol. 5, No. 10, pp. 58~64 (1985)
- 3) M. Tammien and H. Samet: Efficient Octree Conversion by Connectivity Labeling; Computer Graphics, Vol. 18, No. 3, pp. 43~51 (1984)
- 4) M. Yau and S. N. Srihari: A Hierarchical Data Structure for Multidimensional Images; Comm. ACM, Vol. 26, No. 7, pp. 504~515 (1983)
- 5) R. F. Sproull and I. E. Sutherland: A Clipping Divider; Proc. of Fall Joint Computer Conference, Thompson Books, pp. 765~775 (1968)
- 6) I. E. Sutherland and G. W. Hodgeman: Reentrant Polygon Clipping; CACM, Vol. 17, No. 1, pp. 32~42 (1974)
- 7) 石黒, 上田, 小堀: 境界表現から Octree データ表現への高精度変換手法; 情報処理学会第 45 回全国大会論文集, 2~405 (1992)
- 8) 石黒, 小堀, 久津輪: 多面体データを空間分割モデルへ高精度変換する一手法; 1993 年電子情報通信学会秋季大会論文集, 6~325 (1993)
- 9) W. E. Lorensen and H. E. Cline: Marching Cubes a High Resolution 3D Surface Construction Algorithm; Computer Graphics, Vol. 21, No. 4, pp. 163~169 (1987)
- 10) 石井, 安田, 横井, 鳥脇: マーチングキューブ法の改良アルゴリズムについて; グラフィックスと CAD, Vol. 60, No. 9, pp. 63~70 (1992)
- 11) 登尾, 福田, 有本: BReps からオクトツリーへの変換アルゴリズムとその評価; 情報処理学会論文誌, Vol. 28, No. 10, pp. 1003~1012 (1987)
- 12) 登尾, 福田, 有本: オクトツリーを用いた高速干渉チェック法; 日本ロボット学会誌, Vol. 5, No. 3, pp. 21~29 (1987)