

面積ゼロ3角形を用いた3角形 BRep

荒 川 佳 樹†

多角形面で境界面が構成される BRep は、データ構造および処理アルゴリズムが非常に複雑となり、その処理系も大規模なものとなる。そこで、3角形処理の持つ単純性に注目して、3角形面を用いる BRep が提案されてきている。3角形 BRep ではデータ構造および処理アルゴリズムは単純化されるが、データ量(3角形の数)は非常に増える。そこで、これまでに発表されている3角形 BRep はいずれも、多角形と3角形の両方を用いて処理系を実現し、両者の優れた特性を利用し欠点を補い合っている。しかし、このような BRep では多角形と3角形が併存することとなり、いかに効率的かつ適切に多角形を3角形に分割するかという「多角形分割の問題」が生じる。そこで、筆者らは面積ゼロ3角形を新たに導入することにより、3角形の数の増大などの3角形 BRep が持つ欠点を解消し、多角形面をまったく用いない3角形面のみを用いた BRep を構築した。面積ゼロ3角形とは、その3つの頂点が同一直線上にある面積がゼロの3角形である。これにより、多角形分割の問題を根元的に排除した。そして、面積ゼロ3角形を用いた形状演算のアルゴリズムを提案している。

Triangulated BRep using Linear Triangles

YOSHIKI ARAKAWA†

BRep using arbitrary polygon faces has as its main problem the fact that it requires complicated data structures and algorithms, aggravating robust processing. A triangulated BRep, the simplest among the BRep models in terms of data structures and algorithms, overcomes these drawbacks but may still result in poorly performing algorithms, suffering from a rapidly increasing number of triangular faces as Boolean set operations are carried out. This is caused by the fact that the division of a triangle causes its neighbouring triangles also to become divided. We solved this problem by introducing a so-called Zero Triangle, a triangle of which the three edges constituting it are collinear. A new algorithm, presented in this paper, performs Boolean set operations on Zero Triangles, that avoid the division of neighbouring triangles and that unify triangles locally. The proposed algorithm performs substantially better than previous algorithms, due to its ability to reduce the number of triangles it processes extremely.

1. はじめに

Boundary Representation (BRep) は代表的かつよく用いられるソリッドモデルである^{1), 11)~13)}。BRep ではこれまで、その境界面は一般的に多角形面(以下、単に多角形と呼ぶ)で表現、処理されてきた²⁾。しかし、このような多角形に基づいた BRep はデータ構造および処理アルゴリズムが非常に複雑となり、その処理系も大規模なものとなる。そこで結果として、高い信頼性を持つ実用性のある処理系を構築することが難しいことが知られている¹⁷⁾。

このような課題を解決するために、3角形処理の持つ単純性に注目して、多角形の代わりに3角形面(以下、単に3角形と呼ぶ)を用いた BRep が提案されてきている^{7), 16)~18)}。一般的に言って、多角形を用いるとデータ量は少なく済むが、データ構造および処理が複雑になる。一方、3角形を用いると、データ構造および処理は単純になるが、データ量は非常に増える。そこで、これまでに発表されている3角形を用いた BRep はいずれも、多角形と3角形の両方を用いて処理系を実現している。これにより、両者の優れた特性を利用し、欠点を補い合っている。しかし、このような BRep では多角形と3角形が併存することとなり、いかに適切かつ効率的に多角形を3角形に分割するかという「多角形分割の問題」が生じる。

筆者らは、面積ゼロ3角形を新たに導入することにより、3角形の数の増大などの3角形 BRep が持つ欠点を解消した。そして、このような面積ゼロ3角形

† 郵政省 通信総合研究所 関西先端研究センター 知覚機構研究室

Auditory & Visual Informatics Section, Kansai Advanced Research Center, Communications Research Laboratory, Ministry of Posts & Telecommunications

を用いた、多角形をまったく用いない 3 角形のみを用いた BRep を提案している。本論文では、面積ゼロ 3 角形を用いた 3 角形 BRep の既要について述べ、その形状演算のアルゴリズムを提案する。

第 2 章ではこれまでに提案されている BRep について述べ、第 3 章では面積ゼロ 3 角形を用いた 3 角形 BRep を新たに提案する。第 4 章ではこの BRep の形状演算アルゴリズムについて述べ、第 5 章ではこのアルゴリズムの計算機実験による評価結果を報告する。

2. 従来の BRep

これまでに発表されている BRep を、その境界面の種類により分類すると次の 3 つのタイプとなる。ただし、ここでは境界面が平面で構成される BRep に議論を限定する。

- (a) 多角形モデル
- (b) 多角形/3 角形モデル
- (c) 3 角形/多角形モデル

(a) は最もオーソドックスな従来からある BRep である²⁾。すなわち、その境界面は任意の多角形で表現され、そのデータ構造および処理アルゴリズムは共に多角形に基づいている。しかし、このような多角形 BRep は、非凸立体あるいは穴などのある複雑な形状に対応しようとする、データ構造および処理アルゴリズムが非常に複雑となり、その処理系も大規模なものとなる。結果として、高い信頼性を持つ実用性のある処理系を構築することが難しいことが知られている¹⁷⁾。

そこで、BRep のデータ構造および処理アルゴリズムを単純化・効率化する試みが、いろいろと取り組まれてきている。その代表的なものが 3 角形処理の導入である。山口らは (b) のタイプを提案している^{16)~18)}。これは多角形を主表現としている点では (a) と変わりなく、多角形データが主データとなる。そして、上述したような多角形処理の持つ複雑性を解消するために、3 角形の持つ優れた特性である単純性を利用しようとするものである。すなわち、多角形を 3 角形に分割してから、形状演算を行っている。これにより、多角形では非常に複雑であった交線を求める処理などが非常に単純化・効率化される。

一方、Hubbard は (c) のタイプを提案している⁷⁾。これは、(b) のタイプとは対極であり、3 角形が主表現となる。そして、3 角形処理が持つデータ量 (3 角形の数) の増大という欠点を補うために、多角形の持

つ優れた特性を利用するものである。すなわち、このモデルではすでに 3 角形表現されているので、(b) のモデルのように多角形から 3 角形を生成することなく、3 角形相互の交差判定から容易に直接的に交線を求めることができる。この交線データから局所的に 3 角形を再分割すると、不必要な 3 角形分割が起こり、3 角形の数が一挙に増えてしまう。そこで、Hubbard はいったん 3 角形から多角形を合成し、この多角形と交線データを用いて「大局的な分割」を行い、生成される 3 角形の数を最小化している。

しかし、これらの方法では次の問題点が生じる。

- (1) 形状演算を行う度に絶えず多角形を 3 角形に分割する処理が発生する。
- (2) しかも、この多角形分割により生成された 3 角形の質の問題が起こる。

(1) に関しては、山口らは多角形を 3 角形に分割する非常に単純なアルゴリズムを提案しているが¹⁶⁾、多角形の分割処理自体が必要なことには変わりなく、それも形状演算の度に分割処理が発生する。形状 (多角形) が多くの凹凸や穴を持ち複雑になってくるとこの処理の演算量も増え、そのオーバーヘッドの問題が生じることが予想される。さらに、このような複雑な多角形をこのアルゴリズムで 3 角形分割した場合、非常にいびつな面積がゼロに近い、いわゆる「不良 3 角形」が生成される場合がある。このような 3 角形は他の処理に精度のあるいは信頼性の点でかなり悪い影響を及ぼすであろう。一方、Hubbard は多角形分割のアルゴリズムとして山口らとは異なる「単調な多角形 (monotone polygon)」を用いるアルゴリズムを採用している⁷⁾。このアルゴリズムにおいても、その生成される 3 角形の質に問題があることは、Hubbard 自身が指摘している⁷⁾。

この多角形の 3 角形分割の問題点について図 1 の例を用いて、さらに具体的に説明する。図 1 (1) では 3 角形 a, b, c, d, e, f は 1 つの平面を構成しており、a, b に交線 p が発生している。Hubbard の方法では、まず a, b, c, d, e, f から構成される多角形が求められて、この多角形が交線により最小個数の 3 角形に分割される (図 1 (2))。そして、不要平面 (ここでは 3 角形 g, h) が消去される。次の形状演算における交線 q が図 1 (3) のように交わっているとすると、前の処理で求められた図 1 (3) の 3 角形はすべて捨てられ、再び多角形分割が行われ、まったく新たな 3 角形群が生成される (図 1 (4))。このように「大

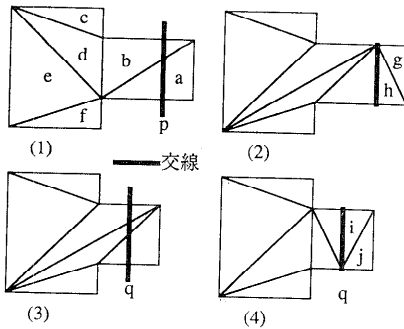


図1 大局的分割

Fig. 1 Global splitting of faces.

局的な分割」は、3 角形の数をも最小化できるが、不必要な多角形分割を引き起こす。同様のことは (b) の山口らの方法にも発生する。一方、「局所的な分割」では、この場合 a と b が分割されるだけで、3 角形 c, d, e, f に処理が及ぶことなく不必要な分割は発生しない。その代わり 3 角形の数是最小とはならず、一般的にその数はかなり増える。

3. 3 角形 BRep とゼロ 3 角形

このように 3 角形処理を導入する効果は大きいですが、多角形と 3 角形を併存させる限り、多角形分割の問題は避けられない。そこで、筆者らは第 4 のタイプであり、1 つの究極のモデルである 3 角形のみを用いた BRep を提案する。これは、(a) の多角形 BRep に対する対極のモデルであり、多角形分割の問題を根元的に排除することができる。3 角形はこれ以上原理的に分割不可能な根元的な基本図形である。そこで、3 角形のみを用いることにより非常に単純な形状表現となり、データ構造および処理アルゴリズムは、ある意味で究極的に単純化・簡略化される。

また近年、曲面を用いた BRep も実用化されつつある。しかしこのような BPep では、交線などを求めて形状演算を行うことは、よりいっそうの困難を伴う。そこで筆者らの立場としては、これらの曲面データを、まず 3 角形面を用いて近似表現し 3 角形 BRep とし、後はすべて 3 角形の枠組みの中で演算・処理しようとするものである。そして、曲面に対する高い表現・演算精度を実現するには、大量かつ微小な 3 角形の演算に耐える形状モデルが必要不可欠となってくる。このようなモデルを実現するにはその第 1 ステップとして、データ構造および処理アルゴリズムを最大限に単純化することが重要であると考えている。これ

が多角形を排除するもう 1 つの大きな理由である。

しかしながら、BRep の境界面を多角形から 3 角形のみになると、これまでに述べてきたように、データ構造および処理アルゴリズムは単純化されるが、データ量 (3 角形の数) は非常に増える。これは、形状演算などを行うと、分割が周辺の 3 角形に伝播し不必要な 3 角形分割が発生することによる (図 2 (1))。そこで、新たに面積ゼロ 3 角形を導入することにより、3 角形分割の伝播を解消した (図 2 (2))。面積ゼロ 3 角形とは、3 角形を構成する 3 つの頂点が同一直線上にある面積がゼロの 3 角形のことである。以下、これを単にゼロ 3 角形と呼び、通常的面積を持つ 3 角形を実 3 角形と呼ぶことにする。そして、ゼロ 3 角形を用いた 3 角形 BRep を、ゼロ 3 角形 BRep と呼ぶことにする。

図 2 (3) の例に示すように通常の 3 角形 BRep では、3 角形 a が交線 c によって分割されると、その隣接 3 角形 b にも分割が及ぶ。なおこの図では a の分割線は省略している。そして、次の交線 d の処理では 5 個の 3 角形との分割処理が行われる。一方、ゼロ 3 角形を用いると、図 2 (4) に示すように、交線 c により 3 角形 a が分割されても、a と b の間にゼロ 3 角形群が形成され、b に分割は及ばない。そこで、交線 d においては 3 角形 b のみの分割処理が行われるだけである。このように、ゼロ 3 角形を用いることにより 3 角形の数の増大を抑制することができる。さらに形状演算において、交線を求めて 3 角形を分割する処理はその処理量のかなりの部分を占めるが、ゼロ 3 角形はこの処理からまったく除外できるので大幅に演算量を減

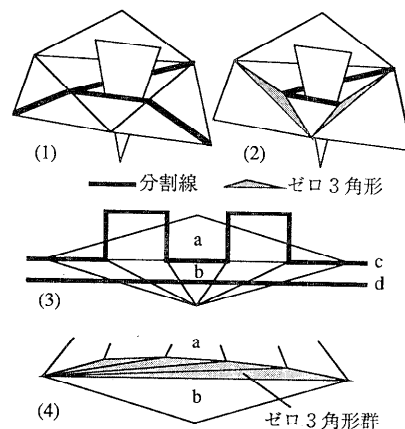


図2 3 角形分割とゼロ 3 角形

Fig. 2 Splitting of triangles and zero triangle.

らすことができる。すなわち、ゼロ 3 角形はデータ量の削減だけではなく処理の効率化にも寄与する。

この表現法のデータ構造は図 3 に示すように、3 角形面データ部と頂点データ部から構成される。3 角形面データは、3 つの頂点データへのポインタ ($v0, v1, v2$)、隣接する 3 つの 3 角形へのポインタ ($t0, t1, t2$) および実 3 角形とゼロ 3 角形とを区別するフラグ f から構成される。このように、ゼロ 3 角形 BRep では 1 次元固定長リストという非常に単純なデータ構造となる。また、エッジデータおよびエッジループは特に必要ではない。従って、多角形 BRep に比べてデータ構造が大幅に単純化・簡略化される。

4. 形状演算アルゴリズム

ゼロ 3 角形を用いた 3 角形 BRep における形状演算のアルゴリズムは、以下のような処理手順となる。ここでは、形状 A と形状 B との形状和をとる場合を想定する。

処理 1. 共通空間処理

前処理として、形状 A と形状 B が共に存在する直方体空間である共通空間を求める。そのためにはまず、形状 A および形状 B を包む包含直方体 ($x_{min}, y_{min}, z_{min}$)—($x_{max}, y_{max}, z_{max}$) を求める。すなわち、それぞれの形状における実 3 角形を順次取り出し、その頂点の X, Y, Z 座標値それぞれにおける最小値 ($x_{min}, y_{min}, z_{min}$) および最大値 ($x_{max}, y_{max}, z_{max}$) を求める。A および B の包含直方体値がそれぞれ (x_1, y_1, z_1)—(x_2, y_2, z_2), (x_3, y_3, z_3)—(x_4, y_4, z_4) であったとすると、両者の共通空間は、($\max(x_1, x_3), \max(y_1, y_3), \max(z_1, z_3)$)—($\min(x_2, x_4), \min(y_2, y_4), \min(z_2, z_4)$) となる。ここで、 $\min, \max$ はそれぞれ、かつこ内の数値の中の最小値、最大値を表す。

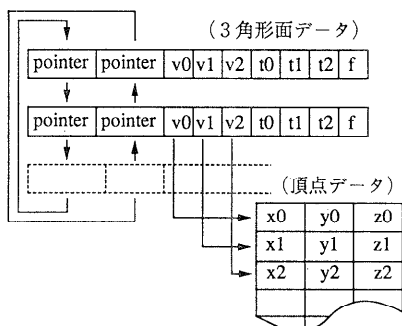


図 3 ゼロ 3 角形 BRep のデータ構造
Fig. 3 Data structure of BRep using Zero triangles.

次に、この共通空間に含まれる形状 A および形状 B の実 3 角形をそれぞれ取り出す。この処理もすべて頂点座標値の比較のみで行うことができる。この共通空間処理により、次のステップの処理である 3 角形の交差判定の回数を一般的にかなり減らすことができる。特に、ゼロ 3 角形 BRep では交線算出処理において、ゼロ 3 角形をまったく除外できるので、交差判定の回数がより一層削減できる。共通空間が存在しない場合は以下のステップの処理は行われない。

処理 2. 3 角形の分割

上記のステップにより取り出された実 3 角形において、形状 A の実 3 角形と形状 B の実 3 角形との交差判定を順次行い、両者が交わる場合はその交線を求める。そして、その交線においてそれぞれの 3 角形を分割する。3 角形の分割パターンは図 4 (1), (2) に示す 2 通りのみである。すなわち、

- (1) 交線が 3 角形の 2 辺と交わる。
- (2) 交線が 3 角形の 1 辺および頂点と交わる。

ゼロ 3 角形 BRep においては、図 4 (3) に示すように交線が辺上となる場合は 3 角形 a を分割する必要はない。これは後で述べるように、ゼロ 3 角形を用いて境界面接続が自由に容易にできるからである。一方、ゼロ 3 角形を用いない場合は分割する必要がある。また、3 角形 BRep においては、3 角形が同一平面上にある場合は、他の同一平面上にない 3 角形により分割処理が行われるので、何の処理も行わなくてよい。すなわち、図 4 (4) の例では 3 角形 a と c は同一平面上にあり、a と b が交わっている。ここでは a は b により分割されるので、a と c の組み合わせにおいては何の処理も行わなくてよい。

交線が 3 角形の辺と交わり (図 4 (1) の両端、図 4 (2) の片端)、かつその辺において隣接する 3 角形が

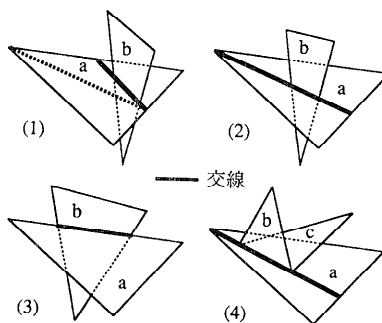


図 4 3 角形の交差パターン
Fig. 4 Intersectional pattern of two triangles.

ゼロ3三角形の場合、この交線と3三角形の辺との交点Vと一致する頂点が存在する可能性がある。そこで、頂点の重複を避けるために、この一致する頂点が存在するかどうかを、「同一点探索」により探す。図6(1)の例ではそのような頂点が存在し、Vはv2と一致している。

同一点探索のアルゴリズムを、図5にC言語風に示す。ここでは、交点 $V(x, y, z)$ は(分割される)実3三角形 t の辺 $(t \rightarrow v0, t \rightarrow v1)$ 上にあるとする(図6(1)参照)。 $t \rightarrow v0, t \rightarrow v1, t \rightarrow v2$ は、3三角形 t の3つの頂点(へのポイント)を表し、 $t \rightarrow t0, t \rightarrow t1, t \rightarrow t2$ はそれぞれ、3三角形 t とその辺 $(t \rightarrow v0, t \rightarrow v1), (t \rightarrow v0, t \rightarrow v2), (t \rightarrow v1, t \rightarrow v2)$ をはさんで隣接する3三角形(へのポイント)を表している(図6(1)参照)。また、 (u, v) という表記は点 u と点 v を結ぶ直線(辺)を表す。そして、このVと等しい頂点 $v2$ が存在するかどうかを探る。

まず、 $v0 = t \rightarrow v1$ とし、 t と隣接するゼロ3三角形 $t \rightarrow t0$ を t とする(ステップ1)。そして、この t を

```

v0 = t->v1;
ax = t->v0->x-v0->x;
ay = t->v0->y-v0->y;
az = t->v0->z-v0->z;
for ( t = t->t0; t->f == ZERO; ) {
    Rotate( t, v0 );
    v2 = t->v2;
    bx = v2->x-x;
    by = v2->y-y;
    bz = v2->z-z;
    d = ax*bx+ay*by+az*bz;
    if ( d < -E ) { t = t->t2; v0 = v2; }
    else if ( d < E ) break;
    else t = t->t1;
}
if ( t->f == ZERO ) return v2;
else return NULL;

```

図5 同一点探索アルゴリズム

Fig. 5 Algorithm of searching the identical vertex.

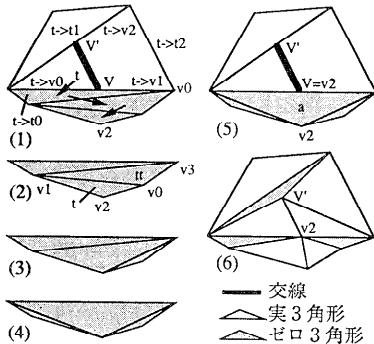


図6 位相変形

Fig. 6 Changing of the topology.

$t \rightarrow v0$ が $v0$ となるように回転する(ステップ2)。次に、 $v2 = t \rightarrow v2$ がVと等しいかどうかを判定し(ステップ3)、等しければ $v2$ を返して処理を終了する(ステップ5)。ここで、 E は十分に小さな正数である。 $v2$ がVと等しくなくかつ $v0$ 寄りにあれば、ステップ4を実行してステップ1に戻る。逆に、 $t \rightarrow v1$ 寄りにあればステップ6を実行してステップ1に戻る。以上のステップ1~6の処理が、 t がゼロ3三角形でなくなるまで繰り返された場合は、Vと等しい $v2$ が存在しなかったことになる。ここで、 $t \rightarrow f$ は、実3三角形(REAL)あるいはゼロ3三角形(ZERO)を示すフラグである。図6(1)の例では矢印の順にゼロ3三角形が探索され、Vと等しい $v2$ が求められる。

上記の同一点探索の結果に応じて、次の2種類の分割処理が行われる。すなわち、一致する頂点が存在しない場合は、ゼロ3三角形を用いて3三角形分割が行われる(図6(6)の V' 側)。一致する頂点が存在する場合は、隣接するゼロ3三角形の「位相変形」を行ってから3三角形の分割が行われる(図6(6)の $v2 = V$ 側)。ここでの位相変形操作は本質的なものではなく、必ずしも行う必要性はない。ここでは不必要なゼロ3三角形の発生をなくすためと、次のステップの3三角形消去処理においても位相変形操作が必要であるが、その回数を減らすために行っている。

次に、この位相変形操作のアルゴリズムを図7を用いて説明する。ここでは初期状態として、交点Vと一致する頂点は $v2$ であり、 $v2 = t \rightarrow v2$ となっている(図6(2)参照)。そして、ベクトル (ax, ay, az) は同一点探索で用いたものと同一である。

まず、3三角形 t とその辺 $(t \rightarrow v0, t \rightarrow v1)$ をはさん

```

v0 = t->v0;
v1 = t->v1;
t1 = t->t1;
t2 = t->t2;
for ( tt = t->t0; tt->f == ZERO; ) {
    Rotate( tt, v0 );
    v3 = tt->v1;
    tt0 = tt->t0;
    tt2 = tt->t2;
    t->v0 = v0; t->v1 = v3; t->v2 = v2;
    t->t0 = tt0; t->t1 = t1; t->t2 = tt2;
    tt->v0 = v3; tt->v1 = v1; tt->v2 = v2;
    tt->t0 = tt2; tt->t1 = t1; tt->t2 = t2;
    Neighbour( t2, t, tt );
    Neighbour( tt0, tt, t );
    bx = v3->x-v2->x;
    by = v3->y-v2->y;
    bz = v3->z-v2->z;
    d = ax*bx+ay*by+az*bz;
    if ( d < 0.0 ) { v0 = v3; t1 = t; t = tt; tt = tt2; }
    else { v1 = v3; t2 = tt; tt = tt0; }
}

```

図7 位相変形アルゴリズム

Fig. 7 Algorithm of changing the topology.

で隣接するゼロ 3 角形 $t \rightarrow t_0$ を tt とする (ステップ 1) (図 6 (2)). この tt を $tt \rightarrow v_0$ が v_0 となるように回転する (ステップ 2). そして, この t および tt をそれぞれ位相変形する (ステップ 3) (図 6 (3)). これにより, 3 角形 t_2 , tt_0 において隣接 3 角形がそれぞれ, t から tt , tt から t に変化するので変更する (ステップ 4). v_3 が v_2 を境にして v_0 寄りにあればステップ 5 を実行してステップ 1 に戻る. 逆に, v_3 が v_2 を境にして v_1 寄りにあればステップ 6 を実行してステップ 1 に戻る. 以上のステップ 1 ~ 6 の処理は tt がゼロ 3 角形でなくなるまで, すなわち元の分割される実 3 角形に戻るまで繰り返される.

このように位相変形を行うことにより, 例えば図 6 (2) ~ (4) に示すようにゼロ 3 角形の変形が行われ, 頂点 $v_2 (=V)$ と分割される 3 角形の辺 ($t \rightarrow v_0$, $t \rightarrow v_1$) で構成されるゼロ 3 角形 a が生成される (図 6 (5) の V 側). そして, このゼロ 3 角形 a は消去することができる, 3 角形の分割が行われる (図 6 (6) の v_2 側).

このように分割された 3 角形において, 交線と一致する辺に対しては, 交線フラグが ON にセットされる. 交線フラグは 1 つの 3 角形において 3 つあり, それぞれの辺に対応している. そして, 初期値はすべて OFF である. また, 3 角形には 1 つの内外フラグがあり, 内外判定ができる場合はそれが行われ, そのフラグが IN (内側) または OUT (外側) にセットされる. 例えば図 4 (1) ~ (3) における 3 角形 a は 3 角形 b に対して内外判定可能である. しかし, 図 4 (4) における a は b に対して内外判定ができず, 内外が不明となる. このような場合は内外フラグが NULL にセットされる. これらのフラグは, いずれも次のステップの 3 角形の消去を効率化するために用いられる.

処理 3. 3 角形の消去

まず, 形状 A および B の実 3 角形において, 他の形状の内部となる実 3 角形をすべて消去する. この処理では, 実 3 角形を 1 つ 1 つ取り出してその内外判定を行うと非常に処理の効率が悪くなるので, 交線フラグを利用して交線フラグが ON となっている辺で囲まれる, すなわち交線で囲まれる 3 角形群を求める. これらの 3 角形群は一括して処理することができる. そして, この 3 角形群において内外フラグが IN である 3 角形が 1 つでもあれば, この 3 角形群は内部となるので消去する. また, OUT である場合は消去は行わない. NULL の場合は不明であるので, ray casting

処理によりこの 3 角形群の内外判定を行い, 内部であれば消去する.

次に, この実 3 角形の消去に関連して不要となったゼロ 3 角形も消去する. 隣接 3 角形が 2 つ以上存在しない, すなわち隣接 3 角形へのポイントが 2 つ以上 NULL であるゼロ 3 角形は無条件に消去することができる (図 8 (1) 参照). しかし, 隣接 3 角形が 1 つ存在しないゼロ 3 角形の場合は簡単に消去できない場合がある. 例えば図 8 (2) のゼロ 3 角形 t の頂点 v_3 は辺 (v_0, v_1) 上にないので, これを消去すると裂け目ができる. ところで, 図 8 (2) における面 f と g はゼロ 3 角形によりつながっているが, 図 8 (7) のように切り離す必要がある. すなわち, 図 8 (2) のゼロ 3 角形群は消去する必要がある. このために「3 角形分割処理」において用いた「位相変形操作」をここでも行う. t を図 8 (4) の t' のように位相変形すると頂点 v_2 は辺 (v_0, v_1) 上となるので, 裂け目を作らずに消去することができる. そこで, 隣接 3 角形が 1 つ存在しないゼロ 3 角形に対しては「辺上頂点探索」を行い, 隣接 3 角形が存在しない辺上に頂点が存在するかどうかをチェックする. そして, このような頂点が存在すれば, 位相変形を行うことにより消去が可能となる.

この辺上頂点探索のアルゴリズムは, 図 9 に示す処理となり, 同一点探索に似た処理となる. ここでは図 8 (2) に示すように, 3 角形 t の隣接 3 角形が存在しない辺 (v_0, v_1) = ($t \rightarrow v_0$, $t \rightarrow v_1$) 上に頂点 v_2 が存在するかどうかを探索する. まず, 辺 (v_0, v_1) に対して, 3 角形 t の頂点 $v_2 = t \rightarrow v_2$ がどこに位置するかを調べる (ステップ 2). v_2 が (v_0, v_1) 上に存在すれば

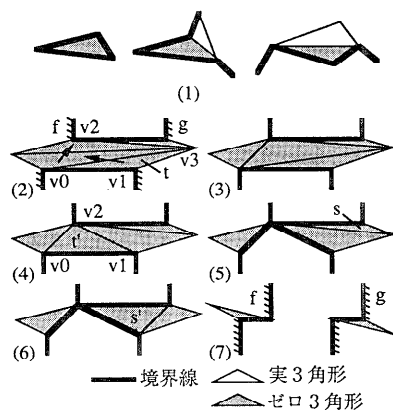


図 8 ゼロ 3 角形の消去
Fig. 8 Deletion of zero triangles.

```

v0 = t->v0;
v1 = t->v1;
ax = v1->x-v0->x;
ay = v1->y-v0->y;
az = v1->z-v0->z;
v = v0;
for (; t->f == ZERO; ) {
    v2 = t->v2;
    bx = v2->x-v0->x;
    by = v2->y-v0->y;
    bz = v2->z-v0->z;
    d = ax*bx+ay*by+az*bz;
    if ( d < -E ) { t = t->t2; v = v2; }
    else {
        bx = v2->x-v1->x;
        by = v2->y-v1->y;
        bz = v2->z-v1->z;
        d = ax*bx+ay*by+az*bz;
        if ( d < E ) break;
        else t = t->t1;
    }
    if ( t == NULL ) return NULL;
    Rotate( t, v );
}
if ( t->f == ZERO ) return v2;
else return NULL;

```

図 9 边上頂点探索アルゴリズム

Fig. 9 Algorithm of searching the vertex on a side.

ば処理を終了して v2 を返す (ステップ 4). v2 が (v0, v1) 上になくかつ v0 寄りにあれば, ステップ 3 の処理を行う. 逆に, v1 寄りにあれば, ステップ 5 の処理を行う. 次に, 新たな t において t->v0 が v となるように回転し (ステップ 6), ステップ 1 に戻る. 以上のステップ 1~6 の処理は (v0, v1) 上の頂点 v2 が見つかるか (ステップ 4 が成立). または t がゼロ 3 角形でなくなるまで繰り返される. 図 8 (2) の例では, 矢印の順にゼロ 3 角形が探索され, 边上頂点 v2 が求められる.

次に, このような頂点 v2 が存在していれば, 図 7 に示した位相変形操作を順次行い (図 8 (2)~(4)), v2 と隣接 3 角形が存在しない辺 (v0, v1) でゼロ 3 角形 t' を形成する (図 8 (4)). そして, このゼロ 3 角形 t' は消去することができる (図 8 (5)). このような頂点が存在しなければ, そのゼロ 3 角形は消去することができないのでそのまま残す. 同様に, 図 8 (5)におけるゼロ 3 角形 s も, 位相変形操作を行いゼロ 3 角形 s' を形成した後, 消去することができる (図 8 (5)~(7)). これにより, 面 f と g は完全に切り離すことができる.

処理 4. 境界面の接続

境界面接続 (3 角形接続) ではまず, 形状 A において, 隣接 3 角形が 1 つ以上存在しない 3 角形 a が取り出される. そして, 形状 B において, この a と少なくとも 1 つの頂点を共有し, かつ辺が同一直線上となる 3 角形 b が検索される (図 10 (1)). このような b が存在しない場合は, 別の a に対して同様の検索を行

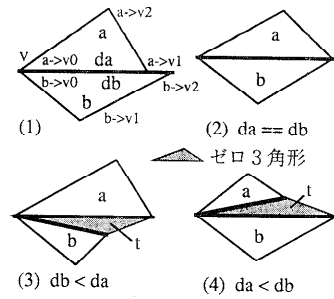


図 10 3 角形接続

Fig. 10 Connection of two triangles.

```

v0 = a->v0;
for (; ) {
    da = Length( a->v0, a->v1 );
    db = Length( b->v0, b->v2 );
    if ( fabs(da-db) < E ) {
        a->t0 = b;
        b->t1 = a;
        v = a->v1;
        if ( v == v0 ) break;
        a = NextA( a ); Rotate( a, v );
        b = NextB( b ); Rotate( b, v );
        if ( OnLine( a->v0, a->v1, b->v2 ) == NO ) break;
    } else if ( db < da ) {
        t = Trialloc();
        t->v0 = b->v2; t->v1 = a->v1; t->v2 = a->v0;
        t->t0 = NULL; t->t1 = b; t->t2 = a; t->f = ZERO;
        a->t0 = t;
        b->t1 = t;
        a = t;
        v = b->v2;
        b = NextB( b ); Rotate( b, v );
    } else {
        t = Trialloc();
        t->v0 = a->v1; t->v1 = b->v0; t->v2 = b->v2;
        t->t0 = a; t->t1 = NULL; t->t2 = b; t->f = ZERO;
        a->t0 = t;
        b->t1 = t;
        b = t;
        v = a->v1;
        a = NextA( a ); Rotate( a, v );
    }
}

```

図 11 3 角形接続アルゴリズム

Fig. 11 Algorithm of connecting two triangles.

い, 上記の条件を満たす a と b のペアが見つかるまで行われる.

次に, この a と b のペアを初期 3 角形として, 図 11 に示すアルゴリズムで 3 角形接続が行われる. このアルゴリズムでは初期状態として, a の頂点 a->v0 と b の頂点 b->v0 が一致しており, 辺 (a->v0, a->v1) と辺 (b->v0, b->v2) を接続する (図 10 (1)). まず, この接続する辺の長さ da, db を求める (ステップ 1). da と db が等しい時は, a と b を直接接続する (ステップ 2) (図 10 (2)). そして, 次の接続 3 角形のペア a と b を NextA, NextB ルーチンにより求める. この新しい a と b において, 頂点 a->v0 および b->v0 が v となるようにそれぞれ回転する (ステップ 4).

$db < da$ の場合には、新たにゼロ 3 角形 t を生成し、これを用いて a と b を接続する (ステップ 6) (図 10 (3)). そして、次の a は t となる. 次の b は NextB ルーチンにより求める. この新しい b において、頂点 $b \rightarrow v_0$ が v となるように回転する (ステップ 7). $da < db$ の場合も同様の処理を a と b を入れ替えて行う (ステップ 8, 9) (図 10 (4)).

以上のステップ 1~9 の処理は a が境界線を一周して元に戻るか (ステップ 3 が成立), ステップ 5 の条件式が成立するまで行われる. ステップ 5 は 3 角形 a の頂点 $a \rightarrow v_0$ と頂点 $a \rightarrow v_1$ を通る直線上に 3 角形 b の頂点 $b \rightarrow v_2$ が乗らない場合であり、両 3 角形は辺を共有せず接続できないので処理を終了する. そして、これらの処理はすべての a と b のペアが接続されるまで行われる. このようにゼロ 3 角形を用いることにより、接続する 3 角形相互の頂点が一致していなくてもよく、自由に 3 角形を接続することができる.

処理 5. 3 角形の統合

最後に、隣接する周辺 3 角形との形状チェックを行い、それらがより大きな 3 角形を形成するのであれば、これらの 3 角形を 1 つの 3 角形に統合する. この処理の主目的は 3 角形の数を減らすことである.

3 角形 BRep における 3 角形の頂点は外点と内点に分類することができる. 外点とはこれを頂点として共有する実 3 角形群において、その面の傾き (法線ベクトル) が 3 つ以上の異なる値を持つ頂点である. 一方、内点とはこれを頂点に持つ実 3 角形群において、その面の傾きが 2 つ以下の異なる値しか持たない頂点である. すなわち、外点はそれを消去すると形状の形が変化する頂点であり、内点はそれを消去しても形状の形が変化する頂点ではない頂点である. ここでは、すべての内点を消去することにより、3 角形の統合を行い、3 角形の数を最小化する.

まず最初に、頂点を順次調べ内点を取り出す. すなわち、頂点周りの実 3 角形の法線ベクトルをすべて調べ、その値が 2 つ以下の異なる値しか持たないのであれば内点となる. 次に、この内点を頂点に持つ 3 角形の形状を順次チェックし、統合できる 3 角形があれば統合を繰り返していく. 統合できるパターンは図 12 に示す 3 つのパターンがある. ここでは、消去する内点を V とする.

(1) のパターンは隣接する 2 つの 3 角形が作る 4 角形が凹でない場合である. この場合は、内点 V に集まる辺を減らすために、図 12 (1) のように位相変形を

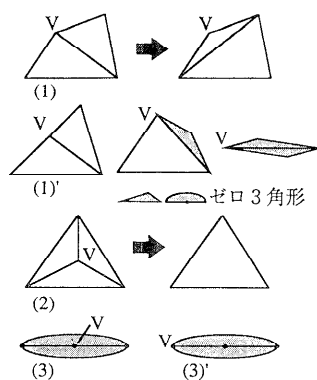


図 12 3 角形の統合

Fig. 12 Unification of triangles.

行う. 図 12 (1)' に示す例もすべて (1) のパターンに属する. (2) のパターンは、内点 V が 3 つの 3 角形で囲まれる場合である. この場合はこの 3 つの 3 角形を一度に統合し、1 つの大きな 3 角形を生成することができる. これにより、内点 V は消えることになる.

(3) は 2 つのゼロ 3 角形により形成されるパターンである. すなわち、隣接する 2 つのゼロ 3 角形が 2 つの辺を共有する場合である. この場合はこの 2 つの 3 角形を消去することができ、頂点 V も消去される. (3)' は (3) と同じパターンであり、2 つのゼロ 3 角形は消去されるが、 V は消去されない. 以上のようなパターンにおいて 3 角形の統合を繰り返し、内点が消去されるまで続ける. そして、このような処理によりすべての内点を消去する. これにより結果として 3 角形の数が最小化される. このようにゼロ 3 角形を用いることにより、局所的な簡単な操作の繰り返しにより、3 角形の数を最小化することができる.

以上が形状和のアルゴリズムである. 形状差 ($A - B$) および形状積 ($A * B$) の場合は処理 3 の 3 角形の消去のみが異なった処理となる. すなわち差の場合は、形状 B の内側となる形状 A の 3 角形および形状 A の外側となる形状 B の 3 角形を消去する. そして、形状 B の残った 3 角形面の表裏をひっくり返す処理を行う. また積の場合は、両形状の 3 角形それぞれにおいて、他の形状の外側となる 3 角形を消去する. このようにゼロ 3 角形を用いることにより、非常に単純かつ局所的なアルゴリズムで形状演算が実現できる.

5. 実験と考察

ゼロ 3 角形を用いた形状演算アルゴリズムの基本的性能を評価するために、以下の 4 項目を計算機実験に

より比較評価した。

(1) 処理時間 (CPU 時間)

(2) 生成された形状の 3 角形面数 (面数)

(3) 新たに生成された 3 角形の総数 (総増加面数)

(4) 生成された 3 角形の最小面積値

ここで、形状 A と形状 B における形状演算前の 3 角形面数を n_{A1} , n_{B1} , 形状演算時の 3 角形分割処理後の 3 角形面数を n_{A2} , n_{B2} とすると、形状演算における増加面数 n は、 $n = n_{A2} + n_{B2} - n_{A1} - n_{B1}$ で与えられる。そこで、すべての形状演算において増加した面数である総増加面数 N は、 n の総和で与えられる。

比較対象形状として次の 5 形状を取り上げた。特

に、(c), (d) の形状は、平面と曲面、穴、薄板、多くの凹凸など種々の複雑な形状の特徴を合わせ持つ。

(a) ハンドル 3 個の直方体の和差。外形の幅 560, 奥行き 200, 高さ 380 (図 13(1))。

(b) 格子 12 個の直方体 (幅 70, 奥行き 70, 高さ 500) の和 (図 13(2), (4))。

(c) 部品 1 プリミティブ 18 個の和差。外形の幅 800, 奥行き 400, 高さ 200 (図 13(5))。

(d) 部品を半分に切断する処理 (図 13(6))。

(e) 3 つのトーラスの和 大径 320, 小径 70 (図 13(3))。

なお、図 13(4) はゼロ 3 角形 BRep モデルからす

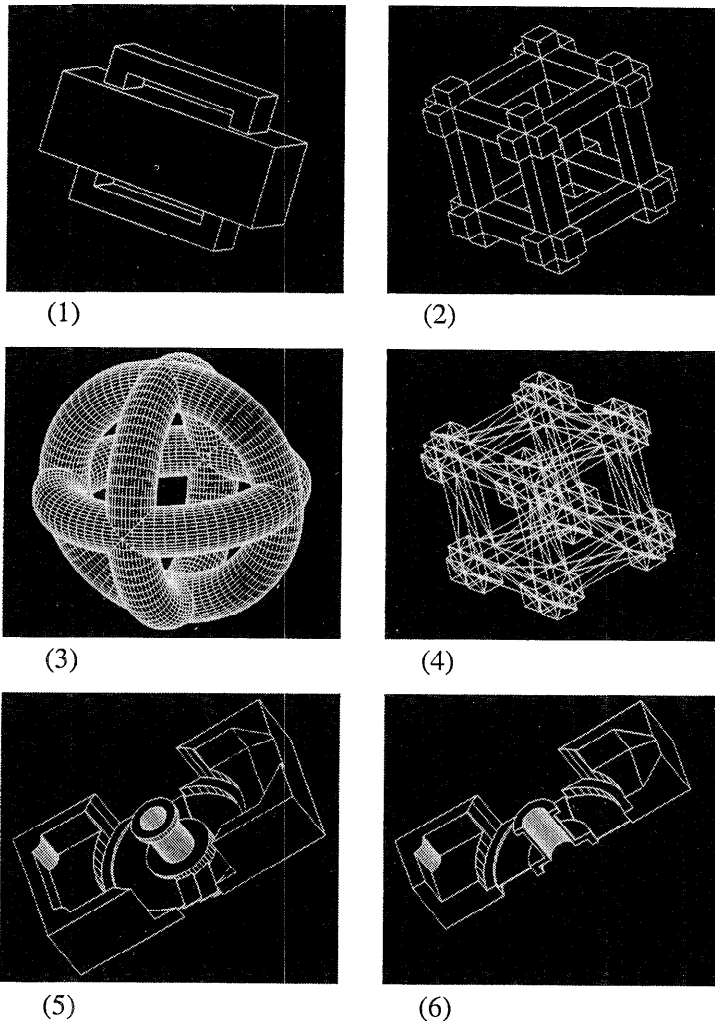


図 13 テスト形状
Fig. 13 Test objects.

すべての 3 角形面を、他はすべて、ゼロ 3 角形 BRep モデルから隠線処理した結果を表示したものである。

その結果は表 1 となった。ここで、Laidlaw の方法は多角形に基づいた形状演算を 3 角形に適用したものであり、「局所的な分割」である。Hubbard の方法は「大局的な分割」である。これらのデータは文献 7) から引用した。また、Real, Zero はそれぞれ、実 3 角形のみを用いた形状演算、ゼロ 3 角形を用いた形状演算である。(U)は 3 角形の統合処理を行うことを表し、これのないものはその処理を行っていない。かつこ内の数値はその数値におけるゼロ 3 角形の数を表している。

計算機は、HP 9000/755 CRX 24 (124 Mips, 40 Mflops, 128 Mbytes memory, 128 Mbytes swap) を使用し、プログラムはすべて C 言語で書き、表示は X-window を用いた。一方、Laidlaw および Hubbard の結果は、Sun SPARCstation 330 GX(16 Mips, 2.6 Mflops, 40 Mbytes memory, 327 Mbytes swap) を用いたものである。処理時間は我々の結果と直接比較することはできない。そこで、両計算機の Mips 比および Mflops 比がそれぞれ $124/16=7.8$, $40/2.6=15.4$ であることから、両者の処理速度比を 20 倍と想定する。そして、比較例も単純な形状であるハンドルー例だけであるので、Laidlaw および Hubbard との比較は参考的な比較であることを断っておく。

ハンドルー例において、Laidlaw と Real はどちらも「局所的な分割」を行っているにもかかわらず、Real の方が 3 角形面数が多くなっている。これは主に 3 角形の分割方法の違いによると推測される。すなわち、

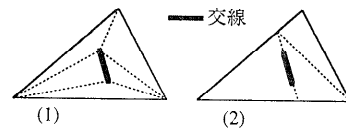


図 14 3 角形の分割方法

Fig. 14 Two methods of splitting a triangle.

Laidlaw の方法では交点の端が 3 角形内部にある場合は、図 14(1) に示すような 3 角形分割を行っている。このため、周辺の 3 角形にその分割が及ぶことが少ない。一方、Real および Zero では交点の端がどこにあろうとも、図 14(2) に示すような分割を行っている。そこで、必ず周辺の 3 角形に分割が波及し 3 角形の数がより増える。しかし、この分割処理は Laidlaw の方法に比べて単純であるので、Real の処理時間が短くなったと推測される。また、Zero はこのような分割法にもかかわらず Laidlaw よりも 3 角形面数が少なく、かつ Hubbard よりも処理時間が短い。

格子の例では、Real では 12 個の直方体の 3 角形面数 $12 \times 12 = 144$ が形状演算を行うと 5,560 にも増えている。Real はこのような単純な形状でも面数が非常に増えてしまい、そのために処理時間も非常に遅くなることが分かる。一方、Zero は Real に比べてかなり少ない 3 角形の増加で済み、処理時間も格段に短い。またすべての形状において、Zero(U) の方が Real(U) よりも処理時間が短い。これは、総増加面数の違いによるところが大きいと判断される。

以上から、ゼロ 3 角形には 3 角形分割の伝播をなくし、3 角形の数をかなり減少させる効果があることが分かる。そして、この分割伝播防止効果は処理時間の短縮にもかなり貢献することが確認された。また、3

表 1 処理時間とデータ量

Table 1 Processing time and data amount of the test objects.

(単位：処理時間一秒 面数, 総増加面数一面)

	アルゴリズム	処理時間	面数	総増加面数	最小面積
ハンドルー	Laidlaw	*0.53	262		
	Hubbard	*0.03	84		
	Real	0.14	378	668	8.04×10^{-3}
	Real (U)	0.14	84	668	8.04×10^{-3}
	Zero	0.01	194(75)	190(32)	2.33
	Zero (U)	0.01	84(5)	190(32)	2.33
格子	Real	20.90	5560	12856	1.40×10^{-8}
	Real (U)	1.01	336	4596	4.29×10^{-3}
	Zero	0.30	1248(432)	1344(0)	19.4
	Zero (U)	0.15	336(0)	1308(0)	13.9
部品	Real (U)	5.13	1992	10136	
	Zero (U)	1.74	1992(2)	6950(1459)	
部品断面	Real (U)	11.40	1042	5336	
	Zero (U)	2.37	1042(5)	1948(354)	
トラス	Real (U)	81.58	16896	5856	
	Zero (U)	60.09	16896(0)	4032(0)	

注) *はHP換算処理時間。原データは10.64, 0.65。

角形の統合処理はそれほど処理のオーバーヘッドがなく、データ量および処理時間をかなり削減できることが確認された。

不良3角形に関しては、Real および Real(U) ではかなり面積の小さな3角形を生成してしまう。また、文献 7) には具体的なデータは示されていないがこの問題が生じることが指摘されている。それに比べて、Zero および Zero(U) ではそのような3角形は生成されず、ゼロ3角形には不良3角形の生成防止効果もあることが確認された。

6. おわりに

面積ゼロ3角形を新たに導入し、3角形面のみを用いた BRep を構築した。そして、このゼロ3角形を用いた形状演算アルゴリズムを提案した。また計算機実験により、ゼロ3角形には次のような優れた効果があることを実証した。

- (1) 3角形分割の伝播をなくすことができ、3角形の数をかなり減らすことができる。
- (2) その結果、処理速度も速くなる。さらに、形状演算において3角形の分割処理は最も演算量を占めるが、ゼロ3角形は全く処理の対象外となるので、その演算量をかなり減らすことができ、処理がより一段と効率化される。
- (3) 不良3角形の生成を抑制できる。

この表現法は、(1)実3角形処理とゼロ3角形処理が多くの場合別処理となり、実3角形のみを用いた場合と比べて処理系が複雑になる。(2)多角形 BRep と比べると一般的にデータ量がかなり大きくなる。(1)に関しては、多角形処理よりはるかに単純となり、多角形の3角形分割処理を完全に排除できる点などを考えると、筆者らはこれをそれほどの欠点とは見なしていない。また、(2)に関しても、現在の計算機ハードウェア技術は加速度的に進歩しているので楽観している。

しかしながら、今回の他の手法との比較評価は十分ではない。今後の課題は、この表現法と他の代表的なモデリング法とを、より総合的かつ詳細に比較評価・検証を行うことである。そして、この評価結果がよければ、次のステップとして、この方法をハードウェア化・並列処理化して、リアルタイム処理に耐える高速処理系を実現する計画である。

参考文献

- 1) Baer, A., Eastman, C. and Henrion, M.: Geometric Modelling: a Survey, *Computer-Aided Design*, Vol. 11, No. 5, pp. 253-272 (1979).
- 2) Baumgart, B.G.: Geometric Modeling for Computer Vision, Rep. No. STAN-CS-74-463, Stanford Univ. (1974).
- 3) Braid, I.C.: The Synthesis of Solids Bounded by Many Faces, *Comm. ACM*, Vol. 18, No. 4, pp. 209-216 (1975).
- 4) Braid, I.C., Hillyard, R.C. and Stroud, I.A.: Stepwise Construction of Polyhedra in Geometric Modelling, *Mathematical Methods in Computer Graphics and Design*, Academic Press, pp. 123-141 (1980).
- 5) Eastman, C. and Weiler, K.: Geometric Modeling Using the Euler Operators, *Proc. First Ann. Conf. Computer Graphics CAD/CAM Systems*, M.I.T., pp. 248-259 (1979).
- 6) Hosaka, M., Kimura, F. and Kakishita, N.: A Unified Method for Processing Polyhedra, *Proc. 1974 IFIP Congress*, Stockholm, pp. 768-772 (1974).
- 7) Hubbard, P.M.: Constructive Solid Geometry for Triangulated Polyhedra, Tech. Rep. No. CS-90-07, Brown Univ. (1990).
- 8) Laidlaw, D.H., Trumbore, W.B. and Hughes, J.F.: Constructive Solid Geometry for Polyhedral Objects, *Proc. SIGGRAPH '86*, Vol. 20, No. 4, pp. 161-170 (1986).
- 9) Mantyla, M. and Sulonen, R.: GWS: A Solid Modeler with Euler Operators, *IEEE Computer Graphics and Applications*, Vol. 2, No. 7, pp. 17-31 (1982).
- 10) Mäntylä, M. and Tamminen, M.: Localized Set Operations for Solid Modeling, *ACM Computer Graphics*, Vol. 17, No. 3, pp. 279-288 (1983).
- 11) Requicha, A.A.G.: Representations for Rigid Solids: Theory, Methods, and Systems, *ACM Computing Surveys*, Vol. 12, No. 4, pp. 437-464 (1980).
- 12) Requicha, A.A.G. and Voelcker, H.B.: Solid Modeling: A Historical Summary and Contemporary Assessment, *IEEE Computer Graphics and Applications*, Vol. 2, No. 2, pp. 9-24 (1982).
- 13) Requicha, A.A.G. and Voelcker, H.B.: Solid Modeling: Current Status and Research Directions, *IEEE Computer Graphics and Applications*, Vol. 3, No. 7, pp. 25-37 (1983).
- 14) Voelcker, H.B. and Requicha, A.A.G.: Geometric Modeling of Mechanical Parts and Processes, *Computer*, Vol. 10, No. 2, pp. 48-57

- (1977).
- 15) Wördenweber, B.: Surface Triangulation for Picture Production, *IEEE Computer Graphics and Applications*, Vol. 3, No. 8, pp. 45-51 (1983).
- 16) Yamaguchi, F. and Tokieda, T.: Bridge Edge and Triangulation Approach in Solid Modeling, *Proc. Computer Graphics Tokyo 84*, pp. 44-65 (1984).
- 17) Yamaguchi, F. and Tokieda, T.: A Unified Algorithm for Boolean Shape Operations, *IEEE Computer Graphics and Applications*, Vol. 4, No. 6, pp. 24-37 (1984).
- 18) Yamaguchi, F.: A Unified Approach to Interference Problems Using a Triangle Processor, *Proc. SIGGRAPH '85*, Vol. 19, No. 3, pp. 141-149 (1985).
- 19) 荒川佳樹: 仮想空間における立体形状モデリング, '91 グラフィクスと CAD シンポジウム論文集, Vol. 91, No. 7, pp. 33-42 (1991).
- 20) 杉原厚吉, 伊理正夫: 計算誤差による暴走の心配のないソリッドモデラの提案, 情報処理学会論文誌, Vol. 28, No. 9, pp. 962-974 (1987).
(平成6年6月6日受付)
(平成6年11月17日採録)



荒川 佳樹 (正会員)

1954年生. 1978年早稲田大学理工学部工業経営学科卒業. 1980年同大学院理工学研究科修士課程修了. 同年松下電器産業(株)入社. 1990年郵政省通信総合研究所入所. 現在, 同研究所関西先端研究センター知覚機構研究室主任研究官. 画像・図形処理の研究に従事. 3次元形状モデリングに興味を持つ. 電子情報通信学会, 精密工学会各会員.