グラフィクスとCAD 91－13
（1998. 8. 28）

d次元多面体の集合演算アルゴリズム

尾下 真樹　黒木 進　牧之内 顕文

九州大学大学院システム情報科学研究科

概要

本論文では，我々の開発した，任意の次元における多面体同士の積・差・和集合を一般的に計算するアルゴリズムについて報告する．本アルゴリズムは，もとの多面体の位相構造をもとに集合演算結果の多面体の形状を計算するので，正しい位相構造を持った演算結果を得ることができる．

本アルゴリズムでは，多面体をその境界面の集合によって階層的に表現し，以下の 2 段階の手続きで集合演算を行なう．まず最初に，2 つの多面体同士の内包点・交点を計算して，集合演算結果を構成する頂点を求める．次に，もとの多面体の形状をもとにして，頂点から階層的に辺を構築していき，最終的に集合演算結果の d 次元多面体を得る．

An Algorithm of Boolean Set Operations of d-Dimensional Polyhedra

Masaki OSHITA, Susumu KUROKI, and Akifumi MAKINOUCHI

Graduate School of Information Science and Electrical Engineering
Kyushu University

Abstract

We propose an algorithm of Boolean set operations of d-dimensional polyhedra in a d-dimension space. Because this algorithm is based on topological structures of both polyhedra, it can calculate the answer that have valid topological structure.

In this algorithm, polyhedra are represented by their boundaries. And this algorithm has two phases. First, it computes included-points and cross-points of two polyhedra. Second, it constructs faces hierarchically by considering topology of original polyhedra.

1. はじめに

CADやComputer Graphics, アニメーション, Virtual Reality, 空間データベースなど, 立体モデルの利用が近年大きく広がっている. 立体モデルを利用する際に, 複数の立体の積・差・和集合として表現される立体や領域を求めることが必要とされる場合がある. 例えば, CADで用いられるCSGモデルでは, 立体を直方体・球・円柱・円錐・三角錐などの基本的要素の集合演算の組み合わせによって表現する.

CADやCGの分野では, Z-bufferやスキャンライン法を用いて集合演算によって表現された立体の2次元画像を作成する手法が多く研究されている[1]. しかし, こういった方法では, 集合演算結果の立体の正確な構造を得ることはできない. 多面体の積・差・和集合を一般的に求めるアルゴリズムは知られておらず, 多くの研究が行われている凸多面体の積集合演算[2]だけで解けるように問題を変形して解決に当たる場合が多い.

我々は, 高次元データベースへの幾何モデルの応用を検討しており[5], このような分野へ立体モデルを利用するためには, 問い合わせや演算の実現のために, 高次元空間における多面体同士の集合演算結果の位相構造を正確に計算することが必要となる. このような問題を解決するために, 我々は多面体の積・差・和集合を一般的に求めるアルゴリズムの開発を行った[6].

2. 定義とデータ表現

2.1. 多面体の定義

本アルゴリズムは, d次元空間における, 2つのd-多面体の積・差・和集合演算を取り扱う. 本アルゴリズムで扱うd-多面体は, 有界で単純なものとする. d-多面体は, その境界面である(d-1)-faceの集合によって表わされる. d-多面体は有界でなければいけないので, d-多面体の境界を表わす(d-1)-faceの集合は, 境界として閉じている必要がある. ただし, 境界面の連結数が1である必要はない(穴の空いている多面体であったも良い). また, d-多面体の境界であるそれぞれのk-face (0<k<d)も, d-多面体と同様に, その境界である(k-1)-faceの集合によって表現される. 従って, d-多面体は, その頂点集合から階層的に定義されることになる.

2.2. 集合演算結果の定義

ユークリッド空間においては, 点や平面は「厚み」をもたないため, 集合演算などの際にこれらの境界面上の領域をどのように扱うかが問題となる. 本論文では, 解と解法の簡略化のために, 集合演算の結果が必ずd-多面体(または空集合)になるような問題を取り扱う. 例えば, 境界面で隣接するd-多面体同士の積集合結果は, (d-1)-faceではなく, 空集合であると定義する.

d-多面体をPとおいた時, 多面体の境界上の領域を∂P, ∂Pを含まないPの外側の領域を$\bar{P}$, 同じく∂Pを含まないPの内側の領域を$(P)^\circ$と表す. これらの記号を用いて, それぞれの集合演算結果とその境界面を定義する.

積集合

2つのd-多面体をP_1, P_2とおいた時, それらの積集合の結果を,

$(P_1)^\circ \cap (P_2)^\circ$

と定義する. また, 積集合演算結果の多面体を表わす境界面は, 以下のようになる.

$\partial(P_1 \cap P_2)=(\partial P_1 \cap (P_2)^\circ) \cup (\partial P_2 \cap (P_1)^\circ)$

差集合

積集合と同様に, 差集合演算結果とその境界面を以下のよう定義する.

$P_1 - P_2 = P_1 \cap \bar{P}_2$

$\partial(P_1 - P_2)=(\partial P_1 \cap \bar{P}_2) \cup (\partial P_2 \cap (P_1)^\circ)$

和集合

積集合・差集合と同様に, 和集合演算結果とその境界面を以下のよう定義する.

$P_1 \cup P_2 = (P_1 \cap \bar{P}_2) \cup (\bar{P}_1 \cap P_2) \cup (P_1 \cap P_2)$

$\partial(P_1 \cup P_2)=(\partial P_1 \cap \bar{P}_2) \cup (\partial P_2 \cap \bar{P}_1)$

2.3. 集合演算結果の多面体の境界に対する定理

本アルゴリズムでは, 集合演算結果の多面体の境界面を頂点から階層的に求めていく. 従って, 以下の定理が重要になる.

定理1

集合演算結果の多面体の境界は, もとの2つのd-多面体の境界に含まれる.

証明

集合演算結果の定義より正しい. □

さらに，以下の定理を導くことができる．

定理2

集合演算結果の多面体の k-face は，次の 2 つのどちらかである．

・もとの多面体の k-face の部分集合

・もとの多面体の(k+1)-face と，もう一方の多面体の境界面の積集合

証明

もとの多面体のある m-face を $e \in \partial P_1$ とすると，e の $(P_2)^\circ$ または $\bar{P}_2$ との積集合は m-face になり，e の ∂P_2 との積集合は(m-1)-face となる．従って，k-face ができるのは，定理で挙げられている 2 つの場合のみとなる．

2.4. d-多面体(d-face)境界面の集合に関する定理

d-多面体の定義で述べたように，d-多面体の境界を表わす(d-1)-face の集合は，境界として閉じていなければならない．これは，集合演算結果の多面体を計算・作成する場合も同様である．ここで，(d-1)-face の集合があった時に，その集合がある d-多面体（k-face）の境界を表現しているかどうかを判定するための条件について述べる．

定理3

有界な k-多面体（k-face）の境界が n 個の(k-1)-face によって表わされているとする．この時，各(k-1)-face の境界であるすべての(k-2)-face は，少なくとも 2 つ以上の(k-1)-face の共通の境界となっている．（ただし，$k \geqq 2$）

証明

k-多面体(k-face)の境界の 1 つである(k-1)-face の境界に，他の(k-1)-face と共通しない(k-2)-face が存在するとすると，その(k-1)-face の表側と裏側は，他の(k-1)-face に遮られず，同一の領域となってしまう．従って，k-face が有界であるという条件に反する．ゆえに，そのような(k-1)-face は境界として存在せず，定理は正しい．□

3. 集合演算アルゴリズム

3.1. 概要

本アルゴリズムは，以下のような 2 つの手続きに分けることができる．

1．多面体同士の交差判定と 0-face の作成

2 つの多面体の交差点，および，もう一方の多面体に内包される多面体の頂点を計算する．そして，それらの点集合のうち，求める集合演算結果を構成する頂点を選択する．

2．(k-1)-face の集合から k-face を作成

もとの多面体の位相構造をもとに，既に求まっている(k-1)-face を境界とする k-face を作成する．この処理を階層的に繰り返し，最終的に演算結果の d-多面体を求める．

3.2. 交差判定

定理2より，もとの2つの多面体の全頂点の部分集合，および，2つの多面体の交点が，集合演算結果の多面体の頂点集合となる．これらの頂点集合を求めるために，一方の多面体の線分ともう一方の多面体の交差判定，同じく頂点と多面体の内包判定を行う必要がある．

頂点や線分と多面体との関係の判定に関しては，凸多面体の場合に特化したアルゴリズムが広く研究されている．本アルゴリズムでも，凸多面体の判定アルゴリズムを利用する．従って，そのためには，集合演算を行う多面体をあらかじめ凸胞複体に分割しておく必要がある．多面体の分割問題に関しても多くの研究が行われているので[3]，ここでは省略する．

凸胞複体と頂点の内包関係は，各凸胞と頂点の内包関係を調べることで判定できる．凸胞複体と線分の交差判定も，各境界面と線分の交差関係により判定できる．ただし，これらすべての組み合わせを判定すると，非常に大きな計算量が必要となってしまう．この問題の解決法に関しては，5 節で述べる．

本アルゴリズムでは，2 つの多面体の交差判定を行う際に，2 つの多面体のどの線分とどの face 同士が交差したかをきちんと判定する必要がある．図1のように，境界面のさらに face と交差している場合，その判定も必要となる．

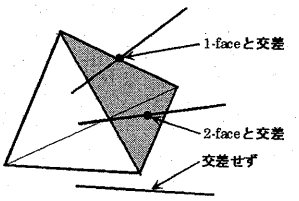

図1　境界面と線分の交差

3.3. 0-face の作成

2 つの多面体の内包点，および交点が求まったら，集合演算の定義より，演算結果の多面体の頂点集合を求めることができる．この頂点集合は，積・差・和それぞれの集合演算によって異なる．以下に，それぞれの集合演算結果を構成する頂点集合を示す．

A と B をそれぞれ多面体とし，A の全頂点集合を V_A，B の全頂点集合を V_Bとおく．また，A の 1-face と B の交点，B の 1-face と A の交点の集合を X とする．この時，それぞれの集合演算結果を構成する頂点の集合は，以下に示す通りである (図 2 はその具体例)．

・積集合 (A∩B) を構成する頂点集合
$(V_A \in (B)^\circ) \cup (V_B \in (A)^\circ) \cup X$

・差集合 (A−B) を構成する頂点集合
$(V_A \in \bar{B}) \cup (V_B \in (A)^\circ) \cup X$

・和集合 (A∪B) を構成する頂点集合
$(V_A \in \bar{B}) \cup (V_B \in \bar{A}) \cup X$

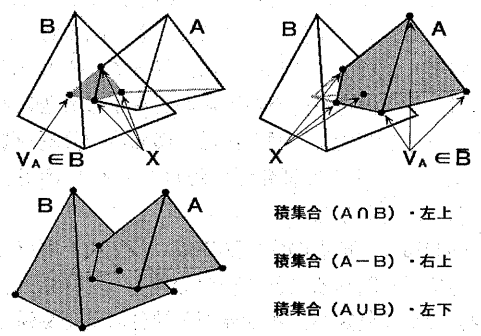

図 2　集合演算結果の多面体の頂点集

3.4. (k-1)-face の集合から k-face を作成

上と同じく定理 2 より，もとの多面体の k-face, (k+1)-face の部分集合として，集合演算結果の多面体の k-face を作成する．先述のように，各 k-face は，すでに作成された(k-1)-face の適切な集合を境界として作成される．

以下では，まず，k-face を作成するための(k-1)-face の集合の求め方を，定理 2 の 2 つのケースのそれぞれについて述べる．その後，求めた(k-1)-face の集合から，k-face を作成するアルゴリズムについて述べる．

3.4.1. 境界となる(k-1)-face ・ケース 1

最初に，定理 2 の最初のケースである，もとの多面体の k-face の部分集合として k-face を作成する場合の(k-1)-face の集合について述べる．この場合は，3.2 節の集合演算結果の多面体の頂点の作成時と同様に，積・差・和集合演算それぞれの場合で，選び出される(k-1)-face の集合が異なる．

上記の頂点の例と同様に，A と B をそれぞれ多面体とし，それらの k-face を f, そして f の内部にすでに作成された(k-1)-face を e とする．すると，f の内部に作成される k-face の境界となる (k-1)-face の集合 E は，以下のように表わされる (図 3 はその具体例)．

・積集合 (A∩B)
$^\exists f \in A,\ E = \{e \mid e \in \partial f \cap (B)^\circ,\ e \in (f)^\circ \cap \partial B\}$
$^\exists f \in B,\ E = \{e \mid e \in \partial f \cap (A)^\circ,\ e \in (f)^\circ \cap \partial A\}$

・差集合 (A−B)
$^\exists f \in A,\ E = \{e \mid e \in \partial f \cap \bar{B},\ e \in (f)^\circ \cap \partial B\}$
$^\exists f \in B,\ E = \{e \mid e \in \partial f \cap (A)^\circ,\ e \in (f)^\circ \cap \partial A\}$

・和集合 (A∪B)
$^\exists f \in A,\ E = \{e \mid e \in \partial f \cap \bar{B},\ e \in (f)^\circ \cap \partial B\}$
$^\exists f \in B,\ E = \{e \mid e \in \partial f \cap \bar{A},\ e \in (f)^\circ \cap \partial A\}$

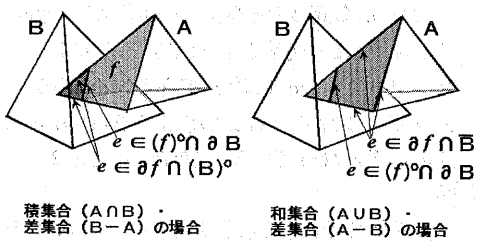

図 3　f の内部に作成される 2-face の境界集合

3.4.2. 境界となる(k-1)-face ・ケース 2

次に，定理 2 の 2 つ目のケースである，もとの多面体の(k+1)-face と，もう一方の多面体の境界面の積集合として k-face を作成する場合の(k-1)-face の集合について述べる．こちらは，集合演算の種類に関わらず，同一の集合が挙げられる．前と同様の表現を用いて，以下のように表わせる (図 4 はその具体例)．

$^\exists f_1 \in A,\ ^\exists f_2 \in B,$
$E = \{e \mid e \in \partial f_1 \cap (f_2)^\circ,\ e \in (f_1)^\circ \cap \partial f_2\}$

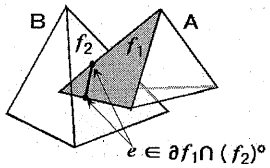

図4　fの内部に作成される 1-face の境界集合

3.4.3. (k-1)-face の集合から k-face を作成 (k≧ 2)

上の 2 つの方法で求めた，(k-1)-face の集合 E から，新しく作成する k-face の境界としての条件を満たす部分集合を求めるアルゴリズムを述べる．ここでは，2.5.2 で示した条件を用いる．つまり，それぞれの(k-1)-face の境界である(k-2)-face に注目し，k-face の境界となる(k-1)-face を集めるのである．ただし，この方法は k≧ 2 の場合でしか用いることができない．1-face（線分）の境界を見つける方法は，次節で述べる．

以下が，そのアルゴリズムの具体的な手続きである．

1. 初期状態を S = {境界を構成する(k-1)-face の集合}, i=1 とする．
2. 空リスト F_i を準備する．また，C（境界チェック用リスト）を空にする．
3. S から 1 つの辺 f を取り出して，F_i に追加．そして，f の境界である(k-2)-face のすべてを C に追加．
4. C が空になるまで以下の処理を繰り返す．
   (1) C から 1 つの(k-2)-face f を取り出す．S 中の辺で，f を境界として持つものを探す．
   (2) 見つかった (k-1)-face を f とする．
   (3) f を F_i に追加する．そして，F_i の全境界と C の全要素で共通するものを C から削除し，C に共通するものがなかった f の境界を C に新たに追加する．
5. 4.の処理で，連結した境界面（(k-1)-face）の集合 F_i が得られた．ここで，まだ S に辺が残っていれば，i++して再び 2.に戻り，次の組を作成する．
6. 以上の操作により，i 組の連結した境界面の集合が求まった．それぞれの集合を境界面として，解となる face が作成される．ただし，2 組以上の解が得られた時には，ある face が別の face の穴を表す場合がある．そこで，各 face 同士の関係を調べ，ある face が別の face の穴になっていれば，それらの境界面の和集合をとり，合せて 1 つの face とする．

3.4.4. 0-face の集合から 1-face を作成

先述のように，上記の方法は k≧ 2 の場合しか用いることができない．ここでは，k=1 の場合の方法について述べる．

多面体のある 1-face と，もう一方の多面体の交点，および，多面体に内包される 1-face の端点が得られた場合，まず，それらの 0-face の集合を順番に整列する．そして，順番に並んでいる点集合を端から順番に 2 つずつ組にして，その 2 点からなる線分を見ていき，3.3.1 や 3.3.2 と同様に，実行する集合演算の種類に応じて求める領域内にあるものを 1-face として作成する．

3.5. d-多面体の作成

解の d-多面体も，3.4 の処理と同様に(d-1)-face の集合から計算することで求まる．この場合も，解として得られる多面体の連結数は 1 とは限らず，演算結果が穴の空いた多面体になったり，複数の独立な多面体の集合になる可能性がある．

4. データ構造

ここでは，本アルゴリズムを実現する際に必要となるデータ構造について，簡単に説明する．本アルゴリズムでは，多面体，およびすべての k-face を，その境界である(k-1)-face の集合によって表すため，全次元の face を表すデータノードと，各 k-face から境界である(k-1)-face へのリンクが必要となる．0-face は，座標値を持つ．

さらに，本アルゴリズムでは，集合演算結果の k-face を作成する際に，すでに作成された(k-1)-face がもとの多面体のどの face の内部に存在するか，という情報が必要となる．従って，face を作成する際には，その face を含むもとの多面体の face ともリンクでむすぶ必要がある．

また，交差判定に凸胞複体を用いる場合などは，それらの情報も表現する必要がある．これについては省略する．

5. 計算効率

本節では，本アルゴリズムの効率に関して，簡単に述べる．d-多面体Pの含む頂点（0-face）数を v，線分（1-face）数を e，境界面（(d-1)-face）数を h とおく．また，それらを含む全次元の face の総数を f とする．

3.2 節で述べたように，すべての頂点や線分と多面体の関係を調べようとすると，多面体同士の交差判定に最も計算の手間がかかる．線分と多面体の交差判定を行うには，線分と多面体の各境界面との判定を行う必要があり，そのままでは，各線分ごとに $O(h)$ の計算量が必要となる．しかし，ここで，各境界面を R-tree[4]などの領域探索に適した形で記録しておく事で，この処理を $O(\log h)$ に減らすことができる．よって，全線分の交差判定の計算量は $O(e \log h)$ となる．頂点と多面体の内包関係の判定に関しては，多面体のすべて交点が求まっていれば，1 つの頂点の内包関係を調べることで，頂点グラフをたどるだけで全頂点の内包関係を判定することができる．よって，内包判定の計算量は $O(v)$ となる．

次に，k-face の内部に face を作成する際の計算量は，face の内部に存在する(k-1)-face の数 m に比例する．ここで m はデータ量とは無関係なので，$O(1)$ と見なすことができる．各 face ごとにこの処理を行うので，全 face を作成する処理の計算量は，$O(f)$ である．

以上により，全体の計算量は $O(f+e \log h)$ となる．

6. まとめ

本文では，我々の開発した凸胞複体として表現された多面体同士の集合演算を行なうアルゴリズムについて報告した．本アルゴリズムによって，任意次元の多面体の積・差・和集合演算結果の形状を求めることが可能になった．このことにより，より一般的な立体モデルの利用が考えられる．我々は，高次元空間データベースにおける物体のモデリングと問い合わせオペレーションの実装に，本アルゴリズムの応用を検討している．

しかし，本アルゴリズムを実行するためには，4 節で設計したような，凸胞複体の全 face ノードの関係を表わすデータ構造や，それらと演算結果の多面体の face ノードとの間の相互リンクなど，一般の利用には冗長な情報が必要であるため，より簡潔で効率的なデータ構造へ改良が今後の課題となっている．また，現在のアルゴリズムでは，境界面上の領域は多面体に含まれないとしているので，例えば隣接する多面体の積集合は空集合となってしまう．隣接する多面体同士の積集合として隣接面が選られるなど，境界面上の領域も含めた扱いに対応する事ができれば，より一般的なアルゴリズムとなるだろう．

参考文献

[1] Ari Rappoport, Steven Spitz, *"Interactive Boolean Operations for Conceptual Design of 3-D Solids"*, Computer Graphics Proceedings, Annual Conference Series,1997

[2] Franco P. Preparata, Michael lan Shamos, *"Computational Geometry : An Introduction"*, Springer-Verlag, 1985

[3] Bernard Chazelle, *"Convex Partitions of Polyhedra : A Lower Bound And Worst-Case Algorithm"*, SIAM J.Comput., Vol.13, No.3,August 1984, pp.488-507

[4] A Guttman, "*R-tree : a dynamic index structure for spatial serching*", Proc. ACM SIGMOD Int. Conf. On Managemente of Data, pp.47-57, 1984

[5] 黒木進，王霊哲，牧之内顕文，「４次元データベースシステム Hawks における４次元空間データ型の設計」，電子情報通信学会信学技報 DE96-83, January 1997, pp.55-60

[6] 尾下 真樹，「時空間データモデル Universe における時空間演算アルゴリズムの設計と実装」，九州大学工学部情報工学科卒業論文，1998