DEWS2004 4-B-05

空間データベース基盤システム Hawk's Eye の空間データモデル HA-face complex の効果的な実装

陸 応亮† 田中 美智子† 金子 邦彦‡ 牧之内顕文‡

†九州大学 大学院 システム情報科学研究院 〒812-8581 福岡市東区箱崎 6-10-1

E-mail: †{ riku, tanaka } @db.is.kyushu-u.ac.jp, ‡{ kaneko, akifumi } @is.kyushu-u.ac.jp

あらまし 本論文では，我々が研究開発を行ってきている空間データベース基盤システム Hawk's Eye の空間データモデルの詳細設計と実装について述べる．Hawk's Eye には，新しい空間データモデルとして，HA-face complex モデルが実装されている．HA-face complex モデルでは，超平面アレンジメントの face の複体として，空間物の位置と形を表現する．超平面アレンジメントの face とは，ある次元の空間を，複数の超平面で区切ったときにできる区画のことである．本論文では，HA-face complex モデルのオブジェクトデータベース上への実装に焦点をあてる．HA-face complex そのものは複雑な構造を持つが，データベース上では，超平面，0 次元の face, independent HA-face（HA-face complex を face をノードとするような lattice 構造で表現したときの極大元の face のこと），MBR の 4 つのデータの集まりとして表現できる．0 次元の face と independent HA-face には，+，-，0 からなる位置ベクトル(Position Vector)があるが，位置ベクトルをそのままデータベースに格納するのは無駄が多い．我々は，次元が 1 以上であるような independent HA-face の位置ベクトルのうち冗長な+と-を，符号 r で置き換え，0 次元の face の位置ベクトルの中の+と-を全て符号 d で置き換えた符号ベクトル(Sign Vector)の考え方を導入し，可能な限りコンパクトに HA-face complex を格納する方式を考案した．本論文では，データベース内への符号ベクトルの格納法，符号ベクトルと位置ベクトルの相互変換法，単純に位置ベクトルを格納した場合と，符号ベクトルを格納した場合とのデータサイズの見積もり比較を行う．

キーワード 空間 DB,オブジェクト指向 DB,多次元 DB,計算幾何学

Design and Implementation of a Spatial Data Model HA-Face Complex and a Kernel for Spatial Database Systems Hawk's Eye

Yingliang Lu† Michiko Tanaka† Kunihiko KANEKO‡ and Akifumi MAKINOUCHI‡

†Graduate School of Information Science and Electrical Engineering, Kyushu University 6-10-1 Hakozaki,

Higashi-Ku Fukuoka, 812-8581 Japan

E-mail: †{ riku, tanaka } @db.is.kyushu-u.ac.jp, ‡{ kaneko, akifumi } @is.kyushu-u.ac.jp

Abstract In this paper, we describe the geometric calculation algorithm about the spatial database system with which we have been doing research and development . *HA-face complex* model is implemented on an object database system as a new space data model by *Hawk's Eye*. the face of hyperplane arrangement expressing the position and form of a spatial objects of various dimensions , which is the division made when the space of a certain dimension is divided at two or more hyperplane .

Keyword spatial DB, object database system, spatial query processing, hyperplane arrangement, computational geometry

1．はじめに

空間データベースの重要性は高く，種々の研究が行われてきた[1][2][3]．我々は，この状況を踏まえ，空間データベース構築のための基盤システム Hawk's Eye の研究・開発を行ってきた．Hawk's Eye は，多角形や立体などの広がりを持った空間物を含む種々の次元の空間データを扱う空間データベースを，容易に構築できるようにするための基盤ソフトウエアシステムであり，空間データの格納・幾何計算・検索に関する種々の基本機能を提供する．Hawk's Eye には，新しい空間データモデルとして，HA-face complex モデルが実装されている．このモデルは，次元に特化しない空間データモデルであって，本質的に，任意の次元の空間内の，任意の次元の空間物を扱うことができる．

空間データベースでは，空間データの幾何計算[1]が重要な機能である．ここでいう幾何計算とは，2 つの異なった空間物が交差しているか,包含しているか,離れるかを判定したり，あるいは，交差領域の形を正確に求める計算である．例えば，土地利用状況，行政区画，交通網などの地形データが入ったデータベースにおいて,「ある道路とある道路は,どこで交差していか」,「ある県とある道路はどこで交差しているか」など,複数の空間データを組み合わせたような処理では,空間物同士の幾何計算が実行される．

我々は，今までに，HA-face complex モデル上で動く効率の良い幾何計算アルゴリズムである localized divide-and-conquer の提案を行ってきた[9].3 次元以上の幾何計算アルゴリズムとしては，従来から，逐次添加法[5]と呼ばれる超平面アレンジメント構成アルゴリズムを使って，一度超平面アレンジメントを構成し，その結果を使って幾何計算を行う方式が知られてきた．超平面アレンジメントに基礎を置くアルゴリズムは，3 次元以上の各種の幾何計算が可能であるという点で，2 次元でのみ動く各種の幾何計算アルゴリズム（Half-space intersection 法など）を凌駕し，交差部分の形を求めることができる点で，単なる交差判定（true/false 値を返す）線形計画法，整数計画法を凌駕する．Localized divide-and-conquer アルゴリズムは，既存の超平面アレンジメント構成アルゴリズムに基礎を置き，HA-face complex の特質をうまく利用して，格段の高速化を達成している．Localized divide-and-conquer アルゴリズムは，超平面アレンジメントで無く，幾何計算の中間結果として cell graph という独自の構造を作り上げることに特長がある．HA-face complex モデルでは，超平面アレンジメントの face の複体として，空間物の位置と形を表現する．超平面アレンジメントの face とは,ある次元の空間を複数の超平面で区切ったときにできる区画のことである.超平面アレンジメントを構成するのではなく,Cell graph を構築することで，空間物を構成する face の部分に処理を局在化させるのが localized divide-and-conquer アルゴリズムのアイデアである．Cell graph の利用によって，従来法よりも格段に高速な幾何計算が可能となった[9]．

HA-face complex のディスク上でのサイズが重要な研究課題である．HA-face complex を格納するのに必要なディスク量と，ディスク上の HA-face complex を読み書きする I/O コストはサイズに比例して増大するので，ディスク上では，可能な限りコンパクトな形式で格納しておくことは重要である．Localized divide-and-conquer アルゴリズムでは，各 face に対して付けられた+，-，0 からなる位置ベクトル(Position Vector)を使う．位置ベクトルの長さは，空間物を構成する超平面の数であり,複雑な空間物では,長さが数百や数千を超えることもありえる．位置ベクトルの大部分は+，-であり，0 の登場数は少ないが，+と-の登場はランダムであるため,符号語の登場頻度の偏りを利用して圧縮するという，情報理論的なアプローチを使っての圧縮はこのままでは困難である．

HA-face complex の各 face には，本来，交差する超平面と交差しない超平面とがある．交差しない超平面は，本質的には，localized divide-and-conquer アルゴリズムの処理には影響を与えない．現在までの我々の試作[9]では,交差する超平面と交差しない超平面とを区別せずに,データベースに格納してきた.つまり，データベースには全てを格納しておき，localized divide-and-conquer アルゴリズムの処理の時点でふるい落とすことにしてきた．現在までの実験と考察の結果，交差する超平面と比べて，交差しない超平面の方が大部分を占めることから，無駄なデータベース I/O 処理に時間の多くが費やされていることが分かってきた.この課題を解決するために,「データベースの格納では，交差する超平面と交差しない超平面を区別して格納する」ことを着想した．

我々は，位置ベクトルの中の+，-，0 のうち，face 同士の位相関係の表現にとっては冗長な+と-とを，別の符号で置き換えるというアイデアを提案する．次元が 1 以上であるような independent HA-face の位置ベクトルのうち冗長な+と-を，符号 r で置き換え，0 次元の faceの位置ベクトルの中の+と-を全て符号 dで置き換えた符号ベクトル(Sign Vector)の考え方を導入する．符号ベクトルをデータベースに格納するという方針で，HA-face complex の格納を行う．符号ベクトルでは，大半が r になり，+，-，0 の頻度は少ないという性質を持つので，簡単に圧縮することができる．空間物は，1 つの HA-face complex で表現され，データベース中に格納される．幾何計算では，HA-face complex が，その符号ベクトルとともにデータベースから実メモリ上に読み込まれ，符号ベクトルは位置ベクトルに変換される．幾何計算の localized divide-and-conquer アルゴリズムが実行された後，計算結果である HA-face complex を新たにデータベースに格納することが必要な場合には，位置ベクトルから符号ベクトルへの変換が行われれる．以上のように幾何計算の時点では，符号ベクトルと位置ベクトル間の変換が必要であるが，この処理は少ない計算コストで行うことができ，符号ベクトルを圧縮できることによる効果の方が高い．

本論文では，データベース内への符号ベクトルの格納法，符号ベクトルと位置ベクトルの相互変換法，単純に位置ベクトルを格納した場合と，符号ベクトル格納した場合とのデータサイズの見積もり比較を行う．本論文の構成は以下の通りである．2 章では，HA-face complex と位置ベクトルを簡単に説明しながら，符号ベクトルの考え方を導入する．3 章では，符号ベクトルと位置ベクトルの相互の変換について,幾何計算アルゴリズムとの関係に触れながら説明する．4 章は，HA-face complex のデータサイズを見積もりを行い，単純に位置ベクトルを格納した場合と，符号ベクトルを格納した場合を比較する．

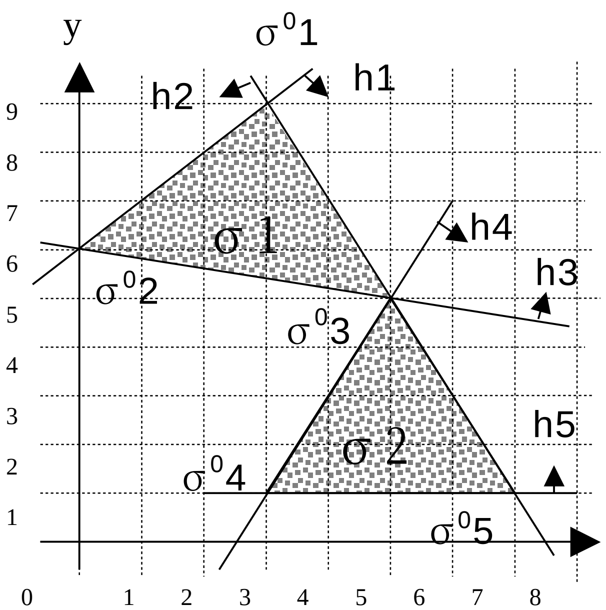

図１．HA-face complex の例

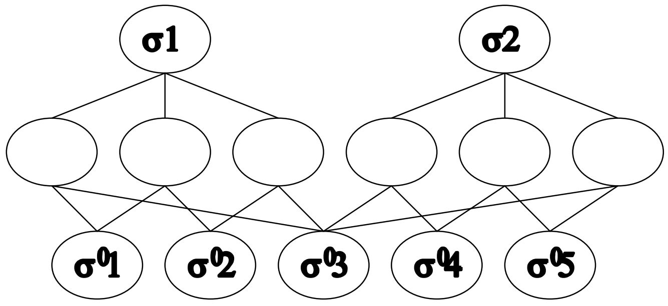

図2. 図1のHA-face complex のグラフ表現

(a) 0-HA-face の位置ベクトル

0-HA-face	位置ベクトル
$\sigma^0 1$	[0 0 + - +]
$\sigma^0 2$	[0 + 0 - +]
$\sigma^0 3$	[+ 0 0 0 +]
$\sigma^0 4$	[+ + - 0 0]
$\sigma^0 5$	[+ 0 - + 0]

(b) Independent HA-face の位置ベクトル

	位置ベクトル
σ1	[+ + + - +]
σ2	[+ + - + +]

表1. 図1の HA-face complex の位置ベクトル

2．HA-face complex と符号ベクトル

2.1. 超平面アレンジメントと HA-face complex

超平面アレンジメントは，空間の分割を表現するために用いられてきた幾何構造[5]である．N次元空間 R^N における N-1 次元の超平面の集合 H は，R^N を 0 から N 次元の区画に分割する．出来た区画は，互いに重なりあうことは無い．この分割のことを H のアレンジメント A(H) (the hyperplane arrangement induced by H) という．出来たそれぞれの区画は face と呼ばれる．face という言葉は，これ以外の意味も持つことがあるから，混乱を防ぐために，以下，超平面アレンジメントの区画である face のことを，HA-face と呼ぶことにする．超平面アレンジメント A(H) は，$(0 \leqq k \leqq d-1)$ なる k に対して，k-HA-face と (k+1)-HA-face との間の全ての接続関係を蓄えた接続グラフ(incidence graph)によって表現できる[5]．接続グラフのノードは HA-face を表し，エッジは HA-face 間の接続関係を表す．

超平面アレンジメントの HA-face を貼り合わせることで，いかような空間物であろうとも，その位置と形を表現することができる[9]．このアイデアを使って，種々の次元の空間物を，同一の形式で表現することができるようになる．現実世界にある空間物は，その境界（例えば，立体であるならその表面）を含むことから，空間物を表現する HA-face 集合も，空間物の境界部分を要素として含む．そこで，空間物を表現するモデルを，超平面アレンジメントの HA-face の複体（complex）として定義[9]する．つまり，k 次元の空間物を，0 から k までの次元の HA-face の複体として表現する．複体とは，本来，空間物における辺と点の相対性と密接な関係がある数学的概念のことであるが，ここでは，複体という言葉を次の意味で使う．

- HA-face の集合 Γ について，Γ が，Γ の任意の要素 k-HA-face (k>0) と接続関係にある全ての (k-1)-HA-face を含むとき，Γ のことを HA-face 複体（HA-face complex）と呼ぶ．

図１には，２つの３角形が貼り合わさった１つの図形の HA-face complex の例を示している．ここには，２個の 2-HA-face σ1，σ2，5 個の 0-HA-face $\sigma^0 1$，$\sigma^0 2$，$\sigma^0 3$，$\sigma^0 4$，$\sigma^0 5$ と，これら 0-HA-face をつなぐ 6 個の 1-HA-face という種々の次元の HA-face の複体として，この図形が表現されている．これら HA-face の関係は，図２に示したようなグラフで表現できる．このグラフでは，σ1，σ2 が極大元である．このように他の(k+1)-HA-face と接続していないような k-HA-face のことを Independent HA-face と呼ぶ[9]．

2.2. HA-face と符号ベクトル

HA-face complex[9] の各 HA-face は，超平面のとの位置関係により位置ベクトル(Position Vector)を持つ．各超平面は，φ=0 という形式の線形式（例えば図１の超平面 h1 は，x-y+6=0）であり，φ>0 である領域が表側，φ<0 である領域が裏側である．位置ベクトルでは HA-face が超平面の表側にあるときに+を，裏側にあるとき-を,超平面上にあるときに 0 を割り当てる．位置ベクトルの長さは，HA-face complex を表現する超平面の数である.位置ベクトルは,定義から次のような性質を持つ．

- N 次元空間では，ある 0 以上から N-2 次元以下の

HA-face を包含する超平面は，最低でも N-k 個あり，N-k 個以上のこともありえる．従って，その HA-face の位置ベクトルは N-k 個以上の 0 を持つ．
- N 次元空間での N-1 次元の HA-face は，1 つの HA-face にしか包含されない．従って，位置ベクトルは 1 個の 0 を持つ．
- N 次元空間での N 次元の HA-face は，いかなる超平面とも交差しない．従って，位置ベクトルは 0 を含まない．

位置ベクトルは，Localized divide-and-conquer アルゴリズムで使用されるが，位置ベクトルをそのままの形で，データベースに格納するのは得策ではない．位置ベクトルの中の+，-，0 のうち，+，-の中には，HA-face の表現にとっては冗長なものがある．例えば，図 1 の HA-face σ1 の表現には，σ1 を取り囲んでいる h1，h2，h3 の 3 つの超平面に対する+，-値 [+ + +] だけで十分であり，その意味で，σ1 と接しているだけであったり，σ1 とは離れている超平面 h4 と h5 に対する分は冗長である．一方，0-HA-face の位置ベクトルでは 0 だけが必要である.例えば，図 1 の $\sigma^0 1$ では，h1 と h2 だけが必要であり，他の超平面は不要である．以上のアイデアから，次に定めるような符号ベクトル(Sign Vector)を導入する．

1 次元以上の HA-face に対して

位置ベクトルの+，-，0 のうち冗長な+と-を r で置き換えたものを符号ベクトルとする．符号ベクトルは，+，-，0，r の 4 つの符号(sign)から構成される．k 次元 HA-face（k≧1）の符号の意味は次のように定める

+　の意味

HA-face に接続していて次元が 1 つ低い HA-face が，対応する超平面の上にあり，かつ HA-face 自身が超平面の表側にある．

-　の意味

HA-face に接続していて次元が 1 つ低い HA-face が，対応する超平面の上にあり，かつ HA-face 自身が超平面の裏側にある．

0　の意味

HA-face は，対応する超平面に包含されている．

r（redundant）の意味

上記のいずれにも当てはまらない．つまり，HA-face の表現にとって冗長である

0 次元の HA-face に対して

位置ベクトルの+,-,0 のうち+と-を全て d で置き換えたものを符号ベクトルとする．0 次元 HA-face の符号ベクトルは，0，d の 2 つの符号から構成される．それぞれの符号の意味は，0 次元 HA-face と，対応する超平面の位置関係により，次のように定める．

0 次元 HA-face の符号(sign)

0　の意味

0-HA-face が，対応する超平面の上にある．

d（disjoint）の意味

0-HA-face が，対応する超平面の上にない．

表 2．図 1 の HA-face complex の表現

(a) 0-HA-face

0-HA-face	座標	符号ベクトル
$\sigma^0 1$	(3 , 9)	[0 0 d d d]
$\sigma^0 2$	(0 , 6)	[0 d 0 d d]
$\sigma^0 3$	(5 , 5)	[d 0 0 0 d]
$\sigma^0 4$	(3 , 1)	[d d d 0 0]
$\sigma^0 5$	(7 , 1)	[d 0 d d 0]

(b) Independent HA-face

	符号ベクトル	接続関係
σ1	[+ + + r r]	[1 2 3]
σ2	[r + r + +]	[3 4 5]

(c) Minimum Bounding Box

	Minimum Bounding Box の形状
MBR	(0 , 1)　(7 , 9)

(d) 超平面の方程式

h1	x – y + 6 = 0
h2	2x + y – 15 = 0
h3	x/5 + y – 6 = 0
h4	2x – y – 5 = 0
h5	y – 1 = 0

2.3. HA-face complex のﾒﾓﾘ上での表現

符号ベクトルを導入した場合の HA-face complex を表現するデータ構造について説明を行う．表 2 に示すように，1 つの HA-face complex は，超平面，0 次元の face，independent HA-face，MBR の 4 つのデータで表現する．超平面と independent HA-face の符号ベクトルは，空間物の位置と形の表現に必要であり，このことが，HA-face complex の核となるアイデアである．表 2 (a) は，空間物の頂点（つまり 0 次元 HA-face）の座標を格納するために設けたデータ項目である．空間物の位置と形状の空間属性（3 次元空間ならば x，y，z の 3 つの値）を，空間物の頂点の座標で表現する．頂点の座標をそのままデータベースに格納するというアプローチの他に，Constraint Database で採用されているように，データベース内には超平面の係数を格納しておき，頂点の座標は超平面の交点計算で求める[2]というアプローチもある．前者は，交点計算に伴う計算誤差の問題を回避できる．HA-face complex の実装では，前者の立場に立ち，次のアイデアを採る．

- 頂点の座標を格納するために，表 2(a)のように 0-HA-face をデータ項目として定める
- 超平面の方程式は，方程式の係数を格納するのではなく，「超平面が通過する 次元-HA-face がどれか」を格納することで，間接的に方程式の係数を表現する．

なお，MBR 及び independent HA-face の中にある「接

続関係」のデータは，localized divide-and-conquer の高速化のためにあり，現在までの予備実験では，高速化の効果が確認できている．

2.4. 0-HA-face の符号ベクトルのデータベース上での表現

本研究での中心となる課題は,表 2 のような HA-face complex について,データベース上にいかなる形式で格納することが最も有利であるかを検討することにある．符号ベクトルでは，大半が r あるいは d になり，+，-，0 の頻度は少ないという性質を持つので，簡単に圧縮することができる．3.4 節で議論するように 1 次元以上の HA-face の符号ベクトルについては問題は無いと考えている.N 次元空間中の 0 次元 HA-face の符号ベクトルは，N 個とは決まっておらず,N 個以上の 0 を持つことから，0 次元 HA-face の格納法として,下記に示す 2 つの案を考えた．案 1 は，N 個を超える 0 があった場合も，すべてを格納する方針であり，案 2 は,不要なものを省いて,N 個を格納する方針である．

案 1

N 次元空間中の 0 次元 HA-face の符号ベクトルについて，「0 である位置」を全て格納する．実際の格納では，0 次元 HA-face と independent HA-face の接続関係に応じて，「N 個の 0 の位置を格納したレコード」を，0-HA-face に接続する independent HA-face の数だけ作る．例えば図 1 の 0 次元 HA-face $\sigma^0 3$ は，σ1 と σ2 の両方に接続していることから，$\sigma^0 3$ の符号ベクトル [d 0 0 0 d] の 0 の位置 {2, 3, 4} をデータベースに格納する．

案 2

符号ベクトルに N 個を超える 0 が含まれるのは,N+1 個以上の超平面が同じ 0 次元 HA-face で交差している場合なのだが，その 0 次元 HA-face に交差している超平面の中で超平面番号が最も小さい N 個を選んで，その N 個の超平面番号のみをデータベースに格納する．例えば,図 1 のように超平面 h2,h3,h4 が同じ頂点で交差している時，0 次元 HA-face $\sigma^0 3$ の符号ベクトルが[d 0 0 d d] であると見なして,データベースには,{2,3} を格納する．

3．幾何計算アルゴリズムの振る舞い

3.1. cell graph 構築

Localized divide-and-conquer アルゴリズムの最初の段階では,各 independent HA-face ごとに cell graph を構築するという処理を行う．cell graph と

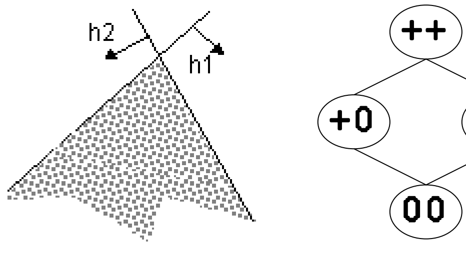

図 3：初期 cell graph

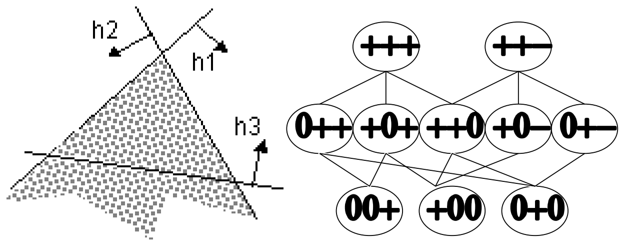

図 4：初期 cell graph に h3 を追加した graph

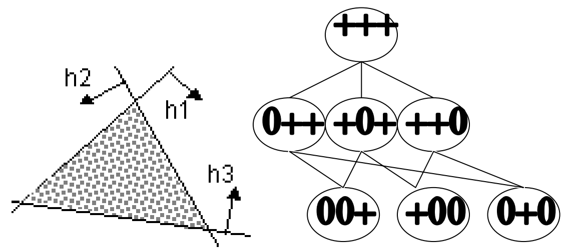

図 5：h1，h2，h3 から得られる cell graph

は，independent HA-face の輪郭を構成する HA-face 間及びもとの Independent HA-face 間の相互の接続関係を表現したグラフであり，各ノードは HA-face を表す.また,各ノードは位置ベクトルを属性として持つ．

cell graph を構築する為に，最初に初期 cell graph を作成する．初期 cell graph は，空間次元 N と，independent HA-face の次元 dim により，直接的に求めることができる．初期 cell graph は，dim 次元の超立方体と同型であり，各ノードは長さ N の位置ベクトルを持つ．図 1 の σ1 の場合，その初期 cell graph は図 3 のように，2 次元の正方形と同型であり，長さ 2 の位置ベクトルを持つ．その後，初期 cell graph に，冗長でない残りの超平面を追加し(図 4)，independent HA-face に含まれていない部分を削除することを繰り返すことで，cell graph を作成する(図 5)．

こうして，cell graph が出来上がったら，0 次元と 1 次元の HA-face が属性として持つ位置ベクトルについて，冗長な超平面に対する符号を補っておく．0 次元 HA-face については，データベース上にある符号ベクトルと対応を取る．その後，0 次元 HA-face の情報を使いながら，1 次元 HA-face に関して，位置ベクトル補完処理を行う（アルゴリズムは図 6)．

以上で説明した cell graph 構築処理について，2.4 で提示した 2 つの案での振る舞いの違いを調べると，以下に示すように，「0 次元の HA-face について，データベース上にある符号ベクトルと対応を取る」という

処理の部分で違いが生じる（案１の③と案２の③）.

案１：

① independent HA-face の符号ベクトルから，「r」以外の符号に対応する超平面を得る.

② ①で得た超平面を使って cell graph を構築する

③ 0 次元の HA-face について，データベース上にある符号ベクトルと対応を取る．データベース中の符号ベクトルは，「①で得た超平面について，必ず N 個の 0 がある（N は空間次元）という性質があり，必ず対応が取れる.

④ １次元 HA-face に関して，位置ベクトル補完処理を行う

案２：

① independent HA-face の符号ベクトルから，「r」以外の符号に対応する超平面を得る.

② ①で得た超平面を使って cell graph を構築する

③ 0 次元の HA-face について，データベース上にある符号ベクトルと対応を取る．位置ベクトルだけを使って、必ずしも対応が取れるとは限らない．対応が取れない場合には，超平面の方程式を使って，座標値を近似的に求めて，これを，データベース内の座標値と比較して，対応を取ることを行う.

④ １次元 HA-face に関して，位置ベクトル補完処理を行う

以上で示すように,案１の方では処理が簡単であり,案２では，空間座標値を使っての選択処理を行う必要が出てくる.従って,我々は案１の考え方を採用する.

```
fillup_1_HA_face(P1,P01,P02)
  入力：1-HA-face の位置ベクトル  P1=[p1_1・・・p1_Nh]
        P に接続している 0-HA-face の位置ベクトル
        P01=[p01_1・・・p01_Nh]
        P に接続している 0-HA-face の位置ベクトル
        P02=[p02_1・・・p02_Nh]
  出力：1-HA-face に符号を補ったベクトル
 1.for i ← 1 to Nh
 2.  do if  p1_i = “？”
 3.      then  if  p01_i = “0” and p02_i = “0”
 4.               then p1_i ← “0”
 5.               else p1_i← “d”
```

図 6．1 次元 HA-face の位置ベクトル補完処理

3.2. 幾何演算の結果の選択

以下，2 つの空間物の交わりを求めるオペレーションである intersection について説明する．二つの HA-face complex の intersection を求める場合，まず，3.1 節で説明した処理手順に従って，片方の HA-face complex 中の各 independent HA-face について cell graph を構築する．その後，もう一つの HA-face complex の超平面を cell graph に追加し，新しいグラフを作るという処理を行う．図 7 は，図１に示した HA-face complex（二つの independent HA-face σ1，σ2 からなる）と，もう一つの HA-face complex（independent HA-faceσ3 からなる）の intersection を示したものである．この時，σ1 の cell graph，σ2 の cell graph の両方に，超平面 h6,h7,h8 が添加され，新しいグラフが出来る．このグラフのノードは HA-face を表しており，位置ベクトルを持っている.

次に，生成されたグラフのノード（ノードは HA-face を表現）の中から，２つの空間物の両方に交わっている HA-face が選び出される．この選び出しの処理では，ノードの属性である位置ベクトルが使用される．図 6 の例では，元の空間物の HA-face であるσ1，σ3 の符号ベクトルと，生成されたグラフのノードの位置ベクトルとの比較により，σ4 と $\sigma^0 6$ などが intersection の解であることが分かる．一方，σ6 のように位置ベクトルの h6 に対する値が-であるものは，本来，σ3 の符号ベクトルの h6 に対する符号が+であったから intersection の解とはならないことが分かる．σ1 とσ3 の intersection は 3 つの 0-HA-face($\sigma^0 6$ $\sigma^0 7$ $\sigma^0 8$)，3 つの 1 次元 HA-face，1 つの independent HA-face(σ4)からなっている． σ2 とσ3 の intersection についても，同様に求められる(表 3).

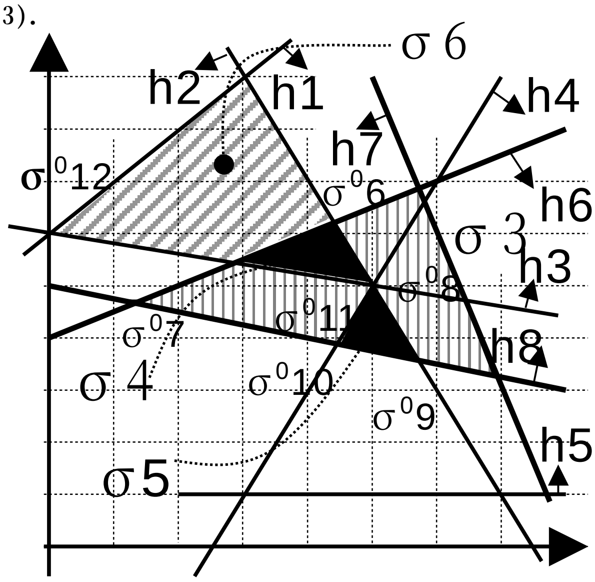

図 7 ２つの空間物の積を求める intersection の処理

表 3．Intersection での HA-face の位置ベクトル例

（1）independent HA-face

Independent HA-face	位置ベクトル
σ4	[+ + + + + +] (h1,h2,h3,h6,h7,h8)
σ5	[+ + + + + +] (h2,h4,h5,h6,h7,h8)

（2）0-HA-face

0-HA-face	位置ベクトル
$\sigma^0 6$	[+ 0 + 0 + +] (h1,h2,h3,h6,h7,h8)
$\sigma^0 7$	[+ + 0 0 + +] (h1,h2,h3,h6,h7,h8)
$\sigma^0 8$	[+ 0 0 + + +] (h1,h2,h3,h6,h7,h8)
$\sigma^0 9$	[0 + + + + 0] (h2,h4,h5,h6,h7,h8)
$\sigma^0 10$	[+ 0 + + + 0] (h2,h4,h5,h6,h7,h8)
$\sigma^0 11$	[0 0 + + + +] (h2,h4,h5,h6,h7,h8)

3.3 位置ベクトルの処理

最後に，求まった位置ベクトルを，符号ベクトルに変換する処理を行う．処理の順序としては，まず，0次元 HA-face を先に行う．0 次元 HA-face では，符号は 0，d の 2 通りであり，位置ベクトルの+，- 値を全て d に変換するという操作を行う．１次元以上の HA-face については,図 8 に示したアルゴリズムで行う．

```
make_sign_vector_R(P,dim,NIDSet)
    入力：independent HA-face の位置ベクトル
        P=[p1・・・pNh]
        independent HA-face の次元 dim
        接続する 0-HA-face の番号の集合 NIDSet
    出力：independent HA-face の位置ベクトルで,
        冗長な部分を r で置き換えたベクトル
    for i ←1 to Nh
    do if  is_num0_more_than_face_dim(
                dim,i,NIDSet) = false
          then pi ← “r”

is_num0_more_than_face_dim(dim,hid,NIDSet)
    入力：independent HA-face の次元 dim
        超平面番号 hid
        0-HA-face の番号の集合 NIDSet
    出力：超平面 hid に対する符号が”0”である
        0-HA-face が dim 個以上あれば true ，なけ
        れば false を返す
    NIDSet の要素をキューQ に入れる
    count ← 0
    while Q≠φ
      do Q の先頭の要素を取り出し k とする
    番号 k の 0-HA-face の位置ベクトルを P=[p1・・・
      pNh]とする
          if  Phid = “0”
            then  count ← count + 1
               if count ≧ dim
                 then  return true
    return false
```

図 8　１次元以上の HA-face に対する処理

更に，データベース上には，同じ符号ベクトルを複数保存することはしないので，同じ符号ベクトルがあった場合は，その中の1つだけを保存する．また，この時に，接続関係を書き換える処理も行う．例えば,下記の$\sigma^0 8$, $\sigma^0 11$ は同じなので，[d 0 0 0 d d d d]を一つだけ保存する．接続関係で，$\sigma^0 11$ と書かれている個所は，$\sigma^0 8$ と書き直す．

$\sigma^0 8$ の符号ベクトル[d 0 0 0 d d d d]

$\sigma^0 11$ の符号ベクトル[d 0 0 0 d d d d]

以上で，independent HA-face と 0-HA-face の符号ベクトルが作成される．出力は，表 4 の様になる．

3.4 Independent HA-face の符号ベクトルのデータベース上での表現

N 次元空間での k 次元の HA-face （但し k＞0）は，N- k 個の 0 を持つと決まっている．2．4 章では 0 次元 HA-face の格納法の案を複数提示したが，１次元以上の HA-face では,このような問題は無い．１次元以上の HA-Face のデータ格納法は,以下で説明する単純な方式を考えている．Independent HA-face の符号ベクトルは，0，+，-，r の 4 通りであるが，大部分が r であるので，0，+，-の値を覚えるという方針である．まず，+，-，0 は，表 5 のように 2bit で表現する．但し，「11」は，r のための区切り記号である．

(1) 実メモリ上にある符号ベクトルの先頭からから符号を順に読む

(2) 0，+，-に対しては，「00」，「01」，「10」をデータベースに格納する．

(3) r に対しては，連続する r の長さを数え，長さが 16 以下の時は「11」「長さ-1」を 6bit のデータとして格納する(図参照)．長さが 16 以上のときは，[長さ/16]個の「111111」を書き込んだ後に，「11」「(長さ mod 16) -1」を 6bit のデータとして格納するという案を考えている．例えば，Independent HA-Face 符号ベクトルを(+ r r r r r r − + r r r) とすると，この処理を行った後は，01 11 0101 10 01 11 0010 である．

表 4． Intersection の計算結果例

(1) independent HA-face

Independent HA-face	符号ベクトル
σ4	[r + + r r + r r]
σ5	[r + r + r r r +]

(2) 0-HA-face

0-HA-face	符号ベクトル
$\sigma^0 6$	[d 0 d d d 0 d d]
$\sigma^0 7$	[d d 0 d d 0 d d]
$\sigma^0 8$	[d 0 0 0 d d d d]
$\sigma^0 9$	[d 0 d d d d d 0]
$\sigma^0 10$	[d d d 0 d d d 0]
$\sigma^0 11$	[d 0 0 0 d d d d]

(3) 接続関係

Independent HA-face	接続関係
σ4	[$\sigma^0 6$, $\sigma^0 7$, $\sigma^0 8$]
σ5	[$\sigma^0 9$, $\sigma^0 10$, $\sigma^0 8$]

表 5. １次元以上の HA-face のデータベース上での符号 bit パターン

bit パターン	00	01	10	11
意味	0	+	-	区切り

4. HA-face complex データサイズ見積もり

まず HA-face Complex のサイズ見積もり式は次のように考えた．

「超平面数」×N×(N+1) ×8
+「Independent HA-face 数」×[ceil(「冗長で無い超平面数」/32)]× (*6/32*)
(冗長で無い超平面数が 16 以下の時は「(6/32)」の項は「1」になる
+「0-HA-face 数」×*N*×*4*
+「Independent HA-face に接する 0-HA-faces 平均数」× 4
+ N × 2 × 8　　**(式 4.1)**

一方，位置ベクトルをそのままの形でデータベースに格納した場合には，HA-face Complex のサイズは次のように見積もることができる．

「超平面数」× N ×（N + 1）× 8
+「Independent HA-face 数」×ceil(「超平面数」/32)
+「0-HA-face 数」× *ceil(「超平面数」/32) × 8*
+「Independent HA-face に接する 0-HA-faces 平均数 × 4
+ N × 2 × 8 **(式 4.2)**

以上の式を使い，次元が 2，Independent HA-face に接する 0-HA-faces 平均数が 2 とし，超平面数，Independent HA-face 数，0-HA-face 数は表 6 にある値を利用して，上の 2 式に代入して比較を行ってみる．HA-face complex の特質からいって，立体や領域図形など，広がりを持った図形の場合でも，次元に関係なく同様の傾向を示すと考えるので，実験データを使って，最も簡単な 2 次元で見積もり計算を試してみる．計算結果は表 6 に示しているように超平面数とともに前者の効果が良くなる．以上のように，見積もり式では効果が確認できたが，今後実際のデータを使っての効果の確認や，見積もり式の正しさの確認を行っていく予定である．

表 6 データのサイズ見積もり（単位は KB）

超平面数	50	100	150	200	300
Independent HAfaces	663	2,752	6,140	10,990	24,856
0-HA-faces	874	3,528	7,803	13,902	31,297
位置ベクトルのまま格納	30	239	604	1,535	4,805
符号ベクトルを格納	19	83	187	368	936

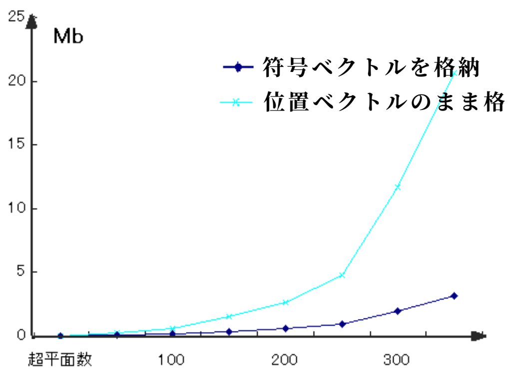

図 9 データサイズの比較

5. おわりに

本論文では，空間データベース Hawk's Eye の空間データモデル HA-face complex のデータベース上での実現について報告を行った．交差しない超平面群と交差する超平面群を区別するビット列（符号ベクトル）を導入した．また Localized divide-and-conquer アルゴリズムの中に，超平面が交差するか交差しないかを高速に判別するアルゴリズム(3.3 節)を組み込み，アルゴリズムの出力として「符号ベクトル」が得られるようにした．このことは，冗長な超平面を区別してデータベースに格納することに必要であった．以上の結果，データベースサイズの削減が可能になったのが本研究の成果である．位置ベクトルのままではベクトルデータの圧縮は難しいが，符号ベクトルは，登場する符号の大部分が r なので，簡単な方法でデータ量を削減できる．

今回提案の方式は，データ量の削減に効果があることを見積もり式で示したが，今後，削減の効果を実際の実験によって確認していきたい．また，我々が提案してきた localized divide-and-conquer アルゴリズムは，今回提案した位置ベクトルの導入の結果，さらなる変更を行うことで，格段の高速化ができると考えており，その設計と実装も今後の課題である．

参 考 文 献

[1] Agnes Voisard,Benoit David:"A Database Perspective on Geospatial Data Modeling",IEEE TKDE,Vol.14,No.2,pp.226-243,2002.

[2] Grunback,S.,Rigaux,P.,and Segoufin,L:"The DedaleSystem for Complex Spatial Queries",Proc.1998 SIGMOD,pp.213-224,1998.

[3] R.H.Guting:"An Introduction to Spatial Database Systems" VLDB Journal,vol.3,no.4,pp357-400,1994.

[4] David P.Dobkin,and Ayellet Tal,Efficient and Small Representation of Line Arrangements with Applications,ACM Symposium on Computational Geometry,pp.293-301,2001

[5] H. Edelsbrunner and J.O'Rourke and R. Seidel, Constructing arrangements of lines and hyperplanes with applications, SIAM J. Comput. vol.15, pp.341-363. 1986.

[6] Herbert Edelsbrunner, Algorithms in Combinatorial Geometry", Springer-Verlag, 1987.

[7] J.E. Goodman and J.O'Rourke,editors.Handbook of Discrete and Computational Geometry,CRC Press LLC,BocaRaton,FL,1997.

[8] Handbook of Computational Geometry, Elsevier Science Publishers B.V. North-Holland, Amsterdam, 2000

[9] 金子邦彦，牧之内顕文，" 超平面アレンジメントに基づく多次元空間幾何アルゴリズムの実装と評価 "，情報処理学会研究報告 2003-DBS-131, pp.219-226, 2003.