空間データモデル Cell Complex の空間データベースシステム格納法

陸 応亮[†] 金子邦彦[†]
田中美智子[†] 牧之内顕文[†]

空間データモデルとしての cell complex は，任意次元の空間データを，一般的に記述できるという良さがある．本論文では，空間データモデル cell complex を空間データベースシステムに格納する際のデータサイズを小さくするために，cell complex 内の cell の位相情報を表す cell の符号ベクトル（Cell Position Vector; CPV）を使う．CPV は，Herbert が提案している点の符号ベクトル（Position Vector）[18] を基礎として，我々が独自に定義を行った．データベースへの格納では，個々の CPV を圧縮して，Compressed Cell Position Vector（CCPV）を作り，全体のデータサイズを小さくする．我々は CPV のデータサイズを圧縮するため，ランレングスアルゴリズムを基礎とした，モディファイドランレングス（modified run length）アルゴリズムを開発した．データサイズの評価のために，CCPV での cell complex の格納と，VRML とのデータサイズの比較実験などを実施した．実験では，2 つの VRML データを使い，VRML ファイルに比べて CCPV のデータサイズがおよそ 50%小さいことを示した．

A Storage Method of the Spatial Data Model Cell Complex on Spatial Database Systems

YL. Lu,[†] Kunihiko Kaneko,[†] Michiko Tanaka[†]
and Akifumi Makinouchi[†]

Spatial data model cell complex can do the unification of the arbitrary dimension spatial data and can calculate the topology of cell complexes efficiently by their incidence graph in any dimension. In this paper, we describe a way named as Compressed Cell Position Vector (CCPV) to compactly express a multidimensional spatial cell's topologic information of a cell complex model in spatial database systems. First of all, a Cell Position Vector (CPV) equivalent to position vector is made. Then, a Compressed Cell Position Vector (CCPV) can be compressed from CPV by a novel compress algorithm. Because signs in CPV are simple, we developed a novel modified run length algorithm with considerable data compression ratio. Though, VRML model handled well on the network could only be used in not more than three-dimensional space. The comparison of the data sizes with the CCPV and VRML has been implement by experiment. As a result, if a three-dimensional object with 1,712 faces is stored as CCPV, it will be more less than 50% of the data size in VRML model.

1. はじめに

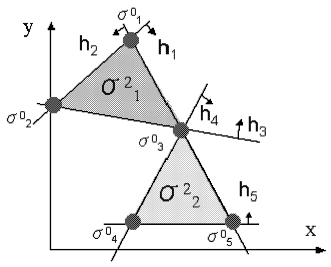
空間オブジェクトの空間データモデルおよびデータベースへの格納法は，空間データベース分野における重要な研究課題である[1),2)]．空間オブジェクトの空間データモデルとして，Indexed Face Set（VRML, X3D などの基礎となるモデル）のサーフェスモデルや，cell complex，linear constraints [17] などが提案されている現状である．この中で cell complex は任意次元空間の任意次元の空間物を一般的に表現することができるという良さがあり，我々は，空間データモデル cell complex の格納と操作の機能を持った空間データベースシステム Hawk's Eye の研究開発を行った．

Cell complex は有界 cell の集合である[12)]．Cell とは空間図形の点，線分，ポリゴンなどの凸な空間物を次元について，一般化した概念である．Cell という言葉は，ユークリッド空間中の凸な空間物という意味で使われる場合と，空間の種類や線形性を仮定しないより抽象的な意味で使われる場合があるが，本論文では前者の意味である．Cell complex の要素は cell であり，cell complex の cell 間の位相関係は，接続グラ

† 九州大学大学院システム情報科学研究院
Graduate School of Information Science and Electrical Engineering, Kyushu University

フ（incidence graph）として表現されるのが一般的である[10),11)]．Cell complex の接続グラフは，cell を節（node）として表現し，次元が 1 つ異なる cell 間に接続関係があるとき，節間の枝（arc）として表現する．枝をポインタを使って表現した接続グラフは，各種空間演算の高速処理が容易であり，実メモリ上の表現形式としては適する[9)]．しかしながら，ポインタ表現の接続グラフを単純にディスクに格納するのは，不可能ではないが，データサイズが大きすぎて得策ではない．この問題を解決するため，本論文では cell の位相関係の表現のために新しく CPV（Cell Position Vector）を導入する．CPV は，点の超平面に対する位相情報を表現する符号ベクトル（position vector）[18)] を基礎として我々が独自に定義を行ったものである．CPV は，cell complex の個々の cell について，cell complex に関係する超平面集合に対する位相関係を表現したベクトルであり，CPV から cell 間の位相関係は容易に分かる（CPV については，3 章でより詳しく説明する）．当然ながら，一般的な cell complex では，関係する超平面の数が多いので CPV の長さは長い．たとえば，我々が，本論文の実験で用いた 3 次元の心臓のデータ（図 9 (b)）は，1,712 個の 2 次元の cell を含む cell complex であるが，この cell complex は，関係する超平面の数が 1,978 であり，cell の CPV の長さは 1,978 になる．そのため，我々は，CPV を，ディスクに格納する際のデータサイズを小さくするため，CCPV（Compressed Cell Position Vector）を提案する．3 章で説明するように，CPV では，cell に無関係な超平面に対応する要素値は **i** であり，CPV では，**i** の割合が高いという性質がある．そこで，既存のランレングスアルゴリズム[19)] を改良し，単純にランレングス法を用いた場合よりも高い圧縮率が得られるモディファイドランレングス（modified run length）アルゴリズムを考案した．このアルゴリズムを使って，CPV の圧縮を行い，CCPV を得る．これで，3 次元以上の cell complex を，データベースに格納する際のデータサイズを小さくすることができる．この CCPV を使った空間データベースシステム Hawk's Eye を実装し，CCPV の有用性を示す実験結果を得た．

本論文の構成は以下のとおりである．2 章では，関連研究の説明を行う．3 章では，空間データモデル cell complex の説明を行う．4 章では，CCPV を説明し，その圧縮と格納アルゴリズムを示して，CPV と CCPV の相互変換法の詳細を説明するとともに，CCPV のデータベースへの格納方法も説明する．5 章では，提案する格納法の評価と実験を行い，6 章で結論を述べる．



$$(x - y + 6 \geq 0 \wedge 2x + y \geq 15 \wedge x/5 + y \geq 6) \vee (2x + y \geq 15 \wedge 2x - y \geq 5 \wedge y \geq 1) \quad (1)$$

図 1 2 つのポリゴンからなる 2 次元空間物を表現する線形制約式の選言の例

Fig. 1 Example of linear constraint formula of a two-dimensional object that consists of two polygons.

2. 関 連 研 究

2.1 制約タプル

従来，線形制約モデル（Linear Constraint Model）を，空間データベースでの空間物の表現に使う試みがあった．これは，空間物を，線形制約項の連言である線形制約式の選言により表現する．たとえば，**図 1** の空間物は，3 つの線形制約項から構成される線形制約式 2 つの選言として表現される（線形制約式の項は，制約等式か制約不等式かのいずれかである）．こうした線形制約モデルを使っての空間データベースシステムの実現の報告としては，DEDALE[13)]，CCUBE[15)]，MLPQ/PReSTO[16)] がある．

DEDALE[13),14)] は，文献 17) に示された線形制約モデルをベースとして，DNF（Disjunctive Normal Form）で空間物を表現する．論文 13), 14) では，SQL 言語で記述された空間問合せ処理方式の研究が行われており，データサイズについては言及されていなかった．DEDALE での DNF を使い cell complex を表現するとき，cell complex の各 top cell を 1 つの線形制約式として表現することになり，cell 間の位相関係は失われる．つまり接続グラフの表現には使えない．なお，DEDALE システムでは，2 次元の空間物までの空間演算しか実装されていない．

2.2 Cell Complex

接続グラフ（incidence graph）[18)] についてのデータ格納法は，従来から，研究者の注目を集めてきた．Simplified Incidence Graph 法[18)] は，d 次元 cell complex に対して，その cell complex を構成するすべての cell についての boundary 関係 $R_{i,i-1}$ $(i = 1, \cdots, d)$ と co-boundary 関係 $R_{i,i+1}$ $(i = 0, \cdots, d-1)$ を格

Collection of 12 tuples forming the cell-tuple data structure for the cell complex shown in the right figure:
(1,a,B), (1,b,B), (2,b,B), (2,c,B), (3,c,B), (3,a,B), (3,d,A), (3,f,A), (4,d,A), (4,e,A), (5,e,A), (5,f,A)

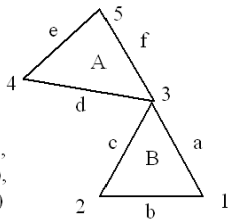


図 2 cell-tuple データ構造に関する例
Fig. 2 Example of the cell-tuple data structure.

納するデータ構造である．これで任意次元の接続グラフを格納できる．この手法では，4 面体（つまり 3-simplex）から構成された 3 次元 cell complex を格納するためには，$786n_v$ *integers*$+3n_v$ *doubles* のデータサイズが必要[6] である（n_t は 4 面体数，n_v は頂点数である．ここで $n_t \approx 6n_v$ を仮定する．このことについて詳しくは文献 8) に説明がある）．一方，我々が提案する CCPV のデータサイズは，4 面体から構成された 3 次元 cell complex の場合，top-cell（cell complex で，包含関係において極大な cell を，本論文では top-cell と表記する．詳しくは 3 章で説明する）のデータサイズは $7n_t$ *integers* となる（参照：図 5）．一方で，0 次元 cell の CCPV の長さは 3 integers（3 つの超平面番号の列），座標のデータサイズは全部で $3n_v$ *doubles* なので，0 次元 cell の CCPV データサイズは $3n_v$ *integers*$+3n_v$ *doubles* である．合計データサイズは $45n_v$ *integers*$+3n_v$ *doubles* と見積もられる．ここで $n_t \approx 6n_v$ を仮定している[8]．なお，0 次元 cell と top-cell の CPV から元の cell complex の他の cell の CPV を求めることは，cell 分割アルゴリズムを用いて容易に行える[21]．

Cell-tuple 構造は，Brisson が提案した任意次元空間の任意次元 cell complex の接続グラフの格納に使えるデータ構造である[3]．それは cell complex のすべての 0 次元 cell と top-cell 間のパス（path）をタプルとして格納する方法である．**図 2** の空間物を cell-tuple で表現すると，(1, a, B)，(1, b, B)，(2, b, B)，(2, c, B)，(3, c, B)，(3, a, B)，(3, d, A)，(3, f, A)，(4, d, A)，(4, e, A)，(5, e, A)，(5, f, A) という 12 個のタプルが得られる．こうしたタプルは，容易に，リレーショナル・データベースシステムなどに格納できる．文献 4), 5) では，cell tuple 構造を使った空間データベースおよび時空間データベースの実現が報告されている．3 次元以上の空間物では，cell tuple 構造のデータサイズが大きくなることは明らかである．d 次元空間中の d-simplex（これは，最も単純な cell complex）を cell tuple で表現すると，$(d+1)!\,(d+1)$ 個のタプルが必要である[7]．たとえば，3 次元空間中の 3-simplex である 4 面体については，cell tuple で規定されている 3 種のリレーション $R_{0,1}$，$R_{1,2}$，$R_{2,3}$ を満足する cell の組の個数は，それぞれ 2，3，4 であり，cell tuple のタプルの総数は $4\times3\times2\times n_t$ である．1 つのタプルの格納のために $d+1$ 個（$d=3$ なら 4 個）の *integer* が必要である．以上から，3 次元空間中の 4 面体については，データサイズは $576n_v$ *integers*$+3n_v$ *doubles* と見積もられる（ここでも $n_t \approx 6n_v$ を仮定している[8]）．

今回提案する CCPV による格納法は，cell-tuple 構造と同様に，任意次元 cell complex の格納に使える．以上の見積り式に示したように，CCPV でのデータサイズは，cell-tuple 構造よりも小さい．

2.3 データ圧縮

データベース分野では可逆圧縮であるテキスト圧縮アルゴリズム[25),26)] が知られている．しかし，既存のテキスト圧縮アルゴリズム LZ77[24)] などを使って，同一文字が数多く連続する傾向にあり，かつ圧縮対象系列の長さが比較的短いビット列のデータの圧縮を行い，データベースに格納する方法は，性能が必ずしも良くない[22)] とされている．したがって，CPV の圧縮には適さないと判断される．我々が提案する CCPV による格納では，圧縮対象は CPV である．CPV は後述するように +，−，0，**i** の 4 種類の符号の列である．それがビット列であるという意味でビットマップインデックス（Bitmap Indices）と類似する．ビットマップインデックスの圧縮アルゴリズムとしてはビットマップインデックス圧縮アルゴリズム Byte-aligned Bitmap Code（BBC）[27)~29)] と Word-Aligned Hybrid（WAH）[23),30),31)] がある．それらは，いずれもランレングス（Run-Length）アルゴリズムを基礎にしている．WAH と BBC の圧縮率を比べると WAH の方が良いとされているうえに，各種処理の性能も良い[23)]．

Byte-aligned Bitmap Code（BBC）は**表 1** に示されるように繰返し bit（0 か 1）の数を保存する圧縮法である[23)]．

一方，Word-Aligned Hybrid（WAH）は，BBC のように繰返し bit（0 か 1）の数を格納するのではなく，00000000 あるいは 7FFFFFFF という 2 種類の 31 ビットデータの繰返し回数を格納するという方法である．WAH は元のデータを，literal word と fill word の組合せとして，word 単位で格納する．Literal word，fill word は，最初の 1 ビットに literal word（0）か fill word（1）であるかを示したフラグを持つ．Fill word では，最後の 1 ビットは，元の 31 ビットデー

表 1 ランレングス圧縮の例
Table 1 Run Length Encoding.

Uncompressed:	0000000000001111000 ··· 00100000000111111110000 ··· 000
Compressed (store counts):	12, 4, 1000,1,8,1000

表 2 WAH の例
Table 2 A WAH example.

元のデータ（128 bits）	1*1, 20*0, 3*1, 79*0, 25*1			
31-bit groups	1*1, 20*0, 3*1, 7*0	62*0	10*0, 21*1	4*1
groups in hex	40000380	00000000 00000000	001FFFFF	0000000F
WAH（hex）	40000380	80000002	001FFFFF	0000000F

タが 00000000 か 7FFFFFFF かの繰返しであることを示すフラグであるそれぞれ 0 か 1 を格納し，その間の 30 ビットには，00000000 あるいは 7FFFFFFF の繰返し回数を格納する．Literal word では残りの 31 ビットに，元のデータを格納している（元のデータが 00000000，7FFFFFFF でない場合）．**表 2** に 128 ビットの長さのデータでの WAH の結果を例として示している．Word 長は 32 ビットなので，元のデータは 31 ビット単位に区切られ，個々の 31 ビットデータが 00000000 あるいは 7FFFFFFF のときは fill word として，それ以外のときは literal word として格納される．表 2 の 2 行目にはデータを 31 ビットに分割する方法を示している．3 行目には 16 進数の表示式が示され，最後の行に実際の WAH データを示す[31)]．

以上のように，BBC および WAH アルゴリズムは値が 0，1 のビットデータ用の圧縮法である．cell の符号ベクトルは，+，−，0，**i** の 4 つの値のベクトルであり，BBC，WAH はそのままで使えず，単純に使っても必ずしも圧縮率は良くない．我々はランレングス（Run-Length）アルゴリズムを基礎とした CPV 圧縮アルゴリズムとしてモディファイドランレングス（Modified Run Length）アルゴリズムを考案した．単純なランレングス（Run-Length）アルゴリズムを使うよりも，本論文で示すアルゴリズムの方が CPV の圧縮率が高い．

3. Cell Complex

Cell complex は任意次元の空間物の幾何構造と位相構造を一様に表現できるという良さがあり，我々が研究開発を行った空間データベースシステム Hawk's Eye では，空間データモデルとして cell complex を実装した．本章では cell complex に関する基本的な概念の説明を行う．

3.1 Cell と Cell Complex

本論文では，d 次元ユークリッド空間を E^d で表記し，空間物はこの中に存在するものとする．cell を σ，cell complex を $\mathbf{\Gamma}$，E^d 中の任意の点を $p = (x_1, x_2, \cdots, x_d)$ と表記する．E^d 中の $(d-1)$ 次元線形部分空間である超平面 h_j（j は超平面番号）は，$d+1$ 個の係数 a_{jk} $(1 \leq k \leq d+1)$ により線形式（式 (2)）を定めたとき，$f_j = 0$ を満足する点の集合である．θ_j を比較演算子 $\leq$ あるいは $\geq$ あるいは $=$ とするとき，「$f_j\ \theta_j\ 0$」は，線形制約項である．

Cell σ は線形制約式を満足する点の集合として定義される．式 (3) のように，cell を定義する線形制約式は，線形制約項（項数を n とする）の連言である．このとき $f_j = 0$ $(1 \leq j \leq n)$ は cell を定義する超平面である．

$$f_j = \sum_{k=1}^{d} a_{jk}x_k + a_{j(d+1)}$$
ただし，a_{jk} $(1 \leq k \leq d)$ のいずれかは 0 ではない． (2)

$$\Phi = \phi_1 \wedge \phi_2 \wedge \ldots \wedge \phi_n$$
$$\theta_j \in \{\leq, \geq, =\},\ p = (x_1, x_2, \cdots, x_d),\ \phi_j : f_j\, \theta_j\, 0 \quad (3)$$

Cell complex $\mathbf{\Gamma}$ は有界 cell の集合である．Cell complex $\mathbf{\Gamma}$ の cell σ が，他の cell のフェイス（face）にならないとき，σ を cell complex $\mathbf{\Gamma}$ の top-cell という．図 1 では 2 次元 cell σ_1^2，σ_2^2 が cell complex $\mathbf{\Gamma}$ の top-cell である．以下 k 次元の cell を k-cell（$0 \leq k \leq d$，d は空間次元）と書く．Cell complex の頂点は 0 次元の cell である．

図 1 は，2 つの 3 角形が貼り合わさった 1 つの図形であり，1 つの cell complex をなす．これは，2 次元の cell complex であり，2 個の 2 次元 cell（top-cell）σ_1^2，σ_2^2，5 個の 0 次元 cell σ_1^0，σ_2^0，σ_3^0，σ_4^0，σ_5^0 と，これら cell とつながる 6 個の 1 次元 cell という 0 から 2 次元の cell を要素として持つ．

3.2 超平面と点の符号ベクトル

E^d 内の n 個の超平面の集合 $H = \{h_1, h_2, \cdots, h_n\}$

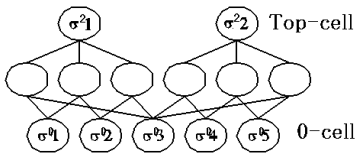


図 3 図 1 の cell complex の接続グラフ
Fig. 3 The incidence graph of Fig. 1.

(h_j は $f_j = 0$ を満足する点の集合) は 0 から d 次元の cell に E^d を分割する．E^d 内の任意の点 $p(x_1, x_2, \cdots, x_d)$ に対し，式 (4) を使ってその符号ベクトル $V(p) = [v_1(p)\ v_2(p)\ \cdots\ v_n(p)]$ が定義され，点 p の符号ベクトル (position vector)[18] と呼ばれる．

$$v_j(p) = \begin{cases} + & f_j > 0 \text{ のとき} \\ 0 & f_j = 0 \text{ のとき} \\ - & f_j < 0 \text{ のとき} \end{cases} \quad (4)$$

3.3 接続グラフ

接続グラフは，cell complex の cell 間の接続関係を表現したグラフであり，node は cell を，arc は cell 間の接続関係を表す．図 1 に書いた cell complex の cell の関係は，**図 3** に示したような接続グラフで表現できる．

3.4 cell の符号ベクトル

Cell は点の集合であり，cell 内の任意の点は cell complex に関係する超平面集合に対しての符号ベクトルを持つ．cell complex が要素とする全 cell についての，cell を定義する超平面集合の集合和を，cell complex に関係する超平面集合と定める．

Cell complex の要素である cell について，cell の内部の任意の点の，cell complex に関係する超平面集合に対する符号ベクトルを作ると，その符号ベクトル中の +，− のうちには，cell の表現にとって無関係な +，− がある．たとえば，図 1 の cell σ_1^2 の表現には，σ_1^2 を取り囲んでいる h_1，h_2，h_3 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，σ_1^2 を取り囲んでいない超平面 h_4，h_5 に対する分は冗長である．

一方，0 次元 cell の符号ベクトルでは符号ベクトル値 **0** だけが必要である．たとえば，図 1 の σ_1^0 では，h_1 と h_2 だけがこの 0 次元 cell の表現に必要であり，他の超平面は不要である．以上のアイデアから，CPV (Cell Position Vector) の定義を以下のように定める．CPV は，cell complex の 0 次元 cell と top-cell についてその cell complex に関係する超平面集合との位相関係を表現したベクトルである．その長さは，cell complex に関係する超平面集合の超平面の個数に等しい．たとえば，図 1 の cell complex の 0 次元 cell の CPV は，**表 3** のようになる．

表 3 図 1 の cell complex の 0 次元 cell の CPV
Table 3 Cell position vectors of 0-cells of the cell complex shown in Fig. 1.

0-cell	点の position vector	CPV
σ_1^0	[0 0 + − +]	[0 0 i i i]
σ_2^0	[0 + 0 − +]	[0 i 0 i i]
σ_3^0	[+ 0 0 0 +]	[i 0 0 0 i]
σ_4^0	[+ + − 0 0]	[i i i 0 0]
σ_5^0	[+ 0 − + 0]	[i 0 i i 0]

表 4 図 1 の cell complex の top-cell の CPV
Table 4 Cell position vectors of top-cells in Fig. 1.

top-cell	点の position vector	CPV
σ_1^2	[+ + + − +]	[+ + + i i]
σ_2^2	[+ + − + +]	[i + i + +]

0 次元 cell σ^0 の CPV 値は，以下のように定める．0 次元の cell は要素数が 1 の点の集合である (この点をここでは説明のため p_{σ^0} と書く)．j 番目の超平面 $f_j = 0$ に対し，$v_j(p_{\sigma^0}) = 0$ ならば (つまり，h_j がこの 0 次元 cell を含むならば) 0 次元 cell の CPV の j 番目の値 s_j を 0 とし，さもなければ CPV の j 番目の値 s_j を **i** (irrelevant) とする．つまり，0 次元 cell の CPV は，0，**i** の 2 値のみからなるベクトルである．表 3 は図 1 の cell complex の 5 個の 0-cell について，その点としての符号ベクトルと CPV をそれぞれ示している．

Top-cell の CPV は，+，−，0，**i** の 4 つの値からなるベクトルである．k 次元 top-cell σ^k $(1 \le k \le d)$ の CPV 値は，以下のように定める．k 次元 top-cell σ^k のフェイスである $k-1$ 次元 cell を含む超平面 $f_j = 0$ に対し，σ^k の内部の任意の点 p が $f_j > 0$ を満たすときは，CPV の j 番目の値 s_j を + と定める．任意の点 p が $f_j < 0$ を満たすときは，CPV の j 番目の値 s_j を − と定める．任意の点 p が $f_j = 0$ を満たすとき，CPV の j 番目の値 s_j は，0 と定める．以上のいずれも成り立たないときは，h_j は σ^k の内部で交わっていて，σ^k を定義する線形制約式とは関係ないので，s_j は，**i** とする．たとえば，図 1 の cell complex の 2 次元 top-cell の CPV は，**表 4** のようになる．

以上で定義したように，top-cell の CPV は，top-cell の内部の任意の点の符号ベクトルの +，− のうち超平面が top-cell の定義に必要ないものを **i** で置き換えたものである．

以上で説明したように，CPV は cell と超平面との位相関係を示す．**表 5** の (a)，(b)，(c) ではそれぞれ

表 5 図 1 の cell complex のメモリ上での表現
Table 5 The data representation in memory for the cell complex shown in Fig. 1.

(a)Vertex

0-cell	座標	cell position vector の 0 の位置
σ_1^0	(3, 9)	1, 2
σ_2^0	(0, 6)	1, 3
σ_3^0	(5, 5)	2, 3, 4
σ_4^0	(3, 1)	4, 5
σ_5^0	(7, 1)	2, 5

(b) Top cell

top-cell	position vector
σ_1^2	$[+ + + \ i \ i]$
σ_2^2	$[i + i + +]$

(c) Hyperplane

	超平面方程式	交差する 0 次元 cell
h_1	y = x + 6	1, 2
h_2	y = −2x + 15	1, 3, 5
h_3	y = −1/5x + 6	2, 3
h_4	y = 2x − 5	3, 4
h_5	y = 1	4, 5

Hawk's Eye での 0 次元 cell，top cell と超平面の実メモリ上での表現を示す．

4. Cell Complex の圧縮と格納

我々は，空間データベースに格納する際のデータサイズを小さくするために，Compressed Cell Position Vector（CCPV）を考案した．本章では CPV の圧縮・復元アルゴリズムを説明する．同時に点の符号ベクトルと CPV の相互変換法も説明する．

4.1 点の符号ベクトルから CPV への変換

3 章で説明したように点の符号ベクトル中の +，− のうちには，cell の表現にとって無関係な +，− がある．点の符号ベクトルから CPV への変換では，こうした無関係な +，− を **i** に変換する．点の符号ベクトルを CPV に変換する処理の順序としては，まず，0 次元 cell σ^0 の点 p_{σ^0} の符号ベクトル $V(p_{\sigma^0})$ の +，− 値をすべて **i** に変換し，σ^0 の CPV を得る．一方，top-cell の内部の点の符号ベクトルから CPV へ変換する方法はアルゴリズム 4.1 で表している．Top-cell については，図 3 のような cell complex の接続グラフより得られる接続関係と，接続している 0 次元 cell の CPV の値を使い，top-cell の符号ベクトルの値が +，− のうち，**i** への変更が必要なものを探して **i** へ変更して，CPV を得る．

Algorithm 4.1: MKTOPCPV(P, A, S)

Input:
$V = [v_1 \cdots v_k]$, position vector of a top-cell
A, Set of 0-cell vectors connecting with cell P
Output:
$S = [s_1 \cdots s_k]$, CPV of a top-cell
for $j \leftarrow 1$ **to** k

$$\textbf{do} \begin{cases} sum_j \leftarrow \text{count of 0 on the j-th sign in A} \\ \textbf{if } sum_j < dim \qquad (1) \\ \quad \textbf{then } s_j \leftarrow \textbf{i} \\ \quad \textbf{else } s_j = p_j \end{cases}$$

rerutn (S)
Condition:
the number of 0 *on the j-th position of all the* 0*-cell CPV in A is less than the maximum dimension of the spatial data.*

たとえば，図 1 では，top-cell σ_1^2 が要素として含んでいる 0 次元 cell としては σ_1^0 と σ_2^0 と σ_3^0 がある．その中の対応する超平面 h_1，h_2，h_3 については，それぞれ 2 つ以上の 0 次元 cell が交差している．この top-cell σ_1^2 の内部の点の符号ベクトル $[+ + + - +]$ から CPV への変換では，超平面 h_1，h_2，h_3 に対応する点の符号ベクトル $[+ + +]$ は変えないで，そのまま CPV 値とする．他の超平面に対応する点の符号ベクトルは，すべて，**i** に変更する．こうして top-cell σ_1^2 の CPV $[+ + + \ \textbf{i} \ \textbf{i}]$ が得られる．

4.2 圧縮（CPV から CCPV へ）

Top-cell の CPV の中には **i** 値が多く含まれるという性質を使って，CPV を圧縮できる．本節では CPV から Compressed Cell Position Vector（CCPV）に圧縮するアルゴリズムを述べる．

Top-cell については，1 つの CPV につき，アルゴリズム 4.2 に示したモディファイドランレングスアルゴリズムを用いて，CPV から CCPV へ圧縮する．

Top-cell の CPV に使うモディファイドランレングスアルゴリズムは以下のとおりである．Top-cell の CPV の値は，+，−，0，**i** の 4 通りであるが，大部分が **i** であるので，アルゴリズム 4.2 では CPV 値 $s_j \in \{+, -, 0, \textbf{i}\}$ について s_j の値が + または − または 0 である番号 j しか覚えないという方針で圧縮する．まず，CPV の中での +，−，0 の登場回数をそれぞれ +，−，0 の順でヘッダとして格納し，その後，本体に +，−，0 の CPV 値を持つ超平面番号（つまり CPV

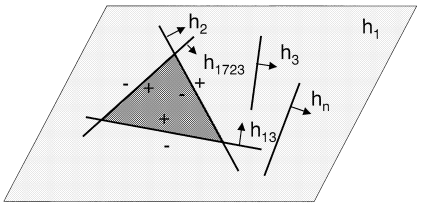


図 4 3 次元空間の 2 次元 cell は h_1，h_2，h_{13}，h_{1723} の 4 超平面で定義されている
Fig. 4 A 2D top-cell defined by h_1, h_2, h_{13} and h_{1723} in 3D space.

の要素番号）を +，−，0 の順で，すべて格納する．

Algorithm 4.2: ENCODETOPCPV(S, ES)

Input:
$S = [s_1 \cdots s_k]$, CPV of a top-cell
Output:
$ES = [es_1 \cdots es_m]$, CCPV of a top-cell
len, Length of ES
for $j \leftarrow 1$ **to** k
do
- **if** $s[j] = +$ **then** $arr\_p[k_1] \leftarrow i; k_1 \leftarrow k_1 + 1$
- **if** $s[j] = -$ **then** $arr\_m[k_2] \leftarrow i; k_2 \leftarrow k_2 + 1$
- **if** $s[j] = \mathbf{0}$ **then** $arr\_0[k_0] \leftarrow i; k_0 \leftarrow k_0 + 1$

$m \leftarrow k_0 + k_1 + k_2 + 2$
$ES[0...2] \leftarrow (k_1, k_2, k_0)$
$ES[3...m] \leftarrow (arr\_p[\,], arr\_m[\,], arr\_0[\,])$
return $(m + 1)$

実験で用いた空間物データ（図 9 (b) に形状を示している）の場合，2 次元 top-cell σ_1^2（**図 4** に示した 3 角形メッシュ中の 1 つの 2 次元 cell）の CPV は，長さが 1,978 のベクトル $[0 - i \cdots \mathrm{i} + \mathrm{i}\,\mathrm{i} \cdots \mathrm{i} + \mathrm{i}\,\mathrm{i} \cdots \mathrm{i}]$ である（**図 5**）．それをアルゴリズム 4.2 で圧縮すると，[2 1 1 13 1,723 2 1] のような CCPV が求まる．

Top-cell の CPV の値は大部分が **i** であるので提案したモディファイドランレングスアルゴリズムは従来のランレングスアルゴリズムより高い圧縮率で圧縮できる．図 4 に示した top cell の CPV の圧縮結果を比較すると**表 6** のようになり CCPV の方が短い．

Algorithm 4.3: ENCODE0CPV(S, ES)

Input:

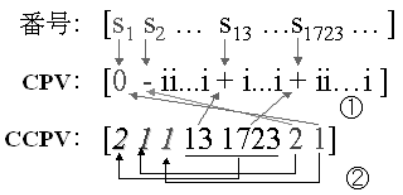


図 5 CPV 圧縮の例
Fig. 5 A compression example of a CPV.

表 6 モディファイドランレングスアルゴリズムとランレングスアルゴリズムの比較
Table 6 A comparison between the modified run length algorithm and the run length algorithm.

アルゴリズム	圧縮の結果
run length	1 0 1 − 10 i 1 + 1,709 i 1 + 254 i
modified run length	2 1 1 13 1,723 2 1

$S = [s_1 \cdots s_k]$,CPV of a 0-cell
Output:
$ES = [es_1 \cdots es_m]$, CCPV of a 0-cell
len, Length of ES
$n \leftarrow 1$
for $j \leftarrow 1$ **to** k
do **if** $s[j] = 0$ **then** { $es[n] \leftarrow i$; $n \leftarrow n + 1$ }
return (n)

0 次元 cell については，CPV 値 0 に対応する超平面番号の列に変換する．つまり，CPV の値 $s_j = 0$ に対応する超平面番号 j の列を得る．こうした番号の列が 0 次元 cell の CCPV である．もし 0 次元 cell σ^0 の CPV の j 番目の値 s_j が 0 であれば，超平面番号の列である CCPV に j を加える．アルゴリズム 4.3 は 0 次元 cell の CPV から CCPV へ圧縮するアルゴリズムである．

4.3 CPV からの top-cell と 0-cell の接続関係抽出

Cell complex を構成する top-cell について，その CPV から，top-cell を定義する超平面集合に対してのみ +，−，または 0 値を持つ符号ベクトルへの変換は，アルゴリズム 4.4 の手順で効率良く行える．CPV から $s_j = 0$，+，− である超平面番号を集めて，結果を $H = \{h_1, h_2, \cdots, h_m\}$ とする（アルゴリズム 4.4）．このアルゴリズムの出力は，top-cell に接続している超平面の番号の列とそれに対応する符号ベクトルである（**図 6**）．

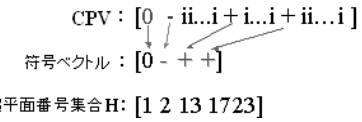


図 6 CPV から点の符号ベクトルへ変換の例
Fig. 6 Example of transformation from a CPV to a position vector of a top-cell.

Algorithm 4.4: CPVTOPVS(S, len, H, V)

Input:
$S = [s_1 \cdots s_n]$, a top-cell CPV
n, Length of S
Output:
$V = [v_1 \cdots v_m]$
$H = [h_1 \cdots h_m]$
$k \leftarrow 1$
for $j \leftarrow 1$ **to** n
do { **if** $s_j \neq$ **i**
then { $v_k \leftarrow s_j$; $h_k \leftarrow j$; $k \leftarrow k + 1$ }
return (k)

0 次元 cell σ^0 の CPV からその 0 次元 cell の点 p_{σ^0} の符号ベクトルへの変換は次の手順で行う．まず，アルゴリズム 4.4 の結果を使い，CPV($[s_{h_1} \cdots s_{h_m}]$) 中に d 個以上（d は空間次元）の 0 値を持つ 0 次元 cell のみを選び出し，その集合を **S** とする．次に，**S** に含まれる 0 次元 cell について，それぞれの座標値 p を得る．座標値の計算は，CPV($[s_{h_1} \cdots s_{h_m}]$) から，$s_{h_j} = 0$ を満たす線形独立な超平面 h_j を任意に d 個選び，それらの交点を計算するなどの手続きで容易に可能である．そして，$s_{h_j} =$ **i** であるような超平面 h_j に対して，f_j の値を計算する．この結果，+，−，0 のベクトルが得られる．s_{h_j} が + (−) であり，かつアルゴリズム 4.4 で求めた top-cell の符号ベクトル ($[v_1 \cdots v_m]$) の v_{h_j} が + (−) でなければ，0 次元 cell は top-cell に接していない．$s_{h_j} =$ **i** を満たすすべての超平面に対して上記の手順を行うことで，ある top-cell に接している 0 次元 cell を cell complex 内から選ぶことができ，top-cell と 0 次元 cell の接続関係が求まる．

4.4 実　　装

我々の空間データベースシステム Hawk's Eye をオブジェクトデータベース出世魚[32] 上に構築し，CCPV 表現された cell complex を格納するよう実装を行った．Cell complex を格納するため，cell complex のすべての 0 次元 cell の座標値を d_Double でデータベースに格納し，top-cell と 0 次元 cell の CCPV を d_String で保存する（d_Double と d_String は ODMG-2.0 のデータタイプ）．0 次元 cell のクラスは該当する 0 次元 cell の座標と CCPV をメンバ変数とする．Top-cell クラスは CCPV をメンバ変数とする．

表 7 2 次元のランダムデータでの実験結果 (KB)
Table 7 Result data of 2D random objects (KB).

超平面数	40	50	60	70	80	90	100
圧縮前	37	74	139	235	350	510	751
圧縮後	16	27	44	65	86	113	146

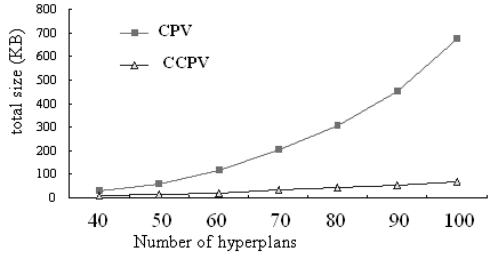


図 7 2 次元のランダムデータでの実験結果
Fig. 7 Result data of 2D random objects.

5. 実　　験

空間データベースシステム Hawk's Eye はオペレーティングシステム Solaris 9 に実装した．本研究室において開発した ODMG2.0[20] 標準でのオブジェクトデータベースシステム出世魚[32] を用いて実験を行った．

5.1 CPV の圧縮前と圧縮後（CCPV）のデータサイズ比較

- 2 次元空間中にランダムに n 個の超平面を作成し，この超平面の集合を **H** とする（n は 40，50，60，70，80，90，100 の 7 通り）．
- 分割してできた cell の中から 0 次元 cell と 2 次元の有界 cell をすべて選ぶ．

上記の手順で作成された実験データの中の 0 次元 cell と 2 次元の有界 cell の CPV を使い，4 章のアルゴリズムで圧縮し，CCPV を得る．CCPV と超平面方程式の合計サイズについて圧縮前と圧縮後の比較を行った．比較結果を**表 7** と**図 7** に示す．

図 7 に示したように，予想どおり，超平面の数が多ければ圧縮率が良くなることが分かった．

5.2 データサイズの計算式

次に，3 次元空間の球面を実験データとして，VRML，DNF でデータベースに格納する場合と，本

表 8 3 次元球面データサイズの比較
Table 8 Comparison result by using data of a 3D sphere.

数		データサイズ (Byte)		
点の数	凸 x 面体	CCPV	DEDALE	VRML
6	8	392	385	240
18	32	1,424	1,540	816
66	128	5,552	6,160	3,120

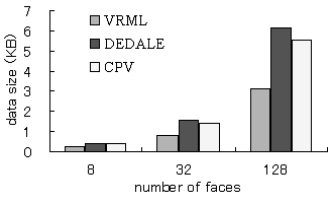


図 8 3 次元球面データサイズの比較
Fig. 8 Comparison result by using data of a 3D sphere.

論文で示した CCPV による cell complex 表現法でデータベースに格納する場合のデータサイズの比較を行った．

本実験では，座標や超平面の値を格納するための d_Double を 8 バイト，DEDALE の DNF 式に使う符号 $(\wedge, \vee)$ を 1 ビットとした．式 (5)，(6)，(7) には，実験データを VRML，DEDALE と CCPV で表現する場合のデータサイズの見積り式を示している．比較のために N_0，N_t，α，γ などのパラメータ値については，球面に内接する凸 8 面体，凸 32 面体，凸 128 面体を実験的に作り，必要なパラメータ値を実測で得た．こうして得られた値を式 (5)，(6)，(7) に代入した結果が**表 8** と**図 8** である．

- d：空間次元
- N_0：0 次元 cell 数
- N_t：top-cell 数
- n：超平面数
- γ：top-cell に接している 0 次元 cell の数
- α：top-cell に接している超平面の数
- a：int のデータサイズ
- b：d_Double のデータサイズ

VRML モデルの計算式

$$Datasize = d \times N_0 \times b + \gamma \times N_t \times a \quad (5)$$

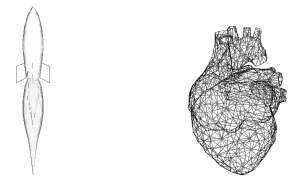
DEDALE の DNF モデルの計算式

$$Datasize = O((d+1) \times n \times b + (\alpha \times a + 1/8) \times N_t) \quad (6)$$

CCPV での cell complex モデルの計算式

$$Datasize = O(d \times N_0 \times b + (\alpha \times a + 3) \times N_t + \gamma \times N_0 \times a) \quad (7)$$

CCPV は，DNF と比べても，それほど大差がないことが分かった．DNF，VRML ともに，空間物の位置と形状の表現に使えるデータ表現法であるが，CCPV では，空間物の位置と形状情報だけでなく，cell complex を構成している cell の位相関係をも表現し，本質的に，含まれている情報量が多い．球面の場合について，CCPV のデータサイズが VRML を下回らないのは，予想できた結果ではあるが，DNF を下回るデータサイズまでに圧縮できることが凸 8 面体，凸 32 面体，凸 128 面体の場合で示せ，CCPV の実用性を示すことができ満足できる結果であった．



(a) rocket (b) heart

図 9 実験で用いた 3 次元 VRML データ
Fig. 9 Test data of 3D objects.

5.3 VRML データでの CCPV の実験

我々は Web [33),34)] からダウンロードした 2 つの 3 次元 VRML データ（**図 9**）を cell complex データに変換してから，4 章のアルゴリズムで CCPV で表現した後，元の VRML ファイルとのデータサイズの比較を行った．実験のために，まず，VRML ファイルの Indexed Face Set 表現から cell complex の CPV 表現へ変換を行う．この変換では，VRML の 2 次元の面を cell complex の top-cell として扱い，VRML データに含まれる頂点データを，0 次元 cell として扱う．この時点で，top-cell と 0 次元 cell の CPV を計算で求めるが，そのために，最初に，top-cell を含む超平面の方程式を求め，超平面集合 **H** を得る．次に，0 次元 cell の座標を，作成した超平面集合 **H** のそれぞれの方程式に代入し，各超平面に対する符号ベクトルの値 v_j を求めて，0 次元 cell の点の符号ベクトルを得る．0 次元 cell の点の符号ベクトルから top-cell の CPV を求める方法は以下のとおりである．ア）Top-cell が 2 次元の cell なので（以下 σ_a^2 で表記する），σ_a^2 に接しているすべての 0 次元 cell について，点の符号ベクトルの j 番目の値 v_j が 0 の場合，σ_a^2 の CPV の値 $s_j = 0$ である．イ）σ_a^2 に接しているすべての 0 次元 cell についての点の符号ベクトルの j 番目の値 v_j に符号 $+$ と $-$ 両方がある場合は σ_a^2 の CPV の j 番目の値 s_j を **i**（irrelevant）にする．ウ）それ以外の場合は，0 次元 cell の j 番目の 0 ではない v_j の値（$+$ か $-$）を σ_a^2 の CPV の j 番目の値 s_j を v_j とす

表 9 VRML テストデータのフェース数，頂点数と超平面数
Table 9 Number of faces, number of vertexs and hyperplane number of VRML test data.

	N_{hp}	N_p	N_{face}
Rocket	1,036	590	1,088
Heart	1,978	964	1,712

表 10 図 9 のテストデータでの VRML と CCPV の比較 (KB)
Table 10 Comparison between VRML and CCPV by test data shown in Fig. 9 (KB).

	VRML ファイル	圧縮前	圧縮後
Rocket	41	449	34
Heart	106	1,348	52

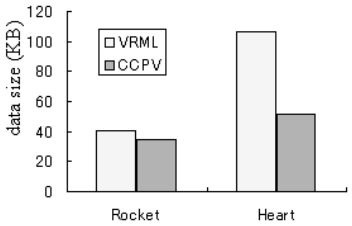


図 10 図 9 のテストデータでの VRML と CCPV の比較
Fig. 10 Comparison between VRML and CCPV by test data shown in Fig. 9.

る．最後に，0 次元 cell についての点の符号ベクトルの +，− 値をすべて **i** に変換し，0 次元 cell の CPV が得られる．このようにして cell complex の top-cell と 0 次元 cell の CPV が作成される．

表 9 において，図 9 に示した cell complex についての N_{hp} は超平面の数，N_p は点の数，N_{face} は 2 次元フェイスの数を示している．実験結果は**表 10** および**図 10** のとおりである．実験より，VRML データのフェイス数が多ければ圧縮効果が良いことが示されている．これは予想どおりの結果である．図 9 (b) の空間物のフェイス数は，図 9 (a) の空間物よりも多いので，圧縮率は，図 9 (a) より図 9 (b) の方が良い．図 9 の空間物の場合，CPV の圧縮前はデータサイズが大きくなるが，CCPV（圧縮後）では元の VRML よりも小さいことを確認できた．今回は，CPV の圧縮の研究であり，0 次元 cell が持つ座標値の圧縮は行わない．座標値の圧縮については，座標値の分布や隣接関係を使って，無損失（lossless）に圧縮する種々の研究が知られており，そうした既存の圧縮法と組み合わせることは，今後の実装上の課題である．

本実験のデータに対して，本論文に提案するモディファイドランレングスアルゴリズムが単純なランレングスアルゴリズムより圧縮率が良いことも実験で確認した．ランレングスアルゴリズム（RLE）で圧縮するとき，ランレングスが有限のビット数に格納されるので（今回の実験では unsigned char を使用する），最大のカウントに達しているとき，このカウント値と +，−，0，**i** の符号の値を格納する（本論文では符号を 2 bit で格納する）．そして，続けて符号のカウント値を格納する[35]．上記のランレングスアルゴリズムとモディファイドランレングスそれぞれでの圧縮結果を，**図 11**，**表 11** と**表 12** に示す．図 11 に示している値は実験から得た top-cell と 0 次元 cell の符号ベクトルを圧縮したデータサイズの合計である．

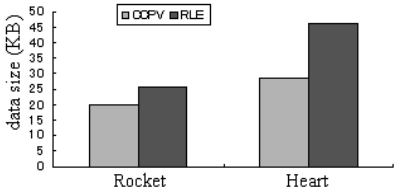


図 11 ランレングスアルゴリズムと CCPV の比較
Fig. 11 Comparison of run length algorithm and CCPV.

表 11 ランレングスアルゴリズム（RLE）と CCPV の圧縮率の比較（rocket）
Table 11 Comparison with run length algorithm (rocket).

	CPV	RLE		CCPV	
	size (KB)	size (KB)	rate (%)	size (KB)	rate (%)
Top-cell	281.8	15.8	6	12.3	4
0-cell	152.8	10.0	7	7.8	5
合計	434.6	25.8	6	20.1	5

表 12 ランレングスアルゴリズム（RLE）と CCPV の圧縮率の比較（heart）
Table 12 Comparison with run length algorithm (heart).

	CPV	RLE		CCPV	
	size (KB)	size (KB)	rate (%)	size (KB)	rate (%)
Top-cell	847.4	30.0	4	17.3	2
0-cell	477.2	16.3	3	11.3	2
合計	1,324.6	46.3	3	28.6	2

本章の実験結果をまとめると，次のとおりである．我々のアルゴリズムで cell complex を格納する際のデータサイズを小さくすることができることを実証した．VRML データで実際に示したように，圧縮率は cell 数の増加とともに高くなる．

6. ま と め

本研究では，Compressed Cell Position Vector（CCPV）による cell complex の格納法を提案し，オブジェクトデータベース管理システム「出世魚」に実装した．CCPV による cell complex の表現は任意次元の空間中の任意次元の cell complex を表現するデー

タサイズを小さくできるという特徴を持つ．実験では，2 次元 cell 数が 1,712 の cell complex（図 9 (b)）を使い，VRML ファイルに比べてデータサイズがおよそ 50%小さいことを示した．本研究の意義は，任意の次元の空間中の 3 次元以上の cell complex を，従来の格納法よりも小さなデータサイズで格納できたことにある．本論文のモディファイドランレングスアルゴリズムは，単純なランレングス法よりも高い圧縮率を達成できる．

謝辞 本研究の一部は，日本学術振興会科学研究費補助金課題番号 15650017，若手研究（B）17700117 による．

参 考 文 献

1) Güting, R.H.: An Introduction to Spatial Database Systems, *The Very Large Data Base J.*, Vol.3, No.4, pp.357–399 (1994).
2) Voisard, A. and David, B.: A Database Perspective on Geospatial Data Modeling, *IEEE TKDE*, Vol.14, No.2, pp.226–243 (2002).
3) Brisson, E.: Representing geometric structures in d dimensions: Topology and order, *DiscreteComput. Geom.*, Vol.9, No.4, pp.387–426 (1993).
4) Raza, A. and Kainz, W.: Cell tuple based spatio-temporal data model: An object approach, *Proc. 7th ACM international symposium on Advances in geographic information systems*, pp.20–25 (1999).
5) Worboys, M.F.: A Unified Model for Spatial and Temporal Information, *Computer Journal*, Vol.37, No.1, pp.26–34 (1994).
6) De Floriani, L. and Hui, A.: A scalable data structure for three-dimensional non-manifold objects, *Proc. Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pp.72–82 (2003).
7) De Floriani, L., Greenfieldboyce, D. and Hui, A.: A data structure for non-manifold simplicial d-complexes, *Proc. Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pp.83–92 (2004).
8) Cignoni, P., De Floriani, L., Magillo, P., Puppo, E. and Scopigno, R.: Selective Refinement Queries for Volume Visualization of Unstructured Tetrahedral Meshes, *IEEE Trans. Visualization and Computer Graphics*, Vol.10, No.1, pp.29–45 (2004).
9) de Berg, M., van Kreveld, M., Overmars, M. and Schwarzkopf, O.: Computational Geometry: Algorithms and Applications, 2nd edition, Springer-Verlag, Berlin (2000).
10) Skiena, S.: *Hasse Diagrams, in Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*, Reading, MA: Addison-Wesley, p.163, pp.169–170 and 206–208 (1990).
11) Trotter, W.T.: *Combinatorics and Partially Orderd Sets: Dimension Theory*, Johns Hopkins Series in the Mathematical Sciences, The Johns Hoplins University Press (1992).
12) Lundell, A.T. and Weingram, S.: *The Topology of CW Complexes*, Van Nostrand Reinhold Comp. (1969).
13) Grumbach, S., Rigaux, P. and Segoufin, L.: The Dedale System for Complex Spatial Queries, *Proc. 1998 SIGMOD*, pp.213–224 (1998).
14) Rigaux, P., Scholl, M., Segoufin, L. and Grumbach, S.: Building a constraint-based spatial database system: Model, languages and implementation, *Information Systems archive*, Vol.28, pp.563–595 (2003).
15) Brodsky, A., Segal, V., Chen, J. and Exarkhopoulo, P.: The CCUBE constraint object-oriented database system, *Proc. 1999 SIGMOD*, pp.577–579 (1999).
16) Revesz, P.: *Introduction to Constraint Databases*, Springer-Verlag (2002).
17) Kanellakis, P.C., Kuper, G.M. and Revesz, P.Z.: Constraint query languages (preliminary report), *Proc. 9th ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, April 2–4, 1990, Nashville, Tennessee, pp.299–313 (1990).
18) Edelsbrunner, H.: *Algorithms in Combinatorial Geometry*, Springer-Verlag (1987).
19) Salomon, D.: *Data compression: The complete reference*, Springer Verlag (1998).
20) Cattell, R.G.G. and Barry, D.K.: *The Object Data Standard ODMG 2.0*, Morgan Kaufmann Publishers (1997).
21) Tanaka, M., Kaneko, K., Lu, Y. and Makinouchi, A.: An Extended cell Splitting Algorithm for Spatial Databases, *IEEE TENCON 2004*, pp.371–374 (Nov. 2004).
22) Johnson, T.: Performance Measurements of Compressed Bitmap Indices, *VLDB*, pp.278–289 (1999).
23) Pinar, A., Tao, T. and Ferhatosmanoglu, H.: Compressing Bitmap Indices by Data Reorganization, *21st IEEE International Conference on Data Engineering* (*ICDE'05*), pp.310–321 (Apr. 2005).
24) Ziv, J. and Lempel, A.: A universal algorithm for sequential data compression, *IEEE Trans.*

Inf. Theory, Vol.23, Issue 3, pp.337–343 (May 1977).
25) Gailly, J.-L. and Adler, M.: zlib 1.1.3 manual, July 1998, Source code available at http://www.info-zip.org/pub/infozip/zlib
26) Jürgens, M. and Lenz, H.-J.: Tree based indexes vs. bitmap indexes — A performance study, *Design and Management of Data Warehouses* (*DMDW*), Heidelberg (1999).
27) Antoshenkov, G.: Byte-aligned bitmap compression, Technical report, Oracle Corp. (1994). U.S. Patent number 5,363,098.
28) Antoshenkov, G. and Ziauddin, M.: Query processing and optimization in ORACLE RDB, *VLDB Journal*, Vol.5, pp.229–237 (1996).
29) Johnson, T.: Performance Measurements of Compressed Bitmap Indices, *VLDB 1999*, pp.278–289 (1999).
30) Amer-Yahia, S. and Johnson, T.: Optimizing queries on compressed bitmaps, *VLDB*, pp.329–338 (2000).
31) Wu, K., Otoo, E.J. and Shoshani, A.: An Efficient Compression Scheme For Bitmap Indices, Technical Report 49626, LBNL (Apr. 2004).
32) 金子邦彦，牧之内顕文，于　戈，王　国仁，佐藤秀樹，長田　正：ODMG2.0 世界標準に準拠する NOW 上の分散並列 ODB 管理システムの開発—E-Commerce のためのスケーラブル ODMS「出世魚」，先端的情報化推進基盤整備事業論文集 (2000).
33) http://www.3dcafe.com/avatar/rocket.wrl
34) http://www.ocnus.com/models/People/
35) http://michael.dipperstein.com

(平成 17 年 9 月 21 日受付)
(平成 17 年 12 月 28 日採録)

（担当編集委員　佐藤 哲司）

陸　応亮

2005 年九州大学大学院システム情報科学研究院知能システム学部門修士課程修了．現在，同大学院博士後期課程在学中，空間データベースシステム，類似検索の研究に従事．

金子　邦彦（正会員）

1990 年九州大学工学部情報工学科卒業．1995 年同大学大学院博士後期課程修了，同助手を経て，1999 年九州大学大学院システム情報科学研究院助教授．電子情報通信学会，ACM，IEEE 各会員．

田中美智子

2004 年九州大学工学部電気情報工学科卒業．現在，同大学大学院システム情報科学府知能システム学部門修士課程在学中．空間データベースシステムの研究に従事．日本データベース学会学生会員．

牧之内顕文（正会員）

1967 年京都大学工学部電子工学科卒業．1970 年グルノーブル大学理学部応用数学科 Docteur-Ingenieur 取得．（株）富士通，（株）富士通研究所，九州大学工学部教授を経て，1967 年九州大学大学院システム情報科学研究科教授．現在，同大学院システム情報科学研究院教授．電子情報通信学会，ACM，IEEE 等の会員．