Research Paper

Binary spatial operations on cell complex using incidence graph implemented at a spatial database system Hawk Eye

Kunihiko KANEKO[1], Akifumi MAKINOUCHI[1]
[1] *Kyushu University*

ABSTRACT

We implemented a spatial database system called Hawk Eye for three- and four-dimensional modeling applications, such as solid modeling, computer simulation and computer vision. Spatial query and manipulation are important system functions for retrieving and analyzing spatial objects. Binary spatial operations are necessary in order to respond to spatial queries and manipulations. Efficient processing of binary spatial operations between two cell complexes is important with respect to a cell-complex-based spatial database because the evaluation of these operations by previous algorithms is time-consuming. We present a new algorithm called the Cell Splitting and Merge Algorithm (CSMA) to evaluate binary spatial operations between two cell complexes. The new algorithm is efficient for cell complexes of three or four dimensions. Key to the algorithm is the use of an incidence graph of the cell complex.

KEYWORDS

Spatial database, spatial query processing, cell complex, cell splitting, cell splitting and merge algorithm

1 Introduction

Spatial data is required in applications such as Geographic Information Systems (GIS), CAD/CAM, medical information systems, image processing, robotics, computer graphics, computer vision, and computer simulation. In many applications, the spatial dimension is either three or four. To represent spatial objects in three-dimensional or four-dimensional space, several types of spatial data models have been proposed, including surface models, three-dimensional solid models, and cell-based models. Among these models, cell-based models are advantageous in that they can be used to represent spatial objects of various dimensions from 0 to d in d-dimensional space ($d = 2, 3, 4$ or higher) uniformly.

Spatial objects can be classified into two types, manifold objects and non-manifold objects, based on the topological structure. [8] A spatial object is *manifold* if the neighbor of every interior point of the object is locally equivalent to an open ball, otherwise the object is non-manifold. [8] For example, the object in Fig. 1 (a) consists of two rectangles that intersect at a point. Such an object is non-manifold. Many researchers have pointed out that non-manifold objects are required in various types of applications to model the complicated topological structure of spatial objects [10], [11], [18] in solid modeling, computer vision and computer simulation. Non-manifold boundary-representation models [10], [11], [18] have been proposed in order to describe three-dimensional objects in the three-dimensional solid modeling area. However, non-manifold boundary-representation models cannot represent mixed-dimensional objects (also called *non-regular non-manifold objects* [8]), although this is possible using the cell-based models.

The two cell-based models, the linear constraint model [2], [9], [15] and the cell complex model [1], [7], [14], [19], have already been implemented in several spatial database systems. In both of these models, cells are combined to form a spatial object. In the linear con-

Received February 23, 2006; Revised April 7, 2006; Accepted May 12, 2006.
1) *kaneko@is.kyushu-u.ac.jp*
2) *akifumi@is.kyushu-u.ac.jp*
DOI: 10.2201/NiiPi.2006.3.3

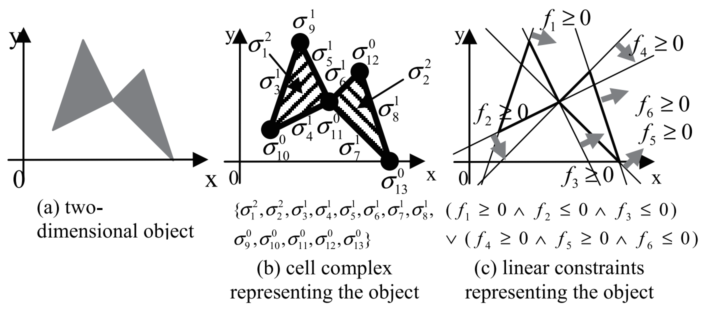

(a) two-dimensional object
$\{\sigma_1^2,\sigma_2^2,\sigma_3^1,\sigma_4^1,\sigma_5^1,\sigma_6^1,\sigma_7^1,\sigma_8^1,\sigma_9^0,\sigma_{10}^0,\sigma_{11}^0,\sigma_{12}^0,\sigma_{13}^0\}$
(b) cell complex representing the object
$(f_1 \geq 0 \wedge f_2 \leq 0 \wedge f_3 \leq 0)$
$\vee (f_4 \geq 0 \wedge f_5 \geq 0 \wedge f_6 \leq 0)$
(c) linear constraints representing the object

Fig. 1 Two-dimensional object example. The object in Fig. 1 (a) is represented by a cell-complex in Fig. 1 (b) and linear constraints in Fig. 1 (c).

straint models, cells may *overlap* (i.e., the interiors of two different cells may intersect). In the cell complex model, cells do not overlap. The intersection of two different member cells in a cell complex is disjoint, or their boundaries intersect (not their interiors). A cell complex includes cells that represent the intersections of any two member cells in the cell complex (as explained in Section 2). Such cells describe the topological structure of spatial objects explicitly. However, the linear constraint model does not contain such cells. Thus, the model does not represent topological structure information explicitly.

We implemented a spatial database system called Hawk Eye, in which the cell complex model is implemented. The system is intended to handle spatial objects of various dimensions, from 0 to d, in d-dimensional space (d = 2, 3, 4). In the system, various types of spatial objects, such as the point, line segment, polygon (convex or concave), and polyhedron (convex or concave), are represented uniformly by the cell complex model, which also enables the topological structure of both manifold and non-manifold (regular non-manifold and non-regular non-manifold) spatial objects to be represented uniformly.

Although previous cell-based spatial database systems are able to store spatial objects in three-dimensional space or four-dimensional space, they cannot respond efficiently to spatial queries and manipulations in some situations. In order to respond to spatial queries and manipulations, efficient processing of binary spatial operations between two objects is a particularly important problem. However, evaluation of binary spatial operations of three or more dimensions is time-consuming when using previous algorithms. [5] In previous cell-based spatial database systems, the binary spatial operations are not implemented [1], [7], [14], [19] (some unary spatial operations are implemented in the systems), or an algorithm that works only for two dimensions [9] is implemented (which enables two-dimensional objects to be handled in two-dimensional or higher space), or an algorithm that enables only two spatial operations, `intersection` and `union`, on the linear constraint model is implemented (which is not efficient for the cell complex model). [2], [15]

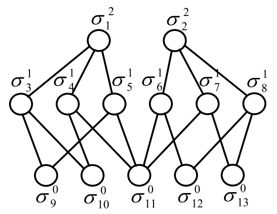


Fig. 2 Incidence graph of the cell complex in Fig. 1 (b). The cell complex consists of two two-dimensional cells, six one-dimensional cells, and five zero-dimensional cells. The incidence graph represents the topological relationship among the cells.

In the present paper, we introduce a new algorithm, called the Cell Splitting and Merge Algorithm (CSMA), by which to evaluate binary spatial operations between two cell complexes. We use the cell splitting algorithm of Chandrajit Bajaj et al. in the CSMA so that the CSMA will work efficiently in three and four dimensions. In addition, some extensions are added to the cell splitting algorithm in order to implement the CSMA.

The present paper is organized as follows. Section 2 presents related studies, and Section 3 explains the system architecture of Hawk Eye. Section 4 explains the CSMA in relation to the evaluation of the three binary spatial operations: `intersection`, `difference`, and `union`. Section 5 presents the experimental results obtained using the CSMA.

2 Related works

In this section, we explain the linear constraint, the cell and the cell complex. In addition, we explain previous algorithms for binary spatial operations for the cell-based models.

2.1 Linear constraints and cell complex

In the present study, we use the following notation: $\boldsymbol{E}^d$ represents a d-dimensional Euclidean space, σ represents a cell, and Γ represents a cell complex. In addition, $p = (x_1, x_2, \ldots, x_d)$ represents a point in $\boldsymbol{E}^d$, and $f_j(p) = a_{j,1}x_1 + a_{j,2}x_2 + \ldots + a_{j,d}x_d + a_{j,d+1}$ (there exists $a_{j,k}$ such that $a_{j,k} \neq 0$) represents a linear function. A linear function in $\boldsymbol{E}^d$ has $d + 1$ coefficients. A hyperplane h_j, which is a $(d-1)$-dimensional linear subspace in $\boldsymbol{E}^d$ is a set of points $\{p|f_j(p) = 0\}$. We use the term *hyperplane number* to refer the number j assigned to each linear function $f_j(p)$. The three comparison operators, $\geq$, $=$ and $\leq$ are used in linear constraints. A linear constraint term is expressed as $f_j(p) \geq$, $f_j(p) = 0$, or $f_j(p) \leq 0$. A linear constraint is a disjunction of a finite number of linear constraint terms, such as $\Phi_k = f_{k,1}(p)\theta_{k,1}0 \wedge f_{k,2}(p)\theta_{k,2}0 \wedge \ldots \wedge f_{k,mk}(p)\theta_{k,mk}0$ ($\theta_{k,i}$ is comparison operator, $\geq$, $=$, or $\leq$ for $1 \leq i \leq m_k$).

A cell is a set of points that satisfy a linear constraint. We use the notation σ_k for the cell defined by Φ_k (the subscript k indicates the cell number). We assume that a linear constraint that defines a cell does not contain redundant terms. A linear constraint $\Phi_k = f_{k,1}(p)\theta_{k,1}0 \wedge f_{k,2}(p)\theta_{k,2}0 \wedge \ldots \wedge f_{k,mk}(p)\theta_{k,mk}0$ *does not contain redundant terms* if any term $f_{k,i}(p)\theta_{k,i}0$ is removed from Φ_k. Then, the solution set is different for any i ($1 \leq i \leq m_k$). If the linear constraint Φ_k has n different linear equations (i.e., there are n equal comparison operators), then the dimension of the cell is $d - n$. In the present paper, the dimension of σ_k is denoted as $dim(\sigma_k)$, an s-dimensional cell is also referred to as an s-cell, and we sometimes use σ_k^s to refer to an s-cell for which the cell number is k.

A cell may be bounded or unbounded. A cell σ_k is said to be *bounded* when there exists a real number r that is larger than the distance between any pair of points in σ_k. Otherwise, σ_k is *unbounded*.

Let Φ_1 and Φ_2 be two linear constraints $f_{1,1}(p)\theta_{1,1}0 \wedge f_{1,2}(p)\theta_{1,2}0 \wedge \ldots \wedge f_{1,m1}(p)\theta_{1,m1}0$ and $f_{2,1}(p)\theta_{2,1}0 \wedge f_{2,2}(p)\theta_{2,2}0 \wedge \ldots \wedge f_{2,m2}(p)\theta_{2,m2}0$ ($m_2 \leq m_1$), respectively. A cell σ_2 is a *face* of σ_1 if the cell satisfies all of the following conditions:

1. $f_{1,i}(p) = f_{2,i}(p)$ for any i ($1 \leq i \leq m_2$).
2. $\theta_{1,i} = \theta_{2,i}$ or $\theta_{2,i}$ is '$=$' for any i ($1 \leq i \leq m_2$).
3. $\{p|f_{1,i}(p)\theta_{1,i}0\} \supset \{p|\Phi_2(p)\}$ for any i ($m_2 + 1 \leq i \leq m_1$) if $m_2 < m_1$.
4. $\{p|\Phi_2(p)\}$ is not empty.

For example, there are seven faces of cell σ_1^2 in Fig. 1 (b). They are σ_1^2, σ_3^1, σ_4^1, σ_5^1, σ_9^0, σ_{10}^0, and σ_{11}^0. The cells are (1) $\sigma_1^2 : f_1(p) \geq 0 \wedge f_2(p) \leq 0 \wedge f_3(p) \leq 0$, (2) $\sigma_3^1 : f_1(p) = 0 \wedge f_2(p) \leq 0 \wedge f_3(p) \leq 0$, (3) $\sigma_4^1 : f_1(p) \geq 0 \wedge f_2(p) = 0 \wedge f_3(p) \leq 0$, (4) $\sigma_5^1 : f_1(p) \geq 0 \wedge f_2(p) \leq 0 \wedge f_3(p) = 0$, (5) $\sigma_9^0 : f_1(p) = 0 \wedge f_3(p) = 0$ (6) $\sigma_{10}^0 : f_1(p) = 0 \wedge f_2(p) = 0$, and (7) $\sigma_{11}^0 : f_2(p) = 0 \wedge f_3(p) = 0$. A cell σ_{k2} is a *direct face* of σ_{k2} if σ_{k1} is a face of σ_{k1} and $dim(\sigma_{k2}) = dim(\sigma_{k1}) - 1$. For example, the direct faces of σ_1^2 in Fig. 1 (b) are the following three one-dimensional cells: σ_3^1, σ_4^1, and σ_5^1.

A cell complex is a set of bounded cells. [12] A set of bounded cells $\Gamma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ is a cell complex if Γ satisfies the following two conditions. [12]

1. For any member cell σ_k in Γ ($1 \leq k \leq n$), any face of σ_k is a member cell in Γ. For example, the cell complex in Fig. 1 (b) includes all seven faces of cell σ_1^2 as its member cells.
2. For any two member cells σ_{k1} and σ_{k2} in Γ ($1 \leq k_1, k_2 \leq n$), the intersection of σ_{k1} and σ_{k2} (i.e., $\Phi_{k1} \wedge \Phi_{k2}$) is a face of the two cells σ_{k1} and σ_{k2} if the intersection of σ_{k1} and σ_{k2} is not empty. For example, the intersection of the two cells σ_1^2 and σ_2^2 in Fig. 1 (b) is the member cell σ_{11}^0. The cell σ_{11}^0 is the face of the two cells σ_1^2 and σ_2^2.

A cell complex Γ is s-dimensional if the maximum dimension of the cells in Γ is s. A cell σ in Γ is called a *top cell* if σ is not the face of any other cell in Γ. The top cells in the cell complex in Fig. 1 (b) are σ_1^2 and σ_2^2.

Let a cell complex Γ have n_t top cells as its member, from σ_1 to σ_{nt}, σ_1, σ_2, $\ldots$, σ_{nt}. The linear constraint of the top cell σ_k is denoted by $\Phi_k = f_{k,1}(p)\theta_{k,1}0 \wedge f_{k,2}(p)\theta_{k,2}0 \wedge \ldots \wedge f_{k,mk}(p)\theta_{k,mk}0$ ($1 \leq k \leq n_t$, where m_k is the number of linear constraint terms in Φ_k). Two linear constraints of two different top cells may contain the same linear function (i.e., $f_{k1,i1}(p) = f_{k2,i2}(p)$ and $k_1 \neq k_2$). In such a case, the total number of distinct linear functions of Γ is less than the total number of linear constraint terms ($= \Sigma m_k$). We hereinafter use n_c to refer to the total number of linear constraint terms ($= \Sigma m_k$) and n_h to refer the total number of distinct linear functions. The hyperplane number associated with Γ is an integer from 1 to n_h. For example, the cell complex in Fig. 1 (b) has two top cells: σ_1^2 and σ_2^2. Their linear constraints are $f_1(p) \geq 0 \wedge f_2(p) \leq 0 \wedge f_3(p) \leq 0$ and $f_4(p) \geq 0 \wedge f_5(p) \geq 0 \wedge f_6(p) \leq 0$, respectively. In this example, the number of top cells is $n_t = 2$, and $n_c = n_h = 6$. In this example, the hyperplane number is an integer from 1 to 6.

The relationship among member cells in a cell complex can be represented graphically using an incidence graph [4], [5], [16]. In the incidence graph of a cell complex, each node represents a cell in the cell complex, and there is an arc between two cells when one cell is a direct face of the other. The incidence graph of the cell complex in Fig. 1 (b) is illustrated in Fig. 2. In this example, thirteen nodes represent the member cells, and arcs represent the direct face relationship.

2.2 Binary spatial operation for the cell-based models

Linear constraint may be either satisfiable or unsatisfiable. If the solution set of a linear constraint is empty, then the linear constraint is unsatisfiable, otherwise it is satisfiable. For example, the term "$x \geq 1 \wedge x \leq 0$" is unsatisfiable. In order to check the satisfiability, several linear constraint solving algorithms have been proposed, including half-space intersection, convex hull, linear programming, and integer programming. The constraint solving algorithms can be used to evaluate the binary spatial operation, `intersect`, between two cells. The operation returns the `true` value when two cells are not disjoint. In addition, the constraint solving algorithms can be used to evaluate the binary spatial operation, `intersection`, for the cell-based models. Let two spatial objects represented by a cell-based model be $\Phi_{1,1} \vee \Phi_{1,2} \vee \ldots \vee \Phi_{1,m}$, and $\Phi_{2,1} \vee \Phi_{2,2} \vee \ldots \vee \Phi_{2,n}$. The `intersection` of the two is $(\Phi_{1,1} \vee \Phi_{1,2} \vee \ldots \vee \Phi_{1,m}) \wedge (\Phi_{2,1} \vee \Phi_{2,2} \vee \ldots \vee \Phi_{2,n}) = (\Phi_{1,1} \wedge \Phi_{2,1}) \vee (\Phi_{1,1} \wedge \Phi_{2,2}) \vee \ldots \vee (\Phi_{1,1} \wedge \Phi_{2,n}) \vee \ldots \vee (\Phi_{1,m} \wedge \Phi_{2,1}) \vee (\Phi_{1,m} \wedge \Phi_{2,2}) \vee \ldots \vee (\Phi_{1,m} \wedge \Phi_{2,n})$, which contains $m \times n$ linear constraints. A linear constraint solving algorithm is invoked $m \times n$ times to eliminate the unsatisfiable linear constraints. The half-space intersection algorithm works efficiently for two-dimensional objects. However, the algorithm does not work for three dimensions or higher. The other linear constraint solving algorithms, convex hull, linear programming, and integer programming work for two dimensions or higher. However, they are time-consuming for three or higher dimension, and so have not been used in spatial database systems to our knowledge. In the DEDALE system, the half-space intersection algorithm is used to evaluate `intersection` for the linear constraint model. [9]

The hyperplane arrangement construction algorithm by Edelsbrunner [5] can be used to evaluate binary spatial operations `intersection`, `difference`, and `union` for the cell-based models. However, using the algorithm for binary spatial operations in three or higher dimensions is very time-consuming. For d dimensional space, $O((n_{1,h} + n_{2,h})^d)$ cells are constructed [5] in an intermediate hyperplane arrangement ($n_{1,h}$ and $n_{2,h}$ are the numbers of linear functions that are used to define the two spatial objects to be evaluated) to evaluate binary spatial operations.

Chandrajit Bajaj et al. presented a cell splitting algorithm by which to split a bounded cell with a hyperplane. [4] Splitting a cell with a hyperplane is a spatial operation in which a cell is split into two cells, so that any line between any point in one cell and any point in the other cell intersects the hyperplane. The cell splitting algorithm can be used for binary spatial operations between cells. In the algorithm, the cost to evaluate binary spatial operations between cells is $O((m_{k1} + m_{k2})^{d/2})$ (for $d = 2, 4, \ldots$) or $O((m_{k1} + m_{k2})^{(d-1)/2})$ (for $d = 3, 5, \ldots$) [5] (m_{k1} and m_{k2} are the numbers of linear functions used to define the two cells). Nirenstein et al. used the cell splitting algorithm for the purpose of the from-region visibility query. [13] In their paper, the binary spatial operation `difference` between cells is evaluated for the query.

In the present paper, we introduce a new efficient algorithm CSMA by which to evaluate binary spatial operations between two cell complexes. We use the cell splitting algorithm in the implementation of CSMA so that the CSMA will work efficiently in three and four dimensions. Because the cell splitting algorithm is intended to cell, some extensions are added to the cell splitting algorithm to handle the cell complexes.

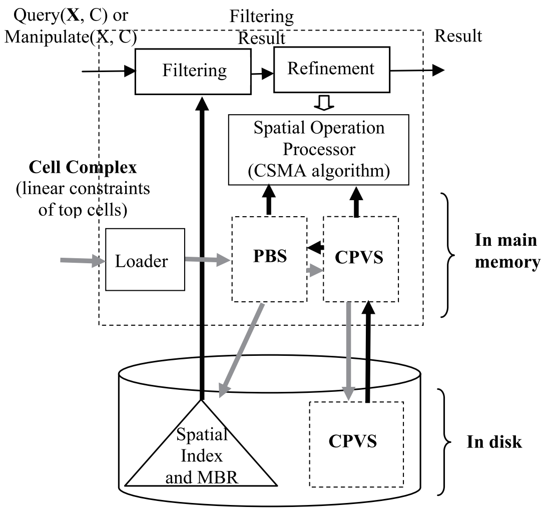


Fig. 3 System architecture of Hawk Eye.

3 System architecture

Query and manipulation of spatial objects are processed in two steps in Hawk Eye, just like other spatial database systems (see Fig. 3). These two steps are filtering and refinement [3]. In the filtering step, the spatial index of spatial objects and the Minimum Bounding Rectangle (MBR) are used. In the refinement step, the spatial operation processor module is invoked to evaluate binary geometric operations.

Two representations for the incidence graph are implemented in Hawk Eye. They are the PBS and the

CPVS. The CPVS is an in-disk representation, and the PBS is an in-memory representation. When query and manipulation of spatial objects are executed, spatial index, the MBR and the CPVS stored in a database are used. For example, to evaluate a spatial query `AREA (X, Y)` (**`X`** is a set of spatial objects, and `Y` is one spatial object), which means "calculate the total size of the intersection of `Y` and objects in **`X`**", the spatial index and MBRs of **`X`** are used in the filtering step. In this step, objects in **`X`** are eliminated such that its MBR does not intersect with `Y`. In the refinement step, the filtering result and `Y` are used. In this step, for each object $\mathtt{x}_k$ in the filtering result, `intersection` ($\mathtt{x}_k$, `Y`) is evaluated. To evaluate the operation, the PBS of `Y` is produced from the CPVS of `Y` on-the-fly. Then, the CPVS of $\mathtt{x}_k$ and the PBS of Y are used in the CSMA algorithm.

3.1 CPVS

The CPVS has two parts: `TopCell` and `Hyperplane`. The `TopCell` part of the CPVS stores the linear constraints of the top cells. One tuple of `TopCell` is equivalent to one linear constraint term. For a top cell σ_k, there are m_k tuples to store the m_k linear constraint terms $f_{k,1}(p)\theta_{k,1}0$, $f_{k,2}(p)\theta_{k,2}0, \ldots,$ and $f_{k,mk}(p)\theta_{k,mk}0$ of σ_k. In total, `TopCell` contains n_c ($= \Sigma m_k$) tuples. The tuple has three attributes, `TopCellNumber`, `HyperplaneNumber`, and `ComparisonOperator`, which store the top cell number, the hyperplane number, and the comparison operator, $\geq$, $=$, or $\leq$, respectively. For a linear constraint term $f_{k,i}(p)\theta_{k,i}0$ ($1 \leq k \leq n_t$ and $1 \leq i \leq m_k$), the tuple is $<k$, 'k, i', $\theta_{k,i}>$. In Table 1, there are six ($= n_c$) tuples. The first tuple is $<1, 1, \geq>$, which means that the cell having the top cell number of 1 has the linear constraint term of $f_1 \geq 0$.

The `Hyperplane` part of the CPVS stores the hyperplane numbers and the coefficients of the linear functions. A hyperplane in $\boldsymbol{E}^d$ has $(d + 1)$ coefficients. In the `Hyperplane` in Table 1, there are six ($= n_h$) tuples, each of which has three coefficients.

3.2 PBS

The PBS has two parts: `Graph` and `Hyperplane`. The `Hyperplane` part of the PBS is the same as that of the CPVS (see Table 1 (b)). The `Graph` part of the PBS has the three attributes to store node attributes of the incidence graph. They are coordinate value, cell position vector, and pointers to represent arcs between nodes. Only the nodes representing the zero-dimensional cell have the coordinate values. Other nodes do not have the coordinate values. Table 2 shows the node attributes of the incidence graph in Fig. 2. The

Table 1 Example of CPVS. The `TopCell` part stores linear constraints of top cells. The `Hyperplane` part stores the coefficients of linear functions.

(a) `TopCell`

Linear Constraint	**Attributes**		
	`Top Cell Number`	**`Hyperplane Number`**	**`Comparison Operator`**
$f_1(p) \geq 0 \wedge f_2(p) \leq 0 \wedge f_3(p) \leq 0$	1	1	$\geq$
	1	2	$\leq$
	1	3	$\leq$
$f_4(p) \geq 0 \wedge f_5(p) \geq 0 \wedge f_6(p) \leq 0$	2	4	$\geq$
	2	5	$\geq$
	2	6	$\leq$

(b) `Hyperplane`

Linear constraint term	**Attributes**			
	`Hyperplane Number`	**`Coefficients`**		
		$a_{j,1}$	$a_{j,2}$	$a_{j,3}$
$3x - y - 2 \geq 0$	1	3	−1	−2
$x - 2y + 1 \leq 0$	2	1	−2	1
$2x + y - 8 \leq 0$	3	2	1	−8
$x - y - 1 \geq 0$	4	1	−1	−1
$x + y - 5 \geq 0$	5	1	1	−5
$3x + y - 15 \leq 0$	6	3	1	−15

Table 2 The `Graph` part of the PBS has three node attributes. They are coordinate value, cell position vector, and pointers to represent arcs. The table shows the coordinate value and the cell position vector of the incidence graph in Fig. 2. Pointers to represent arcs are omitted from this table.

	Linear constraint	**Node Attributes**	
		Coordinate Value	**Cell Position Vector**
σ_1^2	$f_1(p) \geq 0 \wedge f_2(p) \leq 0 \wedge f_3(p) \leq 0$		`[+ − − i i i]`
σ_2^2	$f_4(p) \geq 0 \wedge f_5(p) \geq 0 \wedge f_6(p) \leq 0$		`[i i i + + −]`
σ_3^1	$f_1(p) = 0 \wedge f_2(p) \leq 0 \wedge f_3(p) \leq 0$		`[0 − − i i i]`
σ_4^1	$f_1(p) \geq 0 \wedge f_2(p) = 0 \wedge f_3(p) \leq 0$		`[+ 0 − i i i]`
σ_5^1	$f_1(p) \geq 0 \wedge f_2(p) \leq 0 \wedge f_3(p) = 0$		`[+ − 0 i i i]`
σ_6^1	$f_4(p) = 0 \wedge f_5(p) \geq 0 \wedge f_6(p) \leq 0$		`[i i i 0 + −]`
σ_7^1	$f_4(p) \geq 0 \wedge f_5(p) = 0 \wedge f_6(p) \leq 0$		`[i i i + 0 −]`
σ_8^1	$f_4(p) \geq 0 \wedge f_5(p) \geq 0 \wedge f_6(p) = 0$		`[i i i + + 0]`
σ_9^0	$f_1(p) = 0 \wedge f_3(p) = 0$	(2, 4)	`[0 i 0 i i i]`
σ_{10}^0	$f_1(p) = 0 \wedge f_2(p) = 0$	(1, 1)	`[0 0 i i i i]`
σ_{11}^0	$f_2(p) = 0 \wedge f_3(p) = 0 \wedge f_4(p) = 0 \wedge f_5(p) = 0$	(3, 2)	`[i 0 0 0 0 i]`
σ_{12}^0	$f_4(p) = 0 \wedge f_6(p) = 0$	(4, 3)	`[i i i 0 i 0]`
σ_{13}^0	$f_5(p) = 0 \wedge f_6(p) = 0$	(5, 0)	`[i i i i 0 0]`

zero-dimensional cells σ_9^0, σ_{10}^0, σ_{11}^0, σ_{12}^0 and σ_{13}^0 have coordinate values of (2,4), (1,1), (3,2), (4,3), and (5.0), respectively.

The cell position vector in the PBS represents the linear constraint of a cell. For example, the cell position vector of the top cell σ_1^2 in Fig. 1 (b) is `[+ − − i i i]`, which represents "$f_1(p) \geq 0 \wedge f_2(p) \leq 0 \wedge f_3(p) \leq 0$".

The definition of the cell position vector is as follows. The cell position vector of a member cell σ_k in Γ is an n_h-length vector, such as $\boldsymbol{u}_k = [u_{k,1}, u_{k,2}, \cdots, u_{k,nh}]$. The element value is +, 0, −, or `i`. For a cell σ_k in Γ, the j-th element value $u_{k,j}$ $(1 \leq j \leq n_h)$ is as follows:

(1) $u_{k,j}$ = '`+`' if Φ_k contains the term $f_j(p) \geq 0$.

(2) $u_{k,j}$ = '`0`' if Φ_k contains the term $f_j(p) = 0$.

(3) $u_{k,j}$ = '`−`' if Φ_k contains the term $f_j(p) \leq 0$.

(4) $u_{k,j}$ = '`i`', otherwise.

For example, in Table 2, the length of the cell position vectors is 6 (= n_h). The cell σ_9^0 is `[0 i 0 i i i]`. The second element is '`i`' because the linear constraints "$f_1(p) = 0 \wedge f_3(p) = 0$" of σ_9^0 do not contain the linear function 'f_2'.

3.3 Conversion between PBS and CPVS

The `Hyperplane` part of the CPVS and PBS is same. To convert the PBS of a cell complex to its CPVS, the `Graph` part of the PBS is converted to the `TopCell` part of its CPVS (see Fig. 4). The conversion algorithm is shown in Fig. 5. There are two steps in the conversion.

(1) Selection of top cells

In this step, top cells are selected from the `Graph` part of the PBS. The node in the `Graph` representing a top cell does not have upward pointers in the node. The selection is done by checking the upward pointers.

(2) Insertion of tuples

In this step, tuples are inserted into the `TopCell` part of the CPVS. The cell position vectors of the top cells that are selected in the previous step are used to generate the tuples to be inserted.

To convert the CPVS of a cell complex to its PBS, the `Graph` part of the PBS is generated from the `TopCell` part and the `Hyperplane` part of its CPVS (see Fig. 4). We use the extended cell splitting algorithm [17] which enables to split an unbounded cell with a hyperplane for the generation.

4 Binary geometric operation algorithm

Section 4 explains the details of the CSMA with respect to the evaluation of the three geometric operations: `intersection`, `difference`, and `union`. The algorithm can also be used for the five binary topological predicates [6]: `intersect`, `disjoint`,

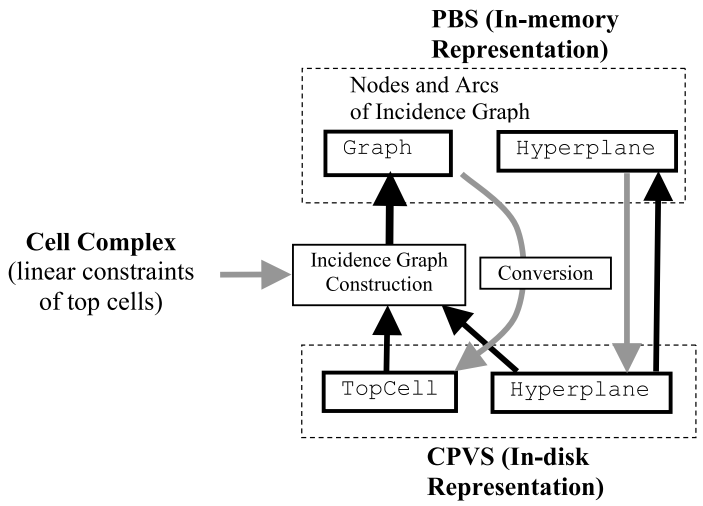

PBS (In-memory Representation)
Nodes and Arcs of Incidence Graph
Graph
Hyperplane
Cell Complex (linear constraints of top cells)
Incidence Graph Construction
Conversion
TopCell
Hyperplane
CPVS (In-disk Representation)

Fig. 4 Two representations of the incidence graph, in which the PBS and the CPVS are implemented in Hawk Eye. The two representations are converted on-the-fly.

```
Algorithm convertPBStoCPVS
input:
   Graph: the Graph part of the PBS of a cell complex $\Gamma$.
   nh: the number of hyperplanes of $\Gamma$.
output:
   TopCell: the TopCell part of the CPVS of $\Gamma$.
1. for each node $\sigma_k$ in Graph do
2.   if $\sigma_k$ does not have upward pointers do /* selection of top cells */
3.     for j = 1 to nh do
4.       if $u_{k,j}$ = '+' do insert tuple <k, j, ≥> into TopCell done
5.       if $u_{k,j}$ = '0' do insert tuple <k, j, => into TopCell done
6.       if $u_{k,j}$ = '−' do insert tuple <k, j, ≤> into TopCell done
7.     done
8. done
```

Fig. 5 Conversion from the PBS to the CPVS.

`meet`, `contains`, and `equal`.

We use the cell splitting algorithm of Chandrajit Bajaj et al. in the CSMA. Given the PBS of a cell σ (not a cell complex) and a hyperplane $f_j(p) = 0$, the cell splitting algorithm produces the new PBS of the cell complex Γ that satisfies the following:

(1) If the hyperplane intersects the cell σ ($f_j(p) > 0 \wedge \sigma$ is not empty and $f_j(p) < 0 \wedge \sigma$ is not empty), then there are two top cells in Γ, i.e., $f_j(p) \geq 0 \wedge \sigma$ and $f_j(p) \leq 0 \wedge \sigma$.

(2) Otherwise, there is only one top cell in Γ, i.e., σ.

As a result of the cell splitting, each node of the PBS of Γ has a new position as its node attribute. For a node representing a cell σ_k, the position value $v_{k,j}$ of σ_k is +, 0, or −. This value represents the relative position of the cell σ_k to the hyperplane $f_j(p) = 0$, as follows:

(1) $v_{k,j}$ = '+' if for any internal point in σ_k, $f_j(p) > 0$.

(2) $v_{k,j}$ = '−' if for any internal point in σ, $f_j(p) < 0$.

(3) $v_{k,j}$ = '0' if for any internal point in σ, $f_j(p) = 0$.

In the CSMA algorithm, a cell is split by more than one hyperplane. In order to split a cell with n_h hyperplanes, the cell splitting algorithm is invoked n_h times in our implementation. As a result of the invocation, the PBS of a cell complex, which represents the splitting result, is obtained. Each node of the PBS has a new *position vector*, which is defined as follows. The length of the position vectors is n_h. For a node of a cell σ_k in the PBS, the j-th element value $v_{k,j}$ is +, 0, or −. This value represents the relative position of the cell σ_k to the j-th hyperplane $(1 \leq j \leq n_h)$ explained above.

There is a difference between the cell splitting algorithm presented by Chandrajit Bajaj et al. and the algorithm implemented herein. As a result of the cell splitting, an arc between cells occurs, which represents the direct face relationship in the PBS. When splitting an s-dimensional cell, the cell splitting algorithm of Chandrajit Bajaj et al. creates arcs between a newly created $(s-1)$-dimensional cell and newly created $(s-2)$-dimensional cells. They assumed that the number of newly created $(s-1)$-dimensional cells is one and that all of the newly created $(s-2)$-dimensional cells have a direct face relationship between with the newly created $(s-1)$-dimensional cell. This assumption holds when the dimension $s = 2$ and the number of hyperplanes is $n_h = 1$. In our implementation, the position vector is used to check the direct face relationship between newly created cells in the PBS.

There are four steps used to evaluate the binary spatial operations between two cell complexes (see Fig. 6). The cell splitting algorithm is applied in the third step. There are two steps before the cell splitting and one step after the cell splitting in order to handle the cell complex.

(1) Conversion of the CPVS to the PBS

The first step is the conversion of the CPVS of a cell complex to its PBS. Although there are two operands in the operation, only one cell complex is converted. We hereinafter use Γ_1 to refer to the converted cell complex, and Γ_2 to refer the other cell complex.

(2) Generation of the PBS of the top cells

In this step, the PBS obtained by the previous step is used. The PBS of the incidence graphs of all of the top cells in Γ_1 is obtained from the PBS of Γ_1.

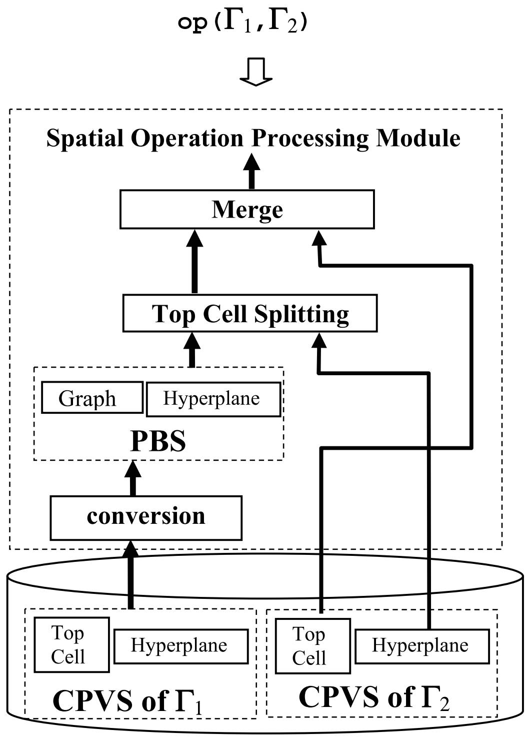


Fig. 6 Steps of the CSMA. To evaluate a binary spatial operation between Γ_1 and Γ_2, CPVS of Γ_1 is converted to the PBS. The incidence graphs represented by the PBS are then split with the hyperplanes using the `Hyperplane` part of the CPVS of Γ_2. Finally, the cells that represent the operation result are collected from the top cell splitting result using the `TopCell` part of Γ_2.

(3) Cell splitting

In this step, the PBS obtained by the previous step, and the `Hyperplane` part of the CPVS of Γ_2 are used. Since there is hyperplane information in the CPVS of Γ_2, the CPVS does not have to be converted to the PBS. The PBS of the top cells in Γ_1 is split with the hyperplanes associated with Γ_2, which is stored in the `Hyperplane` part of the CPVS of Γ_2. In this step, the cell splitting algorithm is applied repeatedly to split the PBS one-by-one by the hyperplanes.

(4) Cell Selection and Merge

In this step, the cells that represent the operation result are selected from the result of the previous step. This is done using the `TopCell` part of Γ_2. The position vector of each node is examined one-by-one, and the nodes that satisfy a particular condition are selected. The way of the selection is different operation by operation to be evaluated. For example, to evaluate `intersection`, only the cells that are

```
Algorithm intersection
input:
  PBS: the PBS of $\Gamma_1$.
  $\mathbf{H} = \{h_1, h_2, \ldots, h_{nh}\}$: the set of hyperplanes associated with $\Gamma_2$
  $\Theta = \{\theta_1, \theta_2, \ldots, \theta_{nh}\}$: the set of comparison operators associated with $\Gamma_2$
output:
  the PBS of intersection $\Gamma_1, \Gamma_2$
1. for each node $\sigma_k$ in PBS do
2.   if $\sigma_k$ does not have upward pointers do /* selection of top cells */
3.     extract the PBS of the incidence graph of $\sigma$ from the PBS,
       and store the PBS into $G_k$
4.     for j = 1 to nh do
5.       $G_k$ := CellSplit($G_k$, $h_j$)
6.     done
7.     for each node $\sigma_i$ in $G_k$ do
8.       for j = 1 to nh do
9.         if $\theta_j$ = '$\geq$' and $u_{i,j}$ = '$-$' do remove $\sigma_i$ from $G_k$
10.        if $\theta_j$ = '$\leq$' and $u_{i,j}$ = '+' do remove $\sigma_i$ from $G_k$
11.        if $\theta_j$ = '=' and $u_{i,j} \neq$ '0' do remove $\sigma_i$ from $G_k$
           /* in the case of the operation difference
             if $\theta_j$ = '$\geq$' and $u_{i,j}$ = '+' do remove $\sigma_i$ from $G_k$
             if $\theta_j$ = '$\leq$' and $u_{i,j}$ = '$-$' do remove $\sigma_i$ from $G_k$
             if $\theta_j$ = '=' and $u_{i,j}$ = '0' do remove $\sigma_i$ from $G_k$ */
12.      done
13.    done
14. done
15. merge all PBSs $G_k$ into one PBS
```

Fig. 7 CSMA for the operation `intersection`

contained in Γ_2 are selected. Finally, the PBS of each cell is merged into the PBS of a cell complex. For this purpose, the nodes in the PBS of each cell are traversed from the dimension of the top cell to dimension zero.

As explained above, the CSMA uses the CPVSs of the two cell complexes, one of which is converted to the PBS. For example, to evaluate `intersection` between Γ_1 and Γ_2, the PBS of the incidence graphs of all top cells in Γ_1 is produced from the CPVS of Γ_1. The PBS of the top cells is then split with the set of hyperplanes associated with Γ_2 in the cell splitting step. In this step, the `Hyperplane` part of the CPVS of Γ_2 is used. Then, the nodes that satisfy one of the following three conditions are obtained.

Select a cell σ in Γ_2 that satisfies the following:

(1) If the j-th comparison operator θ_j is '$\geq$', then j-th value of the position vector of the node is '+' or '0'.

(2) If the j-th comparison operator θ_j is '0', then j-th value of the position vector of the node is '0'.

(3) If the j-th comparison operator θ_j is '$\leq$', then j-th value of the position vector of the node is '–' or '0'.

To do the selection, the `TopCell` part of the CPVS of Γ_2 is used. Finally, the PBS that represents `intersection` between Γ_1 and Γ_2 is produced by merging the incidence graphs. The algorithm of the `intersection` is shown in Fig. 7.

In the case of the `difference` operation, the algorithm is same as `intersection` except for the manner of cell selection in the fourth step. Nodes that satisfy one of the following two conditions are obtained.

Find a cell σ in Γ_2 that satisfies the following:

(1) If the j-th comparison operator θ_j is '$\geq$', then j-th value of the position vector of the node is '–' or '0'.

(2) If the j-th comparison operator θ_j is '$\leq$', then j-th value of the position vector of the node is '+' or '0'.

In the case of `union`, the behavior is different. The CSMA is executed twice. In order to obtain the PBS that represents the union of two cell complexes, Γ_1 and Γ_2, the PBSs of top cells in Γ_1 is first split by the hyperplanes associated with Γ_2. Then, the PBSs of the top cells in Γ_2 are split by the hyperplanes associated with Γ_1. Finally, these two sets of PBSs are merged into one PBS. The result represents the union of two cell complexes, Γ_1 and Γ_2.

5 Experimental Evaluation

The purpose of the test is to measure the execution time of the spatial operation algorithm CSMA implemented herein. The experiments are performed on a SUN Microsystems Blade 100 Workstation (main memory: 512 Mbytes, OS: Solaris 10). Applications are implemented in C and C++ (C/C++ compiler: SUN Forte 6 Developer Update 2, Database System: ShusseUo).

5.1 Test database

The present test uses randomly generated three-dimensional cell complexes and four-dimensional cell complexes. We generated random d-dimensional cell complexes ($d = 3$ or 4) using the following procedure. The procedure has two parameters n_f and n_t, which represent the number of $(d-1)$-dimensional faces and the number of top cells, respectively. In the present test, $n_f = 10$, and $n_t = 20, 40, 60, 80, 100$, or 120.

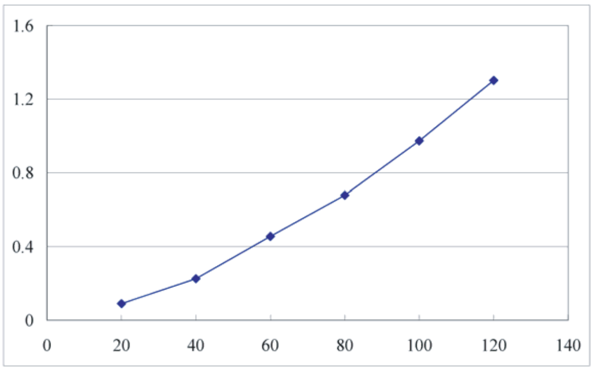
1. Construct the incidence graph representing the three-dimensional cube `C` $[(-10,-10,-10)\ (10,10,10)]$ or four-dimensional cube `C` $[(-10,-10,-10,-10)\ (10,10,10,10)]$. The number of $(d-1)$-dimensional faces of the d-dimensional cube `C` is $d \times 2$.

2. The cube `C` is split with $(n_f d \times 2)$ random hyperplanes using the cell splitting algorithm. Each random hyperplane is generated by a random point in `C` and a random normal vector that represents the orientation of the hyperplane to be generated. We use a uniformly distributed random function to generate the point and vector.

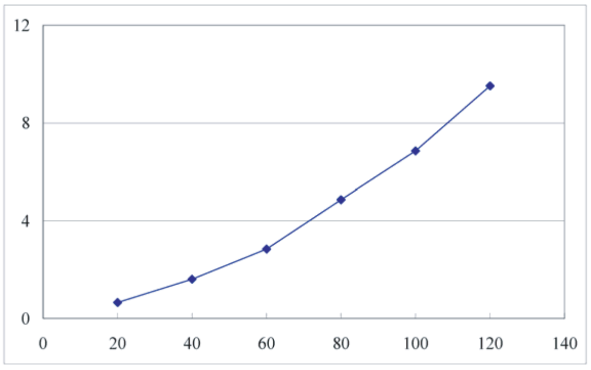
3. To produce a random d-dimensional cell complex, we randomly select n_t d-dimensional cells as its top cells using the incidence graph representing `C`.

5.2 Execution time to evaluate spatial operation

The execution time of the operation `intersection` using the CSMA algorithm is measured. Figure 8 shows the performance curve of the operation `intersection` between two random three-dimensional complexes or two random four-dimensional complexes. The horizontal axis shows the number of top cells of both cell complexes, and the vertical axis shows the average response time in seconds (ten random cell com-



(a) three-dimensional cell complexes



(b) four-dimensional cell complexes

Fig. 8 Performance curve of the binary spatial operations `intersection` between two random three-dimensional cell complexes or four-dimensional cell complexes. The horizontal axis is the number of top cells of both cell complexes, and the vertical axis is the response time in seconds.

plexes were used for each test).

The results of this test show that when the CSMA is used for the operation `intersection` between two three-dimensional cell complexes and four-dimensional cell complexes, the response time is proportional to the number of top cells in one cell complex multiplied by the number of top cells in the other in both the three- and four-dimensional tests. In this test, the CSMA algorithm works efficiently for random cell complexes. Theoretical analysis of the performance of the CSMA and further performance tests of CSMA using practical data will be performed in the future.

6 Conclusion

We implemented a new spatial database system called Hawk Eye. The cell complex model is implemented in the system to represent spatial objects. The incidence graph is used in the new system. In addition, we presented a new algorithm, called the Cell Splitting and Merge Algorithm (CSMA), to evaluate binary spatial operations between two cell complexes. The cell splitting algorithm of Bajaj Chandrajit et al. is used, which enables cells of three or four dimensions to be manipulated efficiently. The CSMA algorithm can be used to produce a cell complex that represents the evaluation results of three binary geometric operations: `intersection`, `difference`, and `union`.

Further analysis of the performance of the CSMA will be performed in the future. We plan to store a three-dimensional CAD dataset and a medical image dataset in Hawk Eye and carry out performance tests of the CSMA using these datasets. There are three research areas that can accelerate spatial date processing: the development of a spatial index to manage CPVS, the development of a technique to select the fastest execution plan, and the development of a data compression method to reduce the disk I/O cost of CPVS.

Acknowledgements

This research was supported in part by a Grant-in-Aid for Scientific Research (17700117) from the Japan Society for the Promotion of Science.

References

[1] E. Birsson, "Representing Geometric Structures in d Dimensions: Topology and Order," *Discrete Comput. Geom.*, vol.9, no.4, pp.387–426, 1993.

[2] A. Brodsky, V. Segal, J. Chen, and P. Exarkhopoulo, "The CCUBE Constraint Object-oriented Database Systems", *Proc. 1999 SIGMOD*, pp.577–579, 1999.

[3] T. Brinkhoff, H.-P. Kriegel, R. Schneider, and B. Seeger, "Multi-step Processing of Spatial Joins," *Proc. 1994 ACM SIGMOD*, pp.197–208, 1993.

[4] C.L. Bajaj and V. Pascucci, *Splitting a Complex of Convex Polytopes In Any Dimension*, ACM Press, 1996.

[5] H. Edelsbrunner, *Algorithms in Combinatorial Geometry*, Springer-Verlag, 1987.

[6] M. Egenhofer, "A Formal Definition of Binary Topological Relationships," *Proc. 3rd Intl. Conf. on Foundations of Data Organization and Algorithms*, pp.457–472, 1989.

[7] L.D. Floriani and A. Hui, "a Scalable Data Structure for Three-dimensional Non-manifold Objects," *Proceedings of the Eurographics/ACM SIGGRAPH Symposium on Geometry processing*, pp.72–82, 2003.

[8] L.C. Glaser, *Geometric Combinatorial Topology*, Van Nostrand Reinhold, New York, 1970.

[9] S. Grumback, P. Rigaux, and L. Segoufin, "The DEDALE System for Complex Spatial Queries," *Proc. ACM SIGMOD Intl. Conf. on Management of Data*, pp.213–224, 1998.

[10] E.L. Gursoz, Y. Choi, and F.B. Prinz, "Vertex-based Representation of Non-manifold Boundaries," In M.J. Wozny, J.U. Turner, and K. Preiss, Ed., *Geometric Modeling for Product Engineering*, Elsevier Science Publishers B.V., North Holland, pp.107–130, 1990.

[11] S.H. Lee and K. Lee, "Partial Entity Structure: a Compact Non-manifold Boundary Representation based on Partial Topological Entities," In *Proceedings Sixth ACM Symposium on Solid Modeling and Applications*. 2001.

[12] A.T. Lundell and S. Weingram, *The Topology of CW Complexes*, Van Nostrand Reinhold Comp., 1969.

[13] S. Nirenstein, E. Blake, and J. Gain, "Exact From-region Visibility Culling," *Proceedings of the 13th Eurographics workshop on Rendering*, 2002.

[14] A. Raza and W. Kainz, "Cell Tuple based Spatio-temporal Data Model: an Object Approach," *Proceedings of the seventh ACM international symposium on Advances in geographic information systems*, pp.20–25, 1999.

[15] P. Revez, *Introduction to Constraint Databases*, Springer-Verlag, 2002.

[16] S. Skiena, *Hasse Diagrams, in Implementing Discrete Mathematics: Combinatorics and Graph Theory with Mathematica*, Addison-Wesley, 1990.

[17] M. Tanaka, K. Kaneko, Y. Lu, and A. Makinouchi, "An Extended Cell Splitting Algorithm for Spatial Databases," *IEEE TENCON 2004*, pp.371–374, 2004.

[18] K. Weiler, *The Radial Edge Data Structure: A Topological Representation for Non-manifold Geometric Boundary Modeling*, J.L. Encarnacao, M.J. Wozny, H.W. McLaughlin, Ed., Geometric Modeling for CAD Applications, pp.3–36. Elsevier Science Publishers B.V. (North Holland), Amsterdam, 1988.

[19] M. Worboys, "A Unified Model for Spatial and Temporal Information," *Comput. J.*, vol.37, no.1, pp.26–34, 1994.

Kunihiko KANEKO

received his Ph.D. degree form Kyushu University. Since 1996, he has been with the Graduate School of Information Science and Electrical Engineering, Kyushu University, Japan, where he is an associate professor. His research interests includes spatial databases, and biomedical databases. He is a member of IPSJ, IEICE, ACM, and IEEE.

Akifumi MAKINOUCHI

received his B.E. degree from Kyoto University, Japan, in 1967, Docteur-Ingereur degree from Univercite de Grenoble, France, in 1970, and D.E. degree from Kyoto University, Japan, in 1985. Since 1989 to 2006, he was a professor in the Graduate School of Information Science and Electrical Engineering, Kyushu University, Japan. Since 2006, he has been with Kurume Institute of Technology where he is a professor. His research interests include spatial databases, multi-dimensional index, database systems for medical applications, and medical data science. He is a member of IPSJ, IEICE, ACM, and IEEE.