An Efficient Signature File Strategy for Similarity Retrieval from Large Iconic Image Databases [1]

Ye-In Chang[†], Hsing-Yen Ann[‡], and Wei-Horng Yeh[†]

[†]Dept. of Computer Science and Engineering
National Sun Yat-Sen University
Kaohsiung, Taiwan
Republic of China
{E-mail: changyi@cse.nsysu.edu.tw}
{Tel: 886-7-5252000 (ext. 4334)}
{Fax: 886-7-5254301}

[‡]Dept. of Applied Mathematics
National Sun Yat-Sen University
Kaohsiung, Taiwan
Republic of China

Abstract

In the previous signature-based approaches to retrieve symbolic pictures from a large iconic image database, they only consider pictures of type-2 similarity, where a picture A is of type-2 similarity with picture B when both pictures have the same spatial relationship between any two objects in the pictures. When queries of objects, type-0 similarity and type-1 similarity are asked, signatures for objects, type-0 similarity and type-1 similarity are required. Note that a picture is of type-0 similarity when all the spatial category relationships of each object pairs are the same to the query picture, and a picture is of type-1 similarity when the picture is of type-0 similarity and all the orthogonal relationships of each object pairs are the same to the query picture. Although these 4 kinds of signatures can be constructed and stored in advance to speed up answering such kinds of queries as in Lee et al.'s approach [33], such a large signature file containing those 4 kinds of signatures really wastes space. In this paper, we first present the revised version of Y. I. Chang and Yang's bit-string-based access strategy [16], which constructs type-2 signatures. Based on the revised version, we then propose algorithms to dynamically convert those type-2 signatures into *object* signatures, *type-0* signatures and *type-1* signatures at run time so that we can answer queries of object similarity, type-0 similarity, type-1 similarity and type-2 similarity by storing only one type of signatures, i.e., the type-2 signatures. While in Lee et al.'s signature file strategy based on the 2D B-string representation [33], they have to construct and store 4 types of signatures in the database to achieve the same goal. From our simulation, we show that our approach can provide a higher rate of a correct match and needs a smaller storage requirement than Lee et al.'s approach.

(*Keywords*: access methods, content-based retrieval, 2D strings, iconic indexing, image databases, signatures, similarity retrieval)

[1] This research was supported in part by the National Science Council of Republic of China under Grant No. NSC-87-2213-E-110-014.

1 Introduction

An image database system is concerned with the problem of storage, retrieval, and manipulation of pictorial data in an efficient manner [1, 5]. Applications which use image databases include office automation, computer aided design, robotics, and medical pictorial archiving. Current approaches [22, 24, 23, 35, 37, 40, 41, 42] to content-based image retrieval differ in terms of image features extracted, the level of abstraction manifested in the features, and the desired degree of domain independence. There are two major categories of features: *primitive* and *logical* [19, 26]. Primitive image features such as object color, texture and shape, which are usually visual features, can be extracted automatically by image processing or semi-automatically. Logical features are abstract representations (or derived attributes) of images at various levels of detail, which contains the spatial relationship features [35]. For example, the QBIC system [22] focused on the primitive image features; while the Intelligent Image Database System (IIDS) [8] covered the logical features. (Note that logical features such as spatial relationship [40] may be synthesized from primitive features, whereas others can only be obtained through considerable human involvement.)

To answer a query "find images containing a yellow ball on the top of a green table," we may first retrieve those images which contain a ball on the top of a table. Next, from those results, we then check the colors (the primitive feature) of the ball and the table. Therefore, the perception of spatial relationships among objects, like the IIDS approach which provides a high level object-oriented search (rather than search based on the low level image primitives of objects), is one of the important criteria to discriminate the images. Consequently, a new data structure, called *iconic indexing*, which preserves the objects' spatial knowledge embedded in iconic images, is required.

Recently, many iconic indexing strategies for iconic image databases have been proposed. (Note that to simplify the design of an efficient storage and retrieval of the extended structure objects, almost all the proposed strategies assume that each object of a picture is abstracted as a minimum bounded rectangle (MBR).) S. K. Chang et al. [4] proposed a pictorial data structure, *2D string*, using symbolic projections to represent symbolic pictures preserving spatial relationships among objects. The basic idea is to project the objects of a picture along the x-axis and y-axis to form two strings representing the relative positions of objects in the x-axis and y-axis, respectively. A picture query can also be specified as a 2D string. Therefore, the problem of pictorial information retrieval then becomes a problem of 2D subsequence matching. However, the representation of 2D strings is not sufficient enough to describe pictures of arbitrary complexity completely, for example, completely or partially overlapping (MBRs of) objects. For this reason, Jungert [29] and S. K. Chang et al. [6] proposed the *2D G-string* representation which introduces more spatial operators and a cutting mechanism to handle more types of spatial relationships among objects in image databases. But a 2D G-string representation scheme is not ideally economic for complex images in terms of storage space efficiency and navigation complexity in spatial reasoning, since each overlapping object is partitioned at the begin-bound or end-bound of the other objects. Therefore, Lee and Hsu [31] proposed a *2D C-string* representation scheme. Since the number of subparts generated by this new cutting mechanism is reduced significantly due to that the cutting lines are performed only at the end-bound points of dominating objects, the lengths of the strings representing pictures are also reduced. However, answering a pictorial query still needs some complicated procedures (which takes $O(N^2)$ time complexity for N nodes) based on the 2D C-string representation. Therefore, Hsu and Lee [26, 27] proposed a *2D C-tree* representation, which can provide a more efficient procedure (which takes $O(N)$ time complexity) for answering a pictorial query.

In the above various 2D string representations, objects may be partitioned into subparts in order to obtain the spatial relations among objects, especially for a complex image with overlapping objects. If there are a large number of subparts, the storage space requirement is high and processing time is long. Therefore, Lee et al. [34] proposed the *2D B-string* representation which preserves all the essential spatial information while at the same time provides indexes for the images without the need of partitioning of any objects.

On the other way, C. C. Chang et al. [11] proposed a new approach of iconic indexing by a *nine direction lower-triangular (9DLT) matrix*. In this strategy, a pictorial query can be processed by using the matrix minus operations; however, only 9 spatial relationships can be handled between any two objects. Later, Y. I. Chang and Yang [15] proposed a *prime-number-based (PN) matrix* strategy, which combines the advantages of the 2D C-string and the 9DLT matrix. In this approach, each spatial operator is represented by a product of some prime numbers, and simple matrix operations and module operations are used to answer a pictorial query. Next, Y. I. Chang et al. [18] proposed a *unique-number-based* strategy in which each spatial operator is represented as a unique number and a range checking operation is applied to answer a pictorial query.

Based upon the variations of 2D strings or the 9DLT matrix, another data structure (for indexing 2D-strings/9DLT matrix), a set of *triples*, to represent the spatial relationship between each pair of objects in a picture, was proposed [12]. For each triple, a hashing value is found and stored. Hence, the problem of image matching becomes a problem of matching hashing value sequences. [2, 12, 14]. Hash oriented algorithms for the similar match retrieval of symbolic images with $O(K)$ search time were also proposed, where K is the number of symbolic images in the database. However, the database grows unwieldy as the space requirement for the index structure is $O(N^2)$ where N is the number of objects in the database.

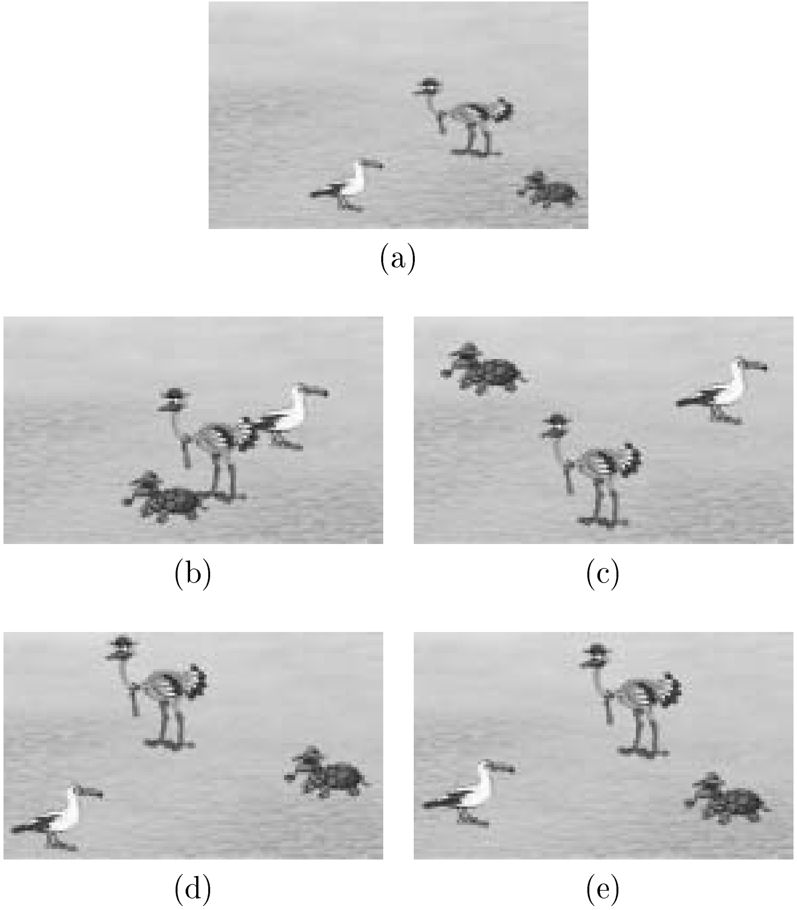
Additionally, 2D-H strings [7] and adaptive 2D-H strings [13, 17] combined the advantages of quad trees and 2D strings. However, 2D strings and 2D-H strings can only represent *directional* relationships. There are some other methods such as the σ-tree [30], the spatial orientation graph [24], the SK-set [28], the 2D-PIR graph [38], the geometry-based $\theta\Re$-string [25], the db-tree [36], and the fuzzy set-based approach [3].

When there are a large number of images in the image database and each image contains many objects, the processing time for image retrievals is tremendous. Actually, the objects or spatial relationships among objects in a symbolic picture can be treated as attributes or keywords of a document. Thus, a signature file can speed up spatial match retrieval [21], since the comparison of each signature (for a picture) is a O(1) bitwise ANDed operation, as compared to the sub-string matching (or matrix comparison) operations for an exactly match. That is, the signature can act as a searching filter to prune (i.e., filter out) most of the unsatisfactory images. Only the records which match the signature need to be examined further to test for exact query matches. Since a signature is a binary codeword associated with each record of the image database, the retrieval of images based on the simple bit conjunction operations can be speeded up tremendously at a very small cost of space overhead. Therefore, to handle large amounts of image databases, several access strategies [9, 10, 16, 20, 32, 33] have been proposed for and two-level signature files [39].

For example, Lee et al. [32] proposed an access strategy for retrieval by subpictures that are represented in 2D strings. Lee et al.'s strategy considers only three spatial operators along the x-axis or the y-axis; it is too rough. To reduce the rate of a false match and to reduce the number of needed record signatures, where a false match is that a record signature matches a query signature but the corresponding record does not satisfy the query, C. C. Chang et al. [9] proposed a strategy with pictures represented in 9DLT matrices. In C. C. Chang et al.'s strategy, each record signature is represented by 9 bits and 9 bit strings of size N, where $N > 0$. Then, C. C. Chang et al. [10] proposed a *module-oriented signature extraction* strategy, in which prime numbers are used to compose the signatures, and the module operation will be applied when the query happens which further reduces the rate of a false match.

The above access strategies consider only 9 spatial relationships. Later, Y. I. Chang and Yang [16] proposed two efficient access strategies with pictures represented in a *spatial matrix* that was described in the PN matrix strategy [15] for image databases, which can handle 169 spatial relationships. In Y. I. Chang and Yang's strategy, each record signature is represented by 26 bits and 26 bit strings of size N, where $N > 0$. In Y. I. Chang and Yang's second strategy, each record signature is represented by 26 bits and 26 products of prime numbers. Given the same image databases, the same query picture and the same hash functions, the rate of a false match in Y. I. Chang and Yang's both strategies can be smaller than that of the previous approaches. On the other way, Lee et al. [33] proposed an integrated signature file structure based on 2D B-strings to handle the retrieval by objects and retrieval by binary spatial relationship to different extent of similarity (type-i, $i = 0$, 1, 2). Figure 1 shows some examples of different extent of similarity. As compared to Figure 1-(a), Figure 1-(b) only contains the same objects, Figure 1-(c) contains the same objects and have the same spatial category (the *disjoin* category) between objects, which is referred as *type-0* similarity. Moreover, as compared to Figure 1-(a), Figure 1-(d) satisfies *type-0* similarity and has the same orthogonal relations, which is referred as *type-1* similarity; while Figure 1-(e) satisfies *type-1* similarity and has the same spatial relationships in x-axis and y-axis, which is referred as *type-2* similarity [33]. (Note that the difference between Figure 1-(d) and Figure 1-(e) is that the turtle immediately follows, or *meets*, the ostrich in x-axis in Figure 1-(e); while it is not in Figure 1-(d).) Hence, there are in total four kinds of signature files of 2D B-strings to be generated.

Although Lee et al.'s signature file approach [33] can answer queries of objects and type-i similarity efficiently in terms of speed, their signature files containing 4 kinds of signatures occupy a large storage space. On the other hand, as shown in [16], Y. I. Chang



(a)

(b)

(c)

(d)

(e)

Figure 1: Types of Similarity: (a) the original picture; (b) object similarity; (c) type-0 similarity; (d) type-1 similarity; (e) type-2 similarity.

and Yang's two access strategies are efficient enough and have a low false match rate as compared to the previous approaches [9, 32]. However, the signatures constructed from Y. I. Chang and Yang's two strategies can only answer whether there exist pictures of type-2 similarity with the query picture. Let's call these signatures, the *type-2* signatures. In this paper, we first present the revised version of Y. I. Chang and Yang's bit-string-based access strategy [16], which constructs type-2 signatures. Based on the revised version, we then propose algorithms to dynamically convert those type-2 signatures into *object* signatures, *type-0* signatures and *type-1* signatures at run time so that we can answer queries of object similarity, type-0 similarity, type-1 similarity and type-2 similarity by storing only one type of signatures, i.e., the type-2 signatures. While in [33], Lee et al. have to construct 4 types of signatures in the database to achieve the same goal. From our simulation, we show that our approach can provide a higher rate of a correct match (= 1 - the rate of a false match) and needs a smaller storage requirement than Lee et al.'s approach.

The rest of this paper is organized as follows. In Section 2, we give some definitions used in our paper. In Section 3, we will present a revised version of the bit-string-signature-based access strategy, and the corresponding type-2 signatures. In Sections 4, 5 and 6, we show how to convert such type-2 signatures into object signatures, type-0 signatures and type-1 signatures, respectively. In Section 7, we give a comparison between our approach and the signature file approach based on 2D B-strings. Finally, Section 8 gives a conclusion.

2 Background

In this Section, we describe the definitions of type-i similarity for *2D strings* (which work well for non-overlapping objects) and *2D B-strings* (which work well for overlapping objects).

2.1 Type-i Similarity for 2D Strings

The *2D string* representation was proposed by S. K. Chang et al. [4]. In this approach, let V be a set of symbols, where each symbol could represent a pictorial object or a pixel. Let A be the set $\{=, <, :\}$, where $=$, $<$ and $:$ are three special symbols not in V. For example, consider the picture shown in Figure 2, $V = \{a, b, c, d, e, f\}$. The 2D string representing the above picture f is as follows:

$(a = d < e : f = b < c, a = e : f < b = c < d)$,

where the symbol $<$ denotes the left-right or below-above spatial relationship. The symbol $=$ denotes the "at the same spatial location as" relationship and the symbol $:$ denotes the "in the same set as" relation. The corresponding *reduced 2D string* is as follows:

$(ad < efb < c, aef < bc < d)$.

The *rank* of each symbol in a string x, which is defined to be one plus the number "$<$" preceding this symbol in x, plays an important role in 2D string matching. Let the rank of symbol b be denoted by $r(b)$. The ranks for each symbol in the string "$ad < efb < c$" are 1, 1, 2, 2, 2 and 3, respectively.

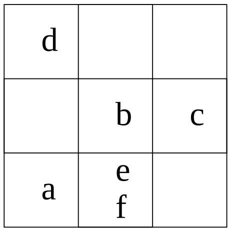


Figure 2: A picture f

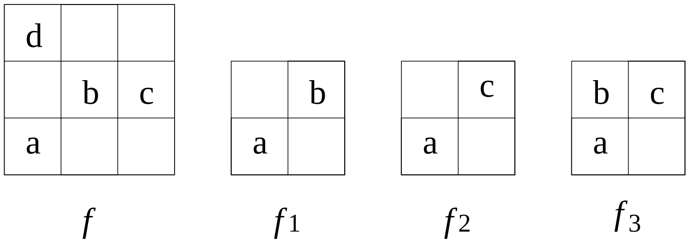


Figure 3: Picture matching examples for 2D strings

A string x is a type-i 1D subsequence of string y if

(1) x is contained in a string y,

(2) if $a_1 w_1 b_1$ is a substring of x, a_1 matches a_2 in y and b_1 matches b_2 in y, then

(type-0) $r(b_2) - r(a_2) \geq r(b_1) - r(a_1)$ or $r(b_1) - r(a_1) = 0$.

(type-1) $r(b_2) - r(a_2) \geq r(b_1) - r(a_1) > 0$ or $r(b_2) - r(a_2) = r(b_1) - r(a_1) = 0$.

(type-2) $r(b_2) - r(a_2) = r(b_1) - r(a_1)$.

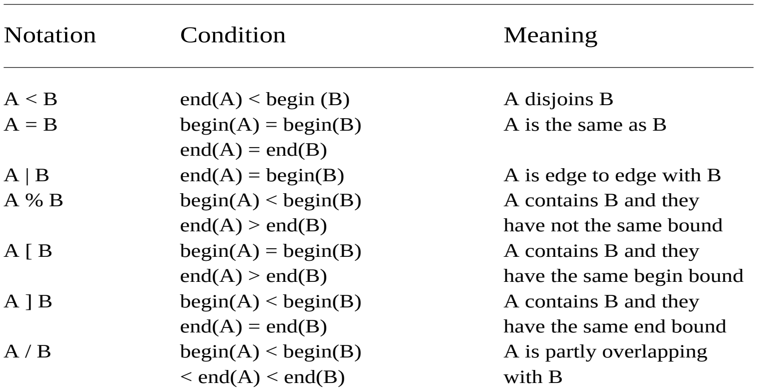
Let (x, y) and (x', y') be the 2D string representation of f and f', respectively. (x', y') is a type-i subsequence of (x, y) if

(1) x' is type-i 1D subsequence of x, and
(2) y' is type-i 1D subsequence of y.

In this case, we say f' is a type-i sub-picture of f. Therefore, the picture matching problem thus becomes a 2D string matching problem. In Figure 3, The 2D string representations for f, f_1, f_2 and f_3 are all type-0 sub-pictures of f; f_1 and f_2 are type-1 sub-pictures of f; only f_1 is type-2 sub-picture of f.

2.2 Type-i Similarity for 2D B-Strings

The 2D C-string is proposed by Lee and Hsu [31], which can represent overlapping objects. Table 1 shows the formal definition of the set of spatial operators defined in the 2D C-string representation, where the notation "begin(A)" denotes the value of begin-bound of object A and "end(A)" denotes the value of end-bound of object A. According to the begin-bound and end-bound of the picture objects, spatial relationships between two enclosing rectangles



Notation	Condition	Meaning
A < B	end(A) < begin (B)	A disjoins B

A = B	begin(A) = begin(B)
	end(A) = end(B)
		A is the same as B

A | B	end(A) = begin(B)	A is edge to edge with B

A % B	begin(A) < begin(B)
	end(A) > end(B)
		A contains B and they
		have not the same bound

A [B	begin(A) = begin(B)
	end(A) > end(B)
		A contains B and they
		have the same begin bound

A] B	begin(A) < begin(B)
	end(A) = end(B)
		A contains B and they
		have the same end bound

A / B	begin(A) < begin(B) < end(A) < end(B)	A is partly overlapping with B

Table 1: Definitions of Lee et al.'s spatial operators

can be categorized into 13 types ignoring their length along the x-(or y-) axis. Therefore, there are 169 types of spatial relationships between two rectangles in 2D space as shown in Figure 4, where operator* means the inverse operator of the related operator. They can be categorized in five types, *disjoin*, *join*, *contain*, *belong* and *partial overlap*. The five types of the spatial relations between objects are defined in Figure 5. The measure criteria for categorization is the area of the intersection of A and B. For the picture f_1 shown in Figure 6, the corresponding 2D C-string representation is as follows:

x-string: A|B%C,
y-string: B]A]C|A[C,

where a cutting occurs at the end bound of symbol B along the y-axis.

For those 169 spatial relationships organized by the 2D C-string representation, Lee et al. proposed another iconic indexing, *2D B-string* [34]. For the picture f_1 shown in Figure 6, the corresponding 2D B-string representation is as follows:

x-string: AA=BCCB,
y-string: BACBCA.

The spatial relationships are derived by using the *ranks* of symbols in the *2D B-string.* The *rank* of each symbol in a string x, which is defined as the position of this symbol minus the number of '=' preceding this symbol in x string. For example, the ranks for each symbol in the string "AA=BCCB" are 1 (denoted as rank-begin(A)), 2 (denoted as rank-end(A)), 2, 3, 4, and 5, respectively. In the 2D B-string representation, three types of similarity measures are defined as follows: (1) C_{AB} denotes the category type between A and B; (2) O_{AB} denotes the orthogonal relation between A and B, including *north*, *south*, *west* and *east* as defined formally in Figure 7; (3) R_{AB} denotes one of the 169 spatial relationships in two dimensional space between A and B. Note that the above C_{AB}, O_{AB} and R_{AB} relationships can be derived from the ranks of symbols A and B. Based on the three types of similarity measures, the type-i subpicture matching can be defined as shown in Figure 8 [34]. For example, in Figure 6, f_1, f_2 and f_3 are all type-0 sub-pictures of f; f_1 and f_2 are type-1 sub-pictures of f; only f_1 is type-2 sub-picture of f.

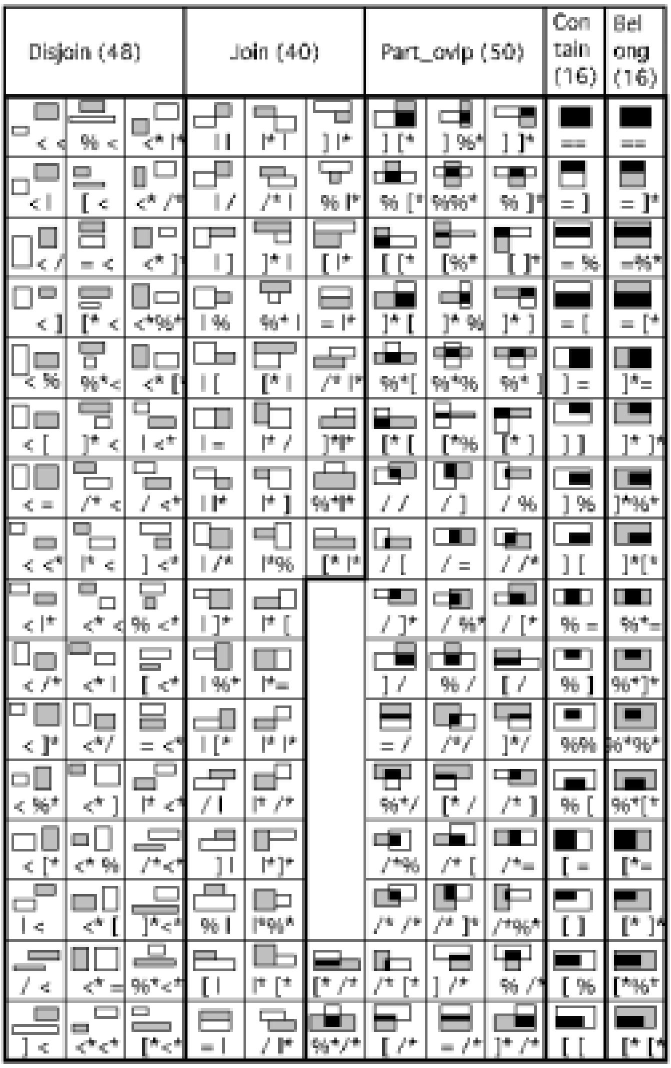


Figure 4: The 169 spatial relationship (R) types of two objects

Disjoin: $A \cap B = \emptyset$
Join: $A \cap B =$ single point or line segment
Contain: $A \cap B = B$
Belong: $A \cap B = A$
Partial overlap: $A \cap B =$ the area of partial A and partial B.

Figure 5: The category (C) rules defined by intersection

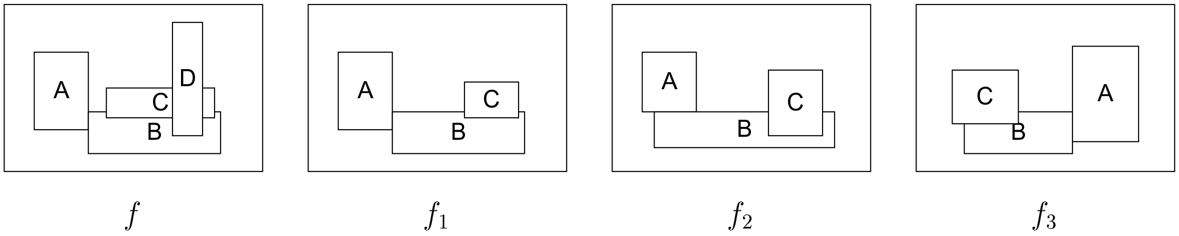


Figure 6: Picture matching examples for 2D B-strings

Rule 1. A is in the east of B iff x-axis: rank-end(A) $>$ rank-end(B).
Rule 2. A is in the west of B iff x-axis: rank-begin(A) $<$ rank-begin(B).
Rule 3. A is in the north of B iff y-axis: rank-end(A) $>$ rank-end(B).
Rule 4. A is in the south of B iff y-axis: rank-begin(A) $<$ rank-begin(B).

Figure 7: The orthogonal relation (O) rules

Picture f' is a type-i subpicture of f if:
(1) all objects in f' are also in f;
(2) for any two objects, A and B:
(type-0) $C'_{AB} = C_{AB}$
(type-1) $C'_{AB} = C_{AB}$ and $O'_{AB} = O_{AB}$
(type-2) $C'_{AB} = C_{AB}$ and $O'_{AB} = O_{AB}$ and $R'_{AB} = R_{AB}$

Figure 8: Three types of similarity (for 169 spatial relationships)

3 A Revised Version of the Bit-String-Signature-Based Access Method

In this section, we present a revised version of Y. I. Chang and Yang's bit-string-signature-based access strategy [16]. The main difference between the revised version and the original one [16] is the assignment of a unique value to each of spatial operators. Based on the new assignments of values to spatial operators, we can construct object, type-0 and type-1 signatures from a given type-2 signature efficiently. (Note that the revised version of the prime-number-based access strategy and related algorithms are shown in Appendix A.)

3.1 A Spatial Matrix

Suppose a picture f contains m objects and let $V = \{v_1, v_2, ..., v_m\}$. Let W be the set of 13 spatial operators $\{ <, <^*, |, |^*, [, [^*,],]^*, \%, \%^*, /, /^*, = \}$ defined in 2D C-string [31], where operator* means the inverse operator of the related operator. A $m \times m$ *spatial matrix* S of picture f is defined as follows:

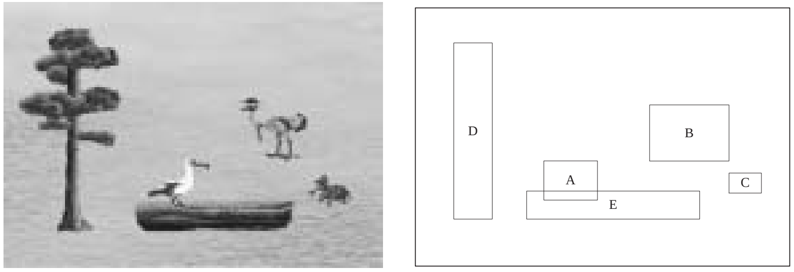
$$S = \begin{array}{cc} & \begin{array}{ccccc} v_1 & v_2 & \cdots & v_{m-1} & v_m \end{array} \\ \begin{array}{c} v_1 \\ v_2 \\ \vdots \\ v_{m-1} \\ v_m \end{array} & \begin{bmatrix} 0 & r^y_{1,2} & \cdots & \cdots & r^y_{1,m} \\ r^x_{1,2} & 0 & \ddots & & \vdots \\ \vdots & \ddots & 0 & \ddots & \vdots \\ \vdots & & \ddots & 0 & r^y_{m-1,m} \\ r^x_{1,m} & \cdots & \cdots & r^x_{m-1,m} & 0 \end{bmatrix} \end{array}$$

where the lower triangular matrix stores the spatial information along the x-axis, and the upper triangular matrix stores the spatial information along the y-axis. That is, $S[v_i, v_j] = r^x_{j,i}$ if $i > j$; $S[v_i, v_j] = r^y_{i,j}$ if $i < j$; $S[v_i, v_j] = 0$ if $i = j$, $\forall v_i, v_j \in V$, $\forall r^x_{j,i}, r^y_{i,j} \in W$, $1 \leq i, j \leq m$, where $r^x_{j,i}$ is the spatial operator between objects v_i and v_j along the x-axis and $r^y_{i,j}$ is the spatial operator between objects v_i and v_j along the y-axis. Note that in this representation, we always record the relationships between two objects v_i and v_j from the view point of object v_i no matter along the x-axis or the y-axis, where $i < j$. That is why $S[v_i, v_j] = r^x_{ji}$ when $i > j$.

For the picture shown in Figure 9, the corresponding spatial matrix S is shown as follows:

$$S = \begin{array}{cc} & \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{array}{c} A \\ B \\ C \\ D \\ E \end{array} & \begin{bmatrix} 0 & | & \% & \%^* & /^* \\ < & 0 & <^* & \%^* & <^* \\ < & | & 0 & \%^* & /^* \\ <^* & <^* & <^* & 0 & [\\ \%^* & /^* & <^* & < & 0 \end{bmatrix} \end{array}$$

Following the idea similar to [18], we let 1, 2, 3, 4, 8, 12, 7, 11, 9, 13, 5, 6 and 10 denote the unique identifier (*uid*) of spatial operators $<$, $<^*$, $|$, $|^*$, $[$, $[^*$, $]$, $]^*$, $\%$, $\%^*$, $/$, $/^*$ and $=$,



(a) (b)

Figure 9: An image and its symbolic representation

operator	$<$	$<^*$	$\vert$	$\vert^*$	$/$	$/^*$	$]$	$[$	$\%$	$=$	$]^*$	$[^*$	$\%^*$
Proposed	1	2	3	4	5	6	7	8	9	10	11	12	13
Y. I. Chang and Yang [16]	1	2	3	4	11	12	7	5	9	13	8	6	10

Figure 10: Uids of 13 spatial operators

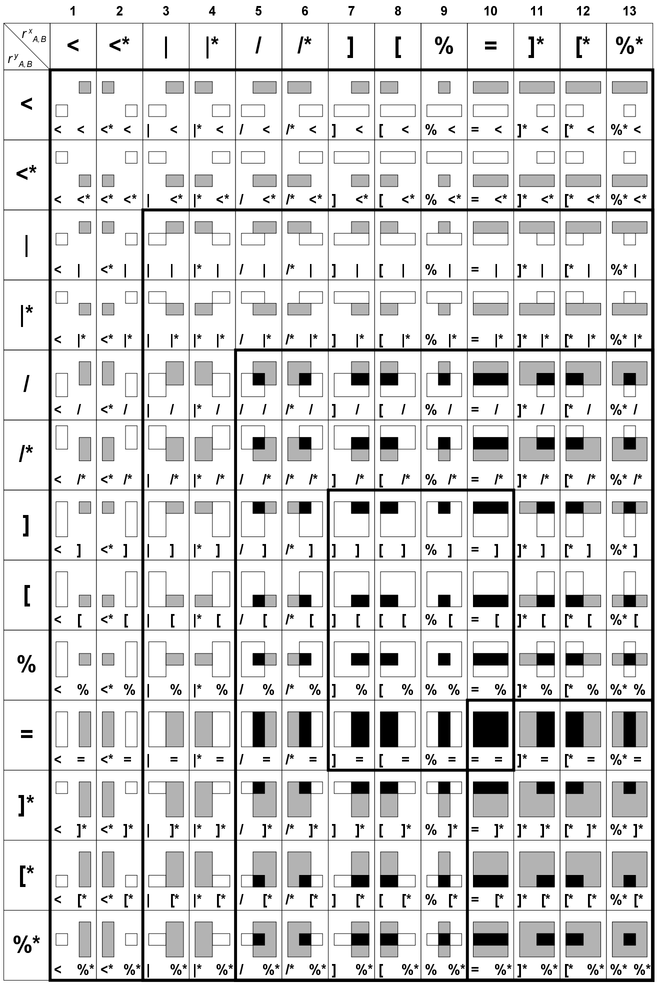
respectively. Figure 10 shows the relationship between our *uids* and Y. I. Chang and Yang's assignments [16] for spatial operators. (Note that in [16], they use 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 and 13 to denote the spatial operators $<$, $<^*$, $|$, $|^*$, $[$, $[^*$, $]$, $]^*$, $\%$, $\%^*$, $/$,$/^*$ and $=$, respectively.) Based on the new assignment of a unique identifier(*uid*) to each of 13 spatial operators as shown in Figure 10, we can rearrange the total 169 spatial relationships defined in the 2D C-string strategy [31], in Table 2. Under this new assignment, a *Category* table is formed such that relationships of the same category are grouped together as shown in Table 3. In this way, the processing of category classification becomes a range checking operation, which can simplify the processing of the decision of the spatial category relationships (as described in Section 5).

Next, the corresponding *reduced spatial matrix* (RSM) is as follows:

$$RSM = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{bmatrix} 0 & 3 & 9 & 13 & 6 \\ 1 & 0 & 2 & 13 & 2 \\ 1 & 3 & 0 & 13 & 6 \\ 2 & 2 & 2 & 0 & 8 \\ 13 & 6 & 2 & 1 & 0 \end{bmatrix} \end{array}$$

Moreover, a spatial x-string set T^x is defined as $\{v_j v_i r_{ij}^x | 1 \leq j < i \leq m, r_{ji}^x \in \{1, 2, \cdots, 12, 13\}\}$, and a spatial y-string set T^y is defined as $\{v_i v_j r_{ij}^y | 1 \leq i < j \leq m, r_{ij}^y \in \{1, 2, \cdots, 12, 13\}\}$. Therefore, the corresponding spatial x-string set for the above RSM is

T^x = { AB1, AC1, AD2, AE13, BC3, BD2, BE6, CD2, CE2, DE1 },



		1	2	3	4	5	6	7	8	9	10	11	12	13
	$r^x_{A,B}$ $r^y_{A,B}$	<	<*	|	|*	/	/*	]	[	%	=	]*	[*	%*
1	<	< <	<* <	| <	|* <	/ <	/* <	] <	[<	% <	= <	]* <	[* <	%* <
2	<*	< <*	<* <*	| <*	|* <*	/ <*	/* <*	] <*	[<*	% <*	= <*	]* <*	[* <*	%* <*
3	|	< |	<* |	| |	|* |	/ |	/* |	] |	[|	% |	= |	]* |	[* |	%* |
4	|*	< |*	<* |*	| |*	|* |*	/ |*	/* |*	] |*	[|*	% |*	= |*	]* |*	[* |*	%* |*
5	/	< /	<* /	| /	|* /	/ /	/* /	] /	[/	% /	= /	]* /	[* /	%* /
6	/*	< /*	<* /*	| /*	|* /*	/ /*	/* /*	] /*	[/*	% /*	= /*	]* /*	[* /*	%* /*
7	]	<]	<*]	|]	|*]	/]	/*]	]]	[ ]	%]	=]	]*]	[*]	%*]
8	[	< [	<* [	| [	|* [	/ [	/* [	] [	[[	% [	= [	]* [	[* [	%* [
9	%	< %	<* %	| %	|* %	/ %	/* %	] %	[%	% %	= %	]* %	[* %	%* %
10	=	< =	<* =	| =	|* =	/ =	/* =	] =	[=	% =	= =	]* =	[* =	%* =
11	]*	<]*	<*]*	|]*	|*]*	/]*	/*]*	]]*	[ ]*	%]*	=]*	]*]*	[*]*	%*]*
12	[*	< [*	<* [*	| [*	|* [*	/ [*	/* [*	] [*	[[*	% [*	= [*	]* [*	[* [*	%* [*
13	%*	< %*	<* %*	| %*	|* %*	/ %*	/* %*	] %*	[%*	% %*	= %*	]* %*	[* %*	%* %*

Table 2: A new organization of 169 spatial relationships

$uid^{x}_{A,B}$ $uid^{y}_{A,B}$	1	2	3	4	5	6	7	8	9	10	11	12	13
1	D	D	D	D	D	D	D	D	D	D	D	D	D
2	D	D	D	D	D	D	D	D	D	D	D	D	D
3	D	D	J	J	J	J	J	J	J	J	J	J	J
4	D	D	J	J	J	J	J	J	J	J	J	J	J
5	D	D	J	J	P	P	P	P	P	P	P	P	P
6	D	D	J	J	P	P	P	P	P	P	P	P	P
7	D	D	J	J	P	P	C	C	C	C	P	P	P
8	D	D	J	J	P	P	C	C	C	C	P	P	P
9	D	D	J	J	P	P	C	C	C	C	P	P	P
10	D	D	J	J	P	P	C	C	C	B,C	B	B	B
11	D	D	J	J	P	P	P	P	P	B	B	B	B
12	D	D	J	J	P	P	P	P	P	B	B	B	B
13	D	D	J	J	P	P	P	P	P	B	B	B	B

Table 3: The category table

and the corresponding spatial y-string set for the above RSM is

T^y = { AB3, AC9, AD13, AE6, BC2, BD13, BE2, CD13, CE6, DE8 }.

3.2 Type-2 Record Signatures

Based on the 13 spatial operators, we now define a Record Signature (RS). A RS consists of two parts, RS^1 and RS^2 as shown in Figure 11. RS^1 contains 2 segments RS^{1x} and RS^{1y}, which represent the record signature flags from the view point of x-axis and y-axis, respectively, and each segment is a 13-bit string. These two 13-bit strings are used to indicate the existence or absence of those 13 spatial operators along the x-axis and y-axis, respectively. RS^2 consists of two segments, RS^{2x} and RS^{2y}. Each of these two segments contains 13 bit strings. The i-th bit string (or signature) among those 13 bit strings is used to record the union of the signatures of those pairs of objects which have the same i-th spatial operator. We use $RS^{2x}(i)$ to represent the i-th segment of RS^{2x}. That is, $RS^{2x}(i)$ = $\cup$ $\theta_r(XYi)$, X, Y $\in$ V, $i \in$ { 1, 2, $\cdots$, 12, 13 }, where θ_r is the hash function of the record signature. The algorithm for efficient data access of image databases is described as follows.

Algorithm (Record Signature)

(Step 1) *According to the reduced spatial matrix, list all the spatial x-string sets T^x and spatial y-string sets T^y.*

(Step 2) *Design the function θ_r according to the given k_r and b_r, which maps each pair of*

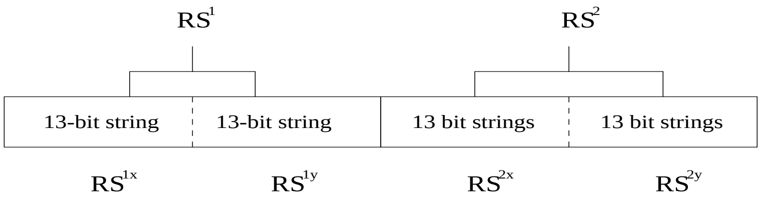


Figure 11: The components of a Record Signature

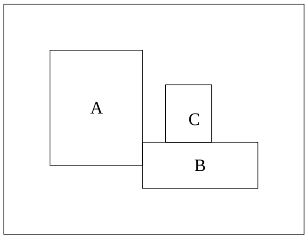


Figure 12: An example

symbols into a unique bit string, where k_r denotes the number of 1's in a record signature and b_r denotes the number of bits in a record signature.

(Step 3) *Set all bits in RS to 0.*

(Step 4) *For each spatial x-string ABi in T^x, we let the i-th bit of RS^{1x} be 1, and then perform $RS^{2x}(i) = RS^{2x}(i) \cup \theta_r(AB)$.*

(Step 5) *Repeat Step 4 by replacing T^x with T^y.*

(Step 6) *Compress RS^2 by removing useless bit strings, resulting in a reduced form of a record signature. If the i-th bit of RS^{1x} (or RS^{1y}) is 0, then remove $RS^{2x}(i)$ (or $RS^{2y}(i)$).*

To illustrate the algorithm, let's see the following example. For the figure shown in Figure 12, first, we construct the spatial matrix and the reduced spatial matrix (RSM).

$$S = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \left[\begin{array}{ccc} 0 & /^* & \% \\ | & 0 & | \\ < & \% & 0 \end{array}\right] \end{array} \qquad RSM = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \left[\begin{array}{ccc} 0 & 6 & 9 \\ 3 & 0 & 3 \\ 1 & 9 & 0 \end{array}\right] \end{array}$$

Applying the algorithm, we can construct the Record Signature of the picture as follows.

(1) Generate the spatial x-string set T^x and spatial y-string set T^y.
$T^x = \{$ AB3, AC1, BC9 $\}$.
$T^y = \{$ AB6, AC9, BC3 $\}$.

(2) Design the function θ_r (where $b_r = 5$, $k_r = 2$) which maps each pair of symbols to a unique bit string.

spatial string	θ_r
AB	10001
AC	10100
BC	01100

(3) Set all bits in RS to 0.
$RS = RS^1 + RS^2$
= 0000000000000 0000000000000 00000 00000 00000 00000 00000 00000
00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000
00000 00000 00000 00000 00000 00000 00000 00000 00000.

(4) If $ABi \in T^x$, we let the i-th bit of RS^{1x} be 1, and then perform $RS^{2x}(i) = RS^{2x}(i) \cup \theta_r(AB)$.
$RS^{1x} = 1010000010000$.
$RS^{2x}(1) = RS^{2x}(1) \cup \theta_r(AC) = 10100$.
$RS^{2x}(3) = RS^{2x}(3) \cup \theta_r(AB) = 10001$.
$RS^{2x}(9) = RS^{2x}(9) \cup \theta_r(BC) = 01100$.

(5) Repeat Step 4 by replacing T^x by T^y. We have
$RS^{2y}(3) = RS^{2y}(3) \cup \theta_r(BC) = 01100$.
$RS^{2y}(6) = RS^{2y}(6) \cup \theta_r(AB) = 10001$.
$RS^{2y}(9) = RS^{2y}(9) \cup \theta_r(AC) = 10100$.

(6) Compress RS^2 by removing useless bit strings. The resulting reduced from the record signature is as follows:
RS = 1010000010000 001001001000 10100 10001 01100 01100 10001 10100.

4 Object Signatures Based on Bit-Strings

To simplify the presentation of our algorithm, we convert each signature back to its *complete* form, instead of the *reduced* form, when a query involving objects, type-0 and type-1 occurs. For example, the signature

0000000000001 1000000000000 10001 10001
means that only the 13-th spatial relationship in x-axis and first spatial relationship in y-axis occur. While the complete form for the same picture is

0000000000001 1000000000000
00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 10001
10001 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000 00000.

Given a type-2 record signature in a complete form, we now present the algorithm to convert such a type-2 record signature into the related object record signature (ORS) as follows.

Algorithm Object
(Convert a Type-2 Record Signature into an Object Record Signature)

(Step 1) *Let the numbers of bits which are set to 1 in* RS^{1x} *and* RS^{1y} *to* C^x *and* C^y*, respectively. If* $C^x \leq C^y$*, let* $TRS^1 = RS^{1x}$ *and* $TRS^2 = RS^{2x}$*; otherwise, let* $TRS^1 = RS^{1y}$ *and* $TRS^2 = RS^{2y}$*.*

(Step 2) *Set every bit in ORS to 0.*

(Step 3) *For* $i = 1$ *to 13 do*
If $TRS^1(i) = 1$ *then* $ORS = ORS \cup TRS^2(i)$*, where* $TRS^1(i)$ *means the i-th bit of* TRS^1 *and* $TRS^2(i)$ *means the i-th bit string of* TRS^2*.*

(Note that the purpose of Step 1 is to reduce the burden in Step 3; that is, we try to choose the one (x-axis or y-axis) which has a small number of non-zero bit-strings.)

To illustrate the algorithm, let's see the following example. Suppose there are four pictures in the database, P_1, P_2, P_3 and P_4 as shown in Figure 13, and the hash function θ_r (where $b_r = 5$, $k_r = 2$) is defined as follows.

spatial string	θ_r
AB	10001
AC	10100
BC	01100
AD	10010
BD	01010
CD	00110

The corresponding reduced spatial matrices are as follows.

$$RSM_1 = \begin{array}{c} \\ A \\ B \\ D \end{array} \begin{array}{c} \begin{array}{ccc} A & B & D \end{array} \\ \begin{bmatrix} 0 & 6 & 8 \\ 3 & 0 & 5 \\ 1 & 9 & 0 \end{bmatrix} \end{array} \qquad RSM_2 = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} 0 & 6 & 9 \\ 3 & 0 & 9 \\ 5 & 6 & 0 \end{bmatrix} \end{array}$$

$$RSM_3 = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} 0 & 3 & 1 \\ 5 & 0 & 5 \\ 6 & 6 & 0 \end{bmatrix} \end{array} \qquad RSM_4 = \begin{array}{c} \\ A \\ B \\ C \end{array} \begin{array}{c} \begin{array}{ccc} A & B & C \end{array} \\ \begin{bmatrix} 0 & 4 & 6 \\ 5 & 0 & 5 \\ 1 & 9 & 0 \end{bmatrix} \end{array}$$

Based on the revised version of Y. I. Chang and Yang's algorithm, the corresponding complete form of type-2 signatures for those pictures are as follows:

RS_1 = 1010000010000 0000110100000
10010 00000 10001 00000 00000 00000 00000 00000 01010 00000 00000 00000 00000
00000 00000 00000 00000 01010 10001 00000 10010 00000 00000 00000 00000 00000,

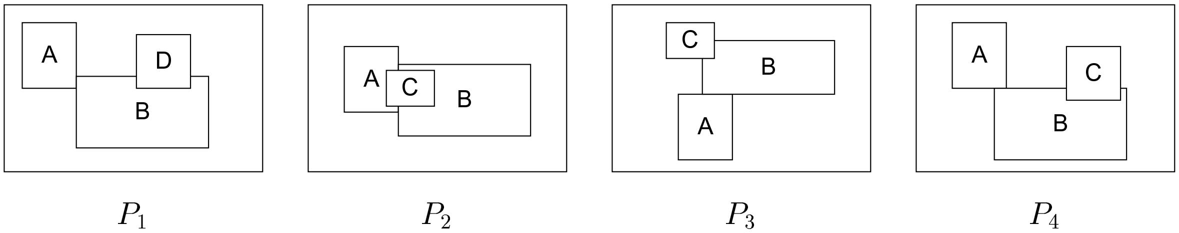


Figure 13: A 4-picture image database

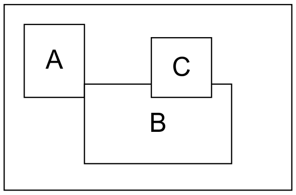


Figure 14: A query picture q_1

RS_2 = 0010110000000 0000010010000
00000 00000 10001 00000 10100 01100 00000 00000 00000 00000 00000 00000 00000
00000 00000 00000 00000 00000 10001 00000 00000 11100 00000 00000 00000 00000,
RS_3 = 0000110000000 1010100000000
00000 00000 00000 00000 10001 11100 00000 00000 00000 00000 00000 00000 00000
10100 00000 10001 00000 01100 00000 00000 00000 00000 00000 00000 00000 00000,
RS_4 = 1000100010000 0001110000000
10100 00000 00000 00000 10001 00000 00000 00000 01100 00000 00000 00000 00000
00000 00000 00000 10001 01100 10100 00000 00000 00000 00000 00000 00000 00000.

Then, let's see how to convert the type-2 signatures into the object signatures. Take picture P_1 as an example.

$ORS_1 = 10010 \cup 10001 \cup 01010 = 11011.$

(Note that in this example, we choose bit-strings in x-axis.)
In the same way, the resulting object signatures for P_2, P_3 and P_4 are as follows:

$ORS_2 = 10001 \cup 11100 = 11101,$
$ORS_3 = 10001 \cup 11100 = 11101,$
$ORS_4 = 10100 \cup 10001 \cup 01100 = 11101.$

Given a query picture q_1 as shown in Figure 14, the corresponding complete form of the type-2 query signature is as follows:

QRS = 1010000010000 0000110100000

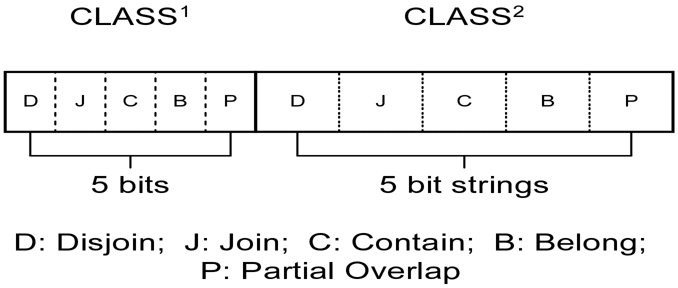


Figure 15: Type-0 signature $CLASS$

10100 00000 10001 00000 00000 00000 00000 00000 01100 00000 00000 00000 00000
00000 00000 00000 00000 01100 10001 00000 10100 00000 00000 00000 00000 00000.

The corresponding query object record signature ($QORS$) is as follows,

$QORS = 10100 \cup 10001 \cup 01100 = 11101$. Next, since $QORS \cap ORS_1 \neq QORS$,

$QORS \cap ORS_2 = QORS$, $QORS \cap ORS_3 = QORS$ and $QORS \cap ORS_4 = QORS$, we conclude that pictures P_2, P_3 and P_4 may have the same objects with the query picture q_1 while picture P_1 has some objects different from the query picture q_1.

5 Type-0 Signatures Based on Bit-Strings

Figure 15 shows the structure of the type-0 signature, $CLASS$. Given a type-2 record signature in a complete from, we now present the algorithm to convert such a type-2 record signature into the related type-0 record signature ($CLASS$) based on the observation from Table 3 as follows.

Algorithm Type-0
(Convert a Type-2 Record Signature into a Type-0 Record Signature)

```
(Step 1) Set every bit in $CLASS$ to 0.

(Step 2) $CLASS^1(1) = \bigcup_{i=1}^{2} RS^{1x}(i) \cup \bigcup_{i=1}^{2} RS^{1y}(i)$        /* Disjoin */
         if $CLASS^1(1) = 1$ then
             $CLASS^2(1) = \bigcup_{i=1}^{2} RS^{2x}(i) \cup \bigcup_{i=1}^{2} RS^{2y}(i)$

(Step 3) $CLASS^1(2) = (\bigcup_{i=3}^{4} RS^{1x}(i) \cap \bigcup_{i=3}^{13} RS^{1y}(i)) \cup$        /* Join */
                       $(\bigcup_{i=5}^{13} RS^{1x}(i) \cap \bigcup_{i=3}^{4} RS^{1y}(i))$
```

```
          if $CLASS^1(2) = 1$ then
              $CLASS^2(2) = (\bigcup_{i=3}^{4} RS^{2x}(i) \cap \bigcup_{i=3}^{13} RS^{2y}(i)) \cup$
                            $(\bigcup_{i=5}^{13} RS^{2x}(i) \cap \bigcup_{i=3}^{4} RS^{2y}(i))$

(Step 4)  $CLASS^1(3) = \bigcup_{i=7}^{10} RS^{1x}(i) \cap \bigcup_{i=7}^{10} RS^{1y}(i)$          /* Contain */
          if $CLASS^1(3) = 1$ then
              $CLASS^2(3) = \bigcup_{i=7}^{10} RS^{2x}(i) \cap \bigcup_{i=7}^{10} RS^{2y}(i)$

(Step 5)  $CLASS^1(4) = \bigcup_{i=10}^{13} RS^{1x}(i) \cap \bigcup_{i=10}^{13} RS^{1y}(i)$          /* Belong */
          if $CLASS^1(4) = 1$ then
              $CLASS^2(4) = \bigcup_{i=10}^{13} RS^{2x}(i) \cap \bigcup_{i=10}^{13} RS^{2y}(i)$

(Step 6)  $CLASS^1(5) = (\bigcup_{i=5}^{6} RS^{1x}(i) \cap \bigcup_{i=5}^{13} RS^{1y}(i)) \cup$          /* Partial overlap */
                        $(\bigcup_{i=7}^{13} RS^{1x}(i) \cap \bigcup_{i=5}^{6} RS^{1y}(i)) \cup$
                        $(\bigcup_{i=7}^{9} RS^{1x}(i) \cap \bigcup_{i=11}^{13} RS^{1y}(i)) \cup$
                        $(\bigcup_{i=11}^{13} RS^{1x}(i) \cap \bigcup_{i=7}^{9} RS^{1y}(i))$
          if $CLASS^1(5) = 1$ then
              $CLASS^2(5) = (\bigcup_{i=5}^{6} RS^{2x}(i) \cap \bigcup_{i=5}^{13} RS^{2y}(i)) \cup$
                            $(\bigcup_{i=7}^{13} RS^{2x}(i) \cap \bigcup_{i=5}^{6} RS^{2y}(i)) \cup$
                            $(\bigcup_{i=7}^{9} RS^{2x}(i) \cap \bigcup_{i=11}^{13} RS^{2y}(i)) \cup$
                            $(\bigcup_{i=11}^{13} RS^{2x}(i) \cap \bigcup_{i=7}^{9} RS^{2y}(i))$
```

Take picture P_1 shown in Figure 13 as an example, we now show how to convert the type-2 signature into the corresponding type-0 signature.

$$
\begin{aligned}
CLASS_1^1(1) &= RS_1^{1x}(1) = 1 \\
CLASS_1^2(1) &= RS_1^{2x}(1) = 10010 \\
\\
CLASS_1^1(2) &= RS_1^{1x}(3) \cap (RS_1^{1y}(5) \cup RS_1^{1y}(6) \cup RS_1^{1y}(8)) = 1 \\
CLASS_1^2(2) &= RS_1^{2x}(3) \cap (RS_1^{2y}(5) \cup RS_1^{2y}(6) \cup RS_1^{2y}(8)) \\
&= 10001 \cap (01010 \cup 10001 \cup 10010) = 10001 \\
\\
CLASS_1^1(3) &= 0 \\
CLASS_1^2(3) &= 00000 \\
\\
CLASS_1^1(4) &= 0 \\
CLASS_1^2(4) &= 00000 \\
\\
CLASS_1^1(5) &= RS_1^{1x}(9) \cap (RS_1^{1y}(5) \cup RS_1^{1y}(6)) = 1 \\
CLASS_1^2(5) &= RS_1^{2x}(9) \cap (RS_1^{2y}(5) \cup RS_1^{2y}(6)) \\
&= 01010 \cap (01010 \cup 10001) = 01010
\end{aligned}
$$

Therefore, $CLASS_1 = 11001\ 10010\ 10001\ 00000\ 00000\ 01010$.

In the same way, the resulting type-0 signatures for P_2, P_3 and P_4 are as follows:

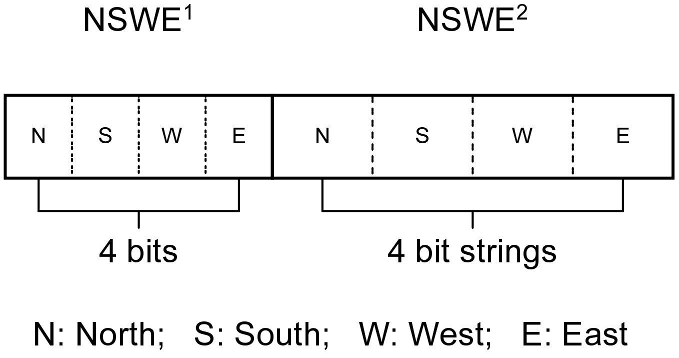


Figure 16: Type-1 signature $NSWE$

$CLASS_2$ = 01001 00000 10001 00000 00000 11100,
$CLASS_3$ = 11001 10100 10001 00000 00000 01100,
$CLASS_4$ = 11001 10100 10001 00000 00000 01100.

Given a query picture q_1 as shown in Figure 14, the corresponding type-0 query signature ($QCLASS$) is as follows:

$QCLASS$ = 11001 10100 10001 00000 00000 01100.

Next, since $QCLASS \cap CLASS_1 \neq QCLASS$, $QCLASS \cap CLASS_2 \neq QCLASS$, $QCLASS \cap CLASS_3 = QCLASS$ and $QCLASS \cap CLASS_4 = QCLASS$, we conclude that pictures P_3 and P_4 are of type-0 similarity with the query picture q_1, while pictures P_1 and P_2 are not of type-0 similarity with the query picture q_1.

6 Type-1 Signatures Based on Bit-Strings

Figure 16 shows the structure of the type-1 signature, $NSWE$. Given a type-2 record signature in a complete from, we now present the algorithm to convert such a type-2 record signature into the related type-1 record signature ($NSWE$) as follows.

Algorithm Type-1
(Convert a Type-2 Record Signature into a Type-1 Record Signature)

```
(Step 1) Set every bit in $NSWE$ to 0.

(Step 2) For $i \in \{2, 4, 6, 8, 9\}$ do                                    /* North */
             if $RS^{1y}(i)$ is 1, set $NSWE^1(1)$ to 1, and let $NSWE^2(1) = NSWE^2(1) \cup RS^{2y}(i)$
```

```
(Step 3) For $i \in \{1,3,5,7,9\}$ do                                        /* South */
             if $RS^{1y}(i)$ is 1, set $NSWE^1(2)$ to 1, and let $NSWE^2(2) = NSWE^2(2) \cup RS^{2y}(i)$
(Step 4) For $i \in \{1,3,5,7,9\}$ do                                        /* West */
             if $RS^{1x}(i)$ is 1, set $NSWE^1(3)$ to 1, and let $NSWE^2(3) = NSWE^2(3) \cup RS^{2x}(i)$
(Step 5) For $i \in \{2,4,6,8,9\}$ do                                        /* East */
             if $RS^{1x}(i)$ is 1, set $NSWE^1(4)$ to 1, and let $NSWE^2(4) = NSWE^2(4) \cup RS^{2x}(i)$
```

For those four pictures P_1, P_2, P_3 and P_4 as shown in Figure 13, only pictures P_3 and P_4 are of the type-0 similarity; therefore, we only have to check whether pictures P_3 and P_4 are of type-1 similarity with a query picture q_1. We now convert the type-2 signatures of P_3 and P_4 into the type-1 signatures. Take picture P_3 as an example.

$$\begin{aligned}
NSWE_1^1(1) &= 0 \\
NSWE_1^2(1) &= 00000
\end{aligned}$$

$$\begin{aligned}
NSWE_1^1(2) &= RS_1^{1y}(1) \cup RS_1^{1y}(3) \cup RS_1^{1y}(5) = 1 \\
NSWE_1^2(2) &= RS_1^{2y}(1) \cup RS_1^{2y}(3) \cup RS_1^{2y}(5) \\
&= 10100 \cup 10001 \cup 01100 = 11101
\end{aligned}$$

$$\begin{aligned}
NSWE_1^1(3) &= RS_1^{1x}(5) = 1 \\
NSWE_1^2(3) &= RS_1^{2x}(5) = 10001
\end{aligned}$$

$$\begin{aligned}
NSWE_1^1(4) &= RS_1^{1x}(6) = 1 \\
NSWE_1^2(4) &= RS_1^{2x}(6) = 11100
\end{aligned}$$

Therefore, $NSWE_3 = 0111\ 00000\ 11101\ 10001\ 11100$.
In the same way, the resulting type-1 signature for P_4 are as follows:

$NSWE_4 = 1111\ 10101\ 01100\ 11101\ 01100$.

Given a query picture q_1 as shown in Figure 14, the corresponding type-1 query signature ($QNSWE$) is as follows:

$QNSWE = 1111\ 10101\ 01100\ 11101\ 01100$.

Next, since $QNSWE \cap NSWE_3 \neq QNSWE$ and $QNSWE \cap NSWE_4 = QNSWE$, we conclude that pictures P_4 is of type-1 similarity with the query picture q_1, while pictures P_3 is not of type-1 similarity with the query picture q_1.

7 A Comparison

The performance of an access strategy is measured by the rate of a false match (or the rate of a correct match which equals to 1 - the rate of a false match). Obviously, as the size of a signature is increased, the rate of a false match is decreased. However, the larger the size of a signature is, the more the space needs. Moreover, the design of perfect function θ_r helps to reduce the false match rate, too. Therefore, the performance of an access strategy depends on the size of a signature and the hash function θ_r, which has been studied in [10]. In general, the rate of a false match in prime-number-based algorithms will be smaller than that in bit-string-based algorithms. The reason is that the union of two bit strings may be the same as another bit string, while the product of two prime numbers will never be the same as another prime number.

As stated in [16], Y. I. Chang and Yang's access strategies can record 169 spatial relationships between objects, as compared to only 9 spatial relationships represented in other access strategies [9, 10, 32]. Due to the same reason, Y. I. Chang and Yang's access strategies can distinguish some similar pictorial pictures, while them seem to be the same in other access strategies. Given the same image databases, the same query picture and the same hash functions, the rate of a false match in Y. I. Chang and Yang's both strategies will never be greater than that of Lee et al.'s strategy [32] or C. C. Chang et al.'s strategies [9, 10]. Since our strategies are the revised versions of Y. I. Chang and Yang's access strategies [16], our strategies also provide the advantages of Y. I. Chang and Yang's access strategies as described above. Moreover, based on our revised version of Y. I. Chang and Yang's access strategies, we can efficiently construct the object, type-0 and type-1 signatures from a given type-2 signature. While in Lee et al.'s signature file strategy based on the 2D B-string representation [33], to answer queries of objects and type-i similarity, they have construct individual object, type-0, type-1 and type-2 signatures, which may result in a large storage cost.

Basically, in Lee et al.'s approach [33], disjoint coding and superimposed coding are combined to generate the signature for type-i similarity retrieval. Each type-i similarity can be classified into five types of category relationships. Naturally, disjoint coding can divide bit pattern width (W) into five disjoint fields. Each field might have many key values, therefore the disjunction of possible key values via superimposed coding is suitable in each field. For the picture shown in Figure 17, Figure 18 shows a comparison of our approach and Lee et al.'s approach. From Figure 18, we show that only 46 bits are needed in our first access strategy, while 56 bits are always needed in Lee et al.'s strategy, where the bit pattern width for objects = 8 and the bit pattern width for type-i = 16 in Lee et al.'s strategy. (Note that here, in our strategy, we use the hash function shown in Figure 19.)

Furthermore, we will use some examples to show that our approach can provide a lower rate of a false match than Lee et al.'s approach. First, for answering queries of retrieval by objects, let's take Figure 20 as an example. Based on Lee et al.'s strategy [33], pictures p_1 and p_2 have the same objects signature as follows: 00001011. The reason is that when $W = 8$, they apply the hash function which maps all bits in W to zero except the bit ('X' -

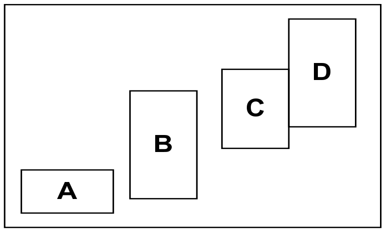


Figure 17: An example for comparing the number of bits in a signature

	(object)	(type-0)	(type-1)	(type-2)
Lee et al.[33]	00001111	0001111001000000	0001111001000000	0000111100001000
Proposed	1010000000000 1000100000000 11111 00110 10010 11111			

Figure 18: A comparison

spatial string	θ_r
AB	10001
AC	10100
BC	01100
AD	10010
BD	01010
CD	00110

Figure 19: The hash function used in Figure 17 (and Figure 22)

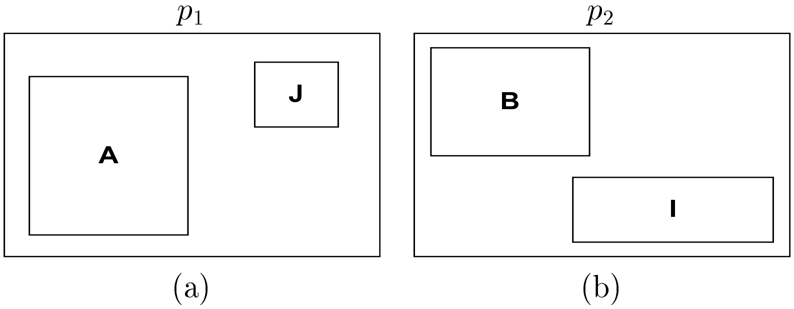


Figure 20: An example for retrieval by objects: (a) picture p_1; (b) picture p_2.

spatial string	θ_r
AB	10001
AI	10100
AJ	11000
BI	10010
BJ	00101
IJ	01010

Figure 21: The hash function used in Figure 20

'A') mod 8 + 1 being set to 1, where X is the object name, for example, B. However, based on our approach, the corresponding objects signatures of pictures p_1 and p_2 constructed from the type-2 signature using the hash function shown in Figure 21 are as follows:

$$p_1\text{: } 11000,$$
$$p_2\text{: } 10010.$$

That is, Lee et al.'s signatures will cause a false match. For the example shown in Figure 20, our approach creates two different objects signatures, while Lee et al.'s approach creates the same signature for two different pictures. Consequently, for objects signatures, our approach can provide a lower rate of false match than Lee et al.'s approach.

Second, for answering queries of type-0 similarity, let's take Figure 22 as an example. Based on Lee et al.'s strategy [33], pictures p_1 and p_2 have the same type-0 signature as follows: 0000000 000 11 01 00. The reason is that when $W = 16$, they use the following function H to create the key value (as the input to the hash function):

$$H = (`X' - `A') \times 13 + (`Y' - `X')$$

and the hash function (called the transformation function):

$$T_j = H\ mod\ w_j + 1,$$

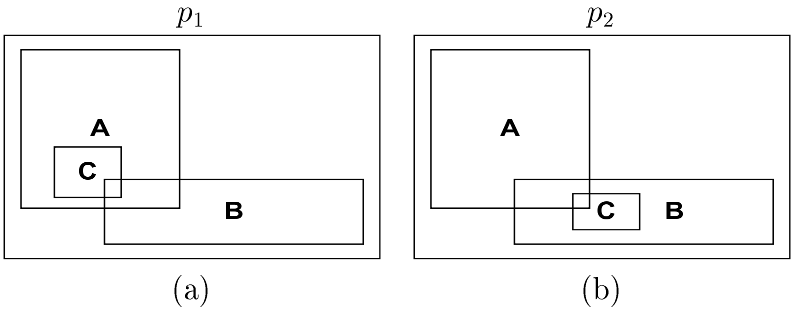


Figure 22: An example for type-0 similarity: (a) picture p_1; (b) picture p_2.

where X and Y are object names, $1 \leq j \leq 5$, $w_1 = 7$, $w_2 = 3$, $w_3 = 2$, $w_4 = 2$ and $w_5 = 2$. Note that w_1, w_2, w_3, w_4, and w_5 denote the bit pattern width of the disjoin, join, partly overlap, contain and belong categories, respectively. However, based on our approach, the corresponding type-0 signatures of pictures p_1 and p_2 constructed from the type-2 signature are as follows:

p_1: 00101 00000 00000 10100 00000 11101,
p_2: 00101 00000 00000 01100 00000 10101.

That is, Lee et al.'s signatures will cause a false match. For the example shown in Figure 22, our approach creates two different type-0 signatures, while Lee et al.'s approach creates the same signature for two different pictures. Consequently, for type-0 signatures, our approach can provide a lower rate of a false match than Lee et al.'s approach, so can be the cases for type-1 and type-2 signatures.

To clearly compare the performance of our proposed strategy and Lee et al.'s [33] strategy, we do a simulation study for both strategies. In this simulation, we consider 20 (= N) different objects and 2000 pictures in the database. For each object, it can appear in a picture with 100000 * 100000 points. For each picture, M different objects are randomly generated to appear in the picture. There are 100 query pictures, where each query picture contains 2 different objects. For Lee et al.'s strategy, we let the bit pattern width for an object record signature be 15, and the total bit pattern width for a type-0 record signature be 50 * 5 (with 50 bits for each category), for a type-1 record signature be 250 * 5 and for a type-2 record signature be 250 * 2, resulting in a total 2015 bits as a record signature for a picture. For our proposed strategy, we let the number of bits in a record signature be 100, and the the number of 1's appearing in each record signature be 2, resulting in a total 2626 bits as a record signature for a picture. Note that the size of 2626 bits is the upper bound of a type-2 record signature of our strategy, but actually, the average size of the reduced form of our type-2 record signatures for a picture is far smaller than 2626. When $M = 10$ and 5, the average size of the reduced form of our type-2 record signatures is 1489 and 942 bits, respectively, from this simulation result. The maximum size of the reduced from of our type-2 record signatures for a picture is 2026 and 1426, when M= 10

	Object	Type-0	Type-1	Type-2
2D-B based	25%	63%	9.6%	1.2%
Proposed	67%	78%	43%	30%

(a)

	Object	Type-0	Type-1	Type-2
2D-B based	49%	49%	5%	0.4%
Proposed	49%	54%	23%	6%

(b)

Table 4: A comparison of the correct match rate: (a) M=5; (b) M= 10.

and 5, respectively. The minimum size of the reduced from of our type-2 record signatures for a picture is 1026 and 526, when M= 10 and 5, respectively. Tables 4-(a) and 4-(b) show the simulation result, the correct match rate, of these two strategies when $M = 5$ and 10, respectively. From this table, we show that our approach can provide a higher rate of a correct match and needs a smaller storage requirement than Lee et al.'s approach. To further reduce the number of comparison with each record signature in the image database, a block signature (BS) can be used in the proposed strategy as in other strategies [9, 16]. The algorithm to find BS is almost the same as RS. The only one difference between them is that we use another function θ_b according to the given k_b and b_b to get the block signatures of object blocks.

8 Conclusions

In the previous approaches to retrieve symbolic pictures from a large iconic image database, they only consider pictures of type-2 similarity. When queries of objects, type-0 similarity and type-1 similarity are asked, signatures for objects, type-0 similarity and type-1 similarity are required. In this paper, first, we have presented revised version of Y. I. Chang and Yang's bit-string-based access strategy [16], which construct type-2 signatures. Based on the revised version, we have proposed algorithms to convert those type-2 signatures into *object* signatures, *type-0* signatures and *type-1* signatures so that we can answer queries of object similarity, type-0 similarity, type-1 similarity and type-2 similarity by storing only one type of signatures, i.e., the type-2 signatures. While in [33], Lee et al. have to construct 4 types of signatures in the database to achieve the same goal. From our simulation, we have shown that our approach can provide a higher rate of a correct match and needs a smaller storage requirement than Lee et al.'s approach.

References

[1] A. F. Abate, M. Nappi, G. Tortora, and M. Tucci, "IME: An Image Management Environment with Content-Based Access," *Image and Vision Computing,* Vol. 17, No. 13, pp. 967-980, Nov. 1999.

[2] S. K. Bhatia and C. L. Sabharwal, "A Fast Implementation of A Perfect Hash Function for Picture Objects," *Pattern Recognition,* Vol. 27, No. 3, pp. 365-376, Mar. 1994.

[3] I. Bloch, "Fuzzy Relative Position Between Objects in Image Processing : A Morphological Approach," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Vol. 21, No. 7, pp. 657-664, July 1999.

[4] S. K. Chang, Q. Y. Shi and C. W. Yan, "Iconic Indexing by 2D Strings," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* Vol. PAMI-9, No. 3, pp. 413-428, May 1987.

[5] S. K. Chang, C. W. Yan, D. C. Dimitroff and T. Arndt, "An Intelligent Image Database System," *IEEE Trans. on Software Eng.,* Vol. 14, No. 5, pp. 681-688, May 1988.

[6] S. K. Chang, E. Jungert and Y. Li, "Representation and Retrieval of Symbolic Pictures using Generated 2D Strings," *Proc. of Visual Communications and Image Processing,* Philadelphia, pp. 1360-1372, 1989.

[7] S. K. Chang and Y. Li, "Representation of Multi-Resolution Symbolic and Binary Pictures Using 2D-H Strings," *Proc. IEEE Workshop on Languages for Automata,* Maryland, pp. 190-195, 1988.

[8] S. K. Chang, E. Jungert and G. Tortora, *Intelligent Image Database Systems,* World Scientific Press, Singapore, 1996.

[9] C. C. Chang and J. C. Lin, "A Fast Spatial Match Accessing Scheme for Symbolic Pictures," *Journal of Electrical Engineering,* Vol. 33, No. 3, pp. 129-137, June 1990.

[10] C. C. Chang and H. C. Wu, "A Module-Oriented Signature Extraction to Retrieve Symbolic Pictures," *Journal of Computer,* Vol. 2, No. 4, pp. 45-54, April 1990.

[11] C. C. Chang, "Spatial Match Retrieval of Symbolic Pictures," *Journal of Information Science and Engineering,* Vol. 7, No. 3, pp. 405-422, Sept. 1991.

[12] C. C. Chang and S. Y. Lee, "Retrieval of Similar Pictures on Pictorial Databases," *Pattern Recognition,* Vol. 24, No. 7, pp. 675-680, July 1991.

[13] C. C. Chang and D. C. Lin, "A Spatial Data Representation: An Adaptive 2D-H String," *Pattern Recognition Letters,* Vol. 17, No. 2, pp. 175-185, Feb. 1996.

[14] C. C. Chang and C. F. Lee, "A Spatial Match Retrieval Mechanism for Symbolic Pictures," *Journal of Systems and Software,* Vol. 44, No. 1, pp. 73-83, Dec. 1998.

[15] Y. I. Chang and B. Y. Yang, “A Prime-Number-Based Matrix Strategy for Efficient Iconic Indexing of Symbolic Pictures,” *Pattern Recognition,* Vol. 30, No. 10, pp. 1745-1757, Oct. 1997.

[16] Y. I. Chang and B. Y. Yang, “Efficient Access Methods for Image Databases,” *Information Processing Letters,* Vol. 64, No. 2, pp. 95-105, Feb. 1997.

[17] Y. I. Chang and H. Y. Ann, “A Note on Adaptive 2D-H strings,” *Pattern Recognition Letters,* Vol. 20, No. 1, pp. 15-20, Jan. 1999.

[18] Y. I. Chang, H. Y. Ann and W. H. Yeh, ”A Unique-Id-Based Matrix Strategy for Efficient Iconic Indexing of Symbolic Pictures,” *Pattern Recognition,* Vol. 33, No. 8, pp. 1263-1276, Aug. 2000.

[19] J. P. Eakins, “Automatic Image Content Retrieval-Are We Going Anywhere?,” *Proc. of the 3rd Int. Conf. on Electronic Library and Visual Information Research, De Montfort University, Milton Keynes,* May. 1996.

[20] E. A. El-kwae and M. R. Kabuka, “Efficient Content-Based Indexing of Large Image Databases,” *ACM Trans. on Information Systems,* Vol. 18, No. 2, pp. 171-210, Apr. 2000.

[21] C. Faloutsos, “Description and Performance Analysis of Signature File Methods for Office Filing,” *ACM Trans. on Office Information Systems,* Vol. 5, No. 4, pp. 237-257, Dec. 1987.

[22] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B.Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D.Steele, and P. Yanker, “Query by Image and Video Content: the QBIC System,” *IEEE Computer,* Vol. 44, No. 9, 23-32, Sept. 1995.

[23] A. Gupta, T.Weymouth, and R. Jain, “Semantic Queries with Pictures: The VIMSYS Model,” *in Proc. of the 17th Int. Conf. on Very Large Databases, Barcelona, Spain,* pp. 69-79, 1991.

[24] V. N. Gudivada and V. V. Raghavan, “Design and Evaluation of Algorithms for Image Retrievals by Spatial Similarity,” *ACM Trans. on Information Systems,* Vol. 13, No. 2, pp. 115-114 , Apr. 1995.

[25] V. N. Gudivada, “$\theta\Re$-string: a Geometry-Based Representation for Efficient and Effective Retrieval of Images by Spatial Similarity,” *IEEE Trans. on Knowledge and Data Engineering,* Vol. 10, No. 3, pp. 504-512, May 1998.

[26] F. J. Hsu and S. Y. Lee, ”Similarity Retrieval by 2D C-Trees Matching in Image Databases,” *Journal of Visual Communication and Image Representation,*, Vol. 9, No. 1, pp. 87-100, March 1998.

[27] F. J. Hsu, S. Y. Lee and B. S. Lin, ”2D C-Tree Spatial Representation for Iconic Image,” *Journal of Visual Languages and Computing,* Vol. 10, No. 2, pp. 147-164, Feb. 1999.

[28] P. W. Huang and Y. R. Jean, “Reasoning about Pictures and Similarity Retrieval for Image Information Systems based on SK-Set Knowledge Representation,” *Pattern Recognition,* Vol. 28, No. 12, pp. 1915-1925, Dec. 1995.

[29] E. Jungert, “Extended Symbolic Projections as a Knowledge Structure for Spatial Reasoning,” *Proc. of the 4th BPRA Conf. on Pattern Recognition,* pp. 343-351, Springer, Cambridge, 1988.

[30] E. Jungert and S. K. Chang, “The σ-tree—a Symbolic Spatial Data Model,” *Proc. of the 11th Int. Conf. on Pattern Recognition,* The Netherlands, pp. 461-465, 1992.

[31] S. Y. Lee and F. J. Hsu, “2D C-String: A New Spatial Knowledge Representation for Image Database Systems,” *Pattern Recognition,* Vol. 23, No. 10, pp. 1077-1087, Oct. 1990.

[32] S. Y. Lee and M. K. Shan, “Access Methods of Image Database,” *International Journal of Pattern Recognition and Artificial Intelligence,* Vol. 4, No. 1, pp. 27-44, Jan. 1990.

[33] S. Y. Lee, M. C. Yang and J. W. Chen, “Signature File as a Spatial Filter for Iconic Image Database,” *Journal of Visual Languages and Computing,* Vol. 3, No. 4, pp. 373-397, April 1992.

[34] S. Y. Lee, M. C. Yang and J. W. Chen, “2D B-String: A Spatial Knowledge Representation for Image Database Systems,” *Proc. of 1992 Int. Computer Symposium,* pp. 609-615, 1992.

[35] M. S. Lew, “Next-Generation Web Searches for Visual Content,” *IEEE Computer,* Vol. 33, No. 11, pp. 46-53, Nov. 2000.

[36] X. Li and X. Qu, “Matching Spatial Relations Using db-Tree for Image Retrieval,” *Proc. of the 14th Int. Conf. on Pattern Recognition,* pp. 1230-1234, 1998.

[37] M. D. Marsicoi, L. Cinque, and S. Levialdi, “Indexing Pictorial Documents by Their Content: A Survey of Current Techniques,” *Image and Vision Computing,* Vol. 15, No. 2, pp. 119-141, Feb. 1997.

[38] M. Nabil, A. H. H. Ngu and J. Shepherd, “Picture Similarity Retrieval Using the 2D Projection Interval Representation,” *IEEE Trans. on Knowledge and Data Engineering,* Vol. 8, No. 4, pp. 533-539, Aug. 1996.

[39] R. Sacks-Davis and A. Kent, “Multikey Access Methods Based on Superimposed Coding Techniques,” *ACM Trans. on Database Systems,* Vol. 12, No. 4, pp. 655-696, Dec. 1987.

[40] A. Soffer and H. Samet, ”Pictorial Queries by Image Similarity,” *Proc. of the 13th Int. Conf. on Pattern Recognition,* Vienna, Austria, pp. 114-119, 1996.

[41] Y. Tao and W. I. Grosky, “Image Indexing and Retrieval Using Object-Based Point Feature Maps,” *Journal of Visual Languages and Computing,* Vol. 11, No. 3, pp. 323-343, March 2000.

[42] J. K. Wu, A. D. Narasimhalu, B. M. Mehtre, C. P. Lam, and Y. J. Gao, “CORE: A Content-Based Retrieval Engine for Multimedia Information Systems,” *ACM Multimedia Systems,* Vol. 3, No. 10, 25-41, Oct. 1995.

Appendix A

A Revised Version of the Prime-Number-Based Access Strategy

Since in C. C. Chang et al.'s fast spatial match access strategy [9], bit-patterns are used as signatures and a bit-wise-or operation is used to union the signatures of the set of object pairs, a false match may occur. Note that the result of a bit-wise-or operation for some signatures may form another signature which is already defined. Therefore, to further reduce the rate of false match, in [10], C. C. Chang proposed a *module-oriented signature extraction* that uses the module operation to filter out the impossible images [10]. In this strategy, the bit strings used in C. C. Chang's fast spatial match access strategy [9] are replaced with prime numbers, the bit-wise-or operation is replaced with a multiply operation, and the bit-wise-and operation in the query processing is replaced with a module operation.

Following the similar idea, Y. I. Chang and Yang presented a prime-number-based access strategy [16]. In Y. I. Chang and Yang's second strategy, the data structure of a record signature is almost the same as their first strategy. The only one difference is that RS^2 consists of 26 products of prime numbers instead of 26 bit strings. And $RS^{2x}(i) = \Pi\ \theta(ABi)$, $ABi \in T^x$, $1 \leq i \leq 13$, $RS^{2y}(i) = \Pi\ \theta(ABi)$, $ABi \in T^y$, $1 \leq i \leq 13$. Similarly, to efficiently construct object, type-0 and type-1 signatures from a given type-2 signature based on the prime number approach, we now present a revised version of Y. I. Chang and Yang's prime-number-based access strategy. The main difference between the revised version and the original one [16] is the assignment of a unique value to each of spatial operators. Take Figure 12 as an example, we can have

RS = 1010000010000 001001001000 3 2 5 5 2 3,

where the hash function θ is shown as follows.

spatial string	θ
AB	2
AC	3
BC	5

The main differences between the algorithms for the prime-number-based version and the bit-pattern-based version are as follows:

1. Replace $OSR = 0$ ($CLASS^2 = 0$, $NSWE^2 = 0$) in the bit-pattern-based version with $OSR = 1$ ($CLASS^2 = 1$, $NSWE^2 = 1$) in the prime-number-based version.

2. Replace the $\cup$ operator in the bit-pattern-based version with the $\times$ operator in the prime-number-based version in some places.

3. Replace the $\cap$ operator in the bit-pattern-based version with the GCD operator in the prime-number-based version in some places.

4. Replace the $\bigcup_i^j$ operator in the bit-pattern-based version with the $\prod_i^j$ operator in the prime-number-based version in some places.

Given a type-2 prime-number-based record signature, the algorithms to convert such a type-2 record signature into the related object record signature, the related type-0 record signature, and the related type-1 record signature, are shown as follows.

Algorithm Object*

(Convert a Type-2 Record Signature into an Object Record Signature)

```
(Step 1) Let the numbers of bits which are set to 1 in $RS^{1x}$ and $RS^{1y}$ to $C^x$ and $C^y$,
         respectively. If $C^x \leq C^y$, let $TRS^1 = RS^{1x}$ and $TRS^2 = RS^{2x}$; otherwise, let
         $TRS^1 = RS^{1y}$ and $TRS^2 = RS^{2y}$.

(Step 2) Set ORS = 1.

(Step 3) For i = 1 to 13 do
         If $TRS^1(i)$ = 1 then ORS = ORS $\times$ $TRS^2(i)$, where $TRS^1(i)$ means the i-th
         bit of $TRS^1$ and $TRS^2(i)$ means the i-th prime number of $TRS^2$.
```

Algorithm Type-0*

(Convert a Type-2 Record Signature into a Type-0 Record Signature)

```
(Step 1) Set every bit in $CLASS^1$ to 0, and set every number in $CLASS^2$ to 1.

(Step 2) $CLASS^1(1) = \bigcup_{i=1}^2 RS^{1x}(i) \times \bigcup_{i=1}^2 RS^{1y}(i)$        /* Disjoin */
         if $CLASS^1(1) = 1$ then
             $CLASS^2(1) = \prod_{i=1}^2 RS^{2x}(i) \times \prod_{i=1}^2 RS^{2y}(i)$
```

```
(Step 3) $CLASS^1(2) = (\bigcup_{i=3}^{4} RS^{1x}(i) \cap \bigcup_{i=3}^{13} RS^{1y}(i)) \cup$        /* Join */
                       $(\bigcup_{i=5}^{13} RS^{1x}(i) \cap \bigcup_{i=3}^{4} RS^{1y}(i))$

         if $CLASS^1(2) = 1$ then
             $CLASS^2(2) = GCD(\prod_{i=3}^{4} RS^{2x}(i), \prod_{i=3}^{13} RS^{2y}(i)) \times$
                           $GCD(\prod_{i=5}^{13} RS^{2x}(i), \prod_{i=3}^{4} RS^{2y}(i))$

(Step 4) $CLASS^1(3) = \bigcup_{i=7}^{10} RS^{1x}(i) \cap \bigcup_{i=7}^{10} RS^{1y}(i)$        /* Contain */
         if $CLASS^1(3) = 1$ then
             $CLASS^2(3) = GCD(\prod_{i=7}^{10} RS^{2x}(i), \prod_{i=7}^{10} RS^{2y}(i))$

(Step 5) $CLASS^1(4) = \bigcup_{i=10}^{13} RS^{1x}(i) \cap \bigcup_{i=10}^{13} RS^{1y}(i)$        /* Belong */
         if $CLASS^1(4) = 1$ then
             $CLASS^2(4) = GCD(\prod_{i=10}^{13} RS^{2x}(i), \prod_{i=10}^{13} RS^{2y}(i))$

(Step 6) $CLASS^1(5) = (\bigcup_{i=5}^{6} RS^{1x}(i) \cap \bigcup_{i=5}^{13} RS^{1y}(i)) \cup$        /* Partial overlap */
                       $(\bigcup_{i=7}^{13} RS^{1x}(i) \cap \bigcup_{i=5}^{6} RS^{1y}(i)) \cup$
                       $(\bigcup_{i=7}^{9} RS^{1x}(i) \cap \bigcup_{i=11}^{13} RS^{1y}(i)) \cup$
                       $(\bigcup_{i=11}^{13} RS^{1x}(i) \cap \bigcup_{i=7}^{9} RS^{1y}(i))$
         if $CLASS^1(5) = 1$ then
             $CLASS^2(5) = GCD(\prod_{i=5}^{6} RS^{2x}(i), \prod_{i=5}^{13} RS^{2y}(i)) \times$
                           $GCD(\prod_{i=7}^{13} RS^{2x}(i), \prod_{i=5}^{6} RS^{2y}(i)) \times$
                           $GCD(\prod_{i=7}^{9} RS^{2x}(i), \prod_{i=11}^{13} RS^{2y}(i)) \times$
                           $GCD(\prod_{i=11}^{13} RS^{2x}(i), \prod_{i=7}^{9} RS^{2y}(i))$
```

Algorithm Type-1*

(Convert a Type-2 Record Signature into a Type-1 Record Signature)

```
(Step 1) Set every bit in $NSWE^1$ to 0, and set every number in $NSWE^2$ to 1.

(Step 2) For $i \in \{2, 4, 6, 8, 9\}$ do        /* North */
             if $RS^{1y}(i)$ is 1, set $NSWE^1(1)$ to 1, and let $NSWE^2(1) = NSWE^2(1) \times RS^{2y}(i)$

(Step 3) For $i \in \{1, 3, 5, 7, 9\}$ do        /* South */
             if $RS^{1y}(i)$ is 1, set $NSWE^1(2)$ to 1, and let $NSWE^2(2) = NSWE^2(2) \times RS^{2y}(i)$

(Step 4) For $i \in \{1, 3, 5, 7, 9\}$ do        /* West */
             if $RS^{1x}(i)$ is 1, set $NSWE^1(3)$ to 1, and let $NSWE^2(3) = NSWE^2(3) \times RS^{2x}(i)$
```

```
(Step 5) For $i \in \{2, 4, 6, 8, 9\}$ do                                  /* East */
             if $RS^{1x}(i)$ is 1, set $NSWE^1(4)$ to 1, and let $NSWE^2(4) = NSWE^2(4) \times RS^{2x}(i)$
```