# A SPATIAL REPRESENTATION MODEL AND SIMILARITY RETRIEVAL METHOD OF ICON IMAGES

Ying-Hong Wang and Tai-Lung Chien

*{inhon, 053672}@mail.tku.edu.tw*

*Department of Information Engineering, TamKang University, Tamsui, Taipei Hsien, 25137, Taiwan, R.O.C.*

## Abstract

*In this paper, a new spatial knowledge representation model named "Two Dimension Begin-End Boundary String"(2D Bε-string) is proposed. The 2D Bε-string represents an icon by its MBR boundaries and a number of "dummy objects". The 2D Bε-string can intuitively and naturally represent the pictorial spatial information without any spatial operator. In addition, an image similarity evaluation method based on the modified "Longest Common Subsequence" (LCS) algorithm is presented. By the proposed evaluation method, not only those images which all of the icons and their spatial relationships fully accord with the query image can be sifted out, but also those images which partial icons and/or spatial relationships are similar to the query image can be applied to. It resolves the problems that the query targets and/or spatial relationships are not certain. Our representation model and similarity evaluation also simplify the retrieval progress of linear transformations, including rotation and reflection of an image.*

*Keyword: image retrieval, image database, spatial knowledge, spatial reasoning, similarity retrieval, 2-D Strings, LCS algorithm, 2D Bε-string*

## 1. Introduction

In pictorial spatial application systems, it is very important to abstract the information existing in original images. For example, how the image icons and their characteristics are recognized, how the symbolic image is encoded and constructed, how to index and retrieve these images and evaluate their similarity corresponding to a query image, … etc. All of these are very important issues in the information retrieval and/or content-based retrieval.

There are three basic types for image indexing and retrieval: (1) by features (e.g., color, texture, shape) of the icons in images, such as the QBIC project [9] and the Virage search engine [11]; (2) by size and location of the image icons, such as R-tree [1], R*-tree [6], Quadtree [4], and Mou's [13]; (3) by relative position of the icons, such as the 2-D Strings [2] model and its variants [3,7,8,10,12,15,16,17]. The third is very suitable for those applications that do not care the actual coordinates of icon objects. For example, one may find all images which icon *A* locates at the left side and icon *B* locates at the right.

Spatial representation by 2-D Strings and its variants have gained increasing attention in a pictorial spatial database. But most of them need a series of cutting for icon objects in an image. The icons must be associated with a lot of spatial operators to accomplish the representation of related spatial information, but their representations are not intuitive. They always generate more split objects after cutting, require more space for saving, and use more complicated algorithm for image retrieval. Their similarity retrievals require massive geometric computations and focus only on those database images that own all of the icons and spatial relationships of the query image. They, however, do not mention that only part of icons and of spatial relationships are similar to the query image.

In this paper, we propose a new spatial knowledge representation model named "Two Dimension Begin-End Boundary String"(2D Bε-string). The 2D Bε-string does not need to cut any image's icons because it simply represents an icon by its MBR (Minimum Bounding Rectangle) boundaries. And by applying a number of "dummy objects", the 2D Bε-string can intuitively and naturally represent the pictorial spatial information without any spatial operators. An algorithm is also introduced, which takes into account the $O(n)$ space complexity in the best and worst cases, to generate a symbolic image, the 2D Bε-string .

In addition, we also propose an image similarity evaluation method based on the modified "Longest Common Subsequence"(LCS) algorithm [5]. By our evaluation method, not only those images that all of the icons and their spatial relationships fully accord with the query image can be sifted out, but also those images which icons and/or spatial relationships are partially similar to the query image. It resolves the problems that the query targets and/or spatial relationships are not certain. The

modified LCS algorithm takes ***O(mn)*** as the space and time complexity, where $m$ and $n$ are the number of icons in a query image and a database image, respectively. It is much easier to retrieve the linear transformations of an image represented by 2D Bε-string. The transformations include 90, 180, 270 degrees clockwise rotations and the reflections on the $x$-axis or $y$-axis.

The remainder of this paper is organized as follows. Section 2 reviews the approaches of 2-D Strings and its variants. In Section 3, we propose a new spatial knowledge representation model, called 2D Bε-string, as well as an algorithm to construct a symbolic picture using the 2D Bε-string. A similarity retrieval algorithm, modified from LCS, and the corresponding similarity evaluation progress are introduced in Section 4. In Section 5, we present a demonstration system, a visualized retrieval system, implemented by the 2D Bε-string and modified LCS algorithm. Finally, the conclusion and the future work are given in the last section.

## 2. Related Work

### 2.1. The representation model of 2-D Strings and its variants

Chang *et al.* [2] proposed an approach, called '2-D Strings', to represent the spatial information in a picture or image. The 2-D Strings uses the symbolic projection of a picture along the $x$-axis and $y$-axis. The 2D G-string [3], a variant of 2-D Strings, extends the spatial relationships into two sets of spatial operators $R_l$ and $R_g$, and cuts all the objects along their MBR boundaries. The 2D G-string unifies the spatial relationship between two cut objects. Only those operators in the set $R_g$ are enough to specify the relationship between two cut objects.

The 2D C-string [7, 10], another variant of 2-D Strings, proposes an approach to minimize the number of cutting objects. The 2D C-string leaves the leading object as a whole. It eliminates some problems associated with superfluous cutting objects generated at the 2D G-string cutting progress. It, however, is $O(n^2)$ cutting objects in the worst case.

Instead of using the cutting process, the 2D B-string [8] represents an object by two symbols. One stands for the beginning boundary of that object, the other for the end. The 2D B-string reduces spatial relationships to a single operator '='. It means that two objects have the same boundary projection if '=' is appeared.

### 2.2. The similarity retrieval and evaluation

The basic similarity retrieval and evaluation idea for 2-D Strings [2], 2D G-string [3], 2D C-string [7] and 2D B-string [8] is the same. First, they always define three types of similarity, type-$i$ ($i$ = 0, 1, 2). Each is constricted by some conditions. Type-1 is stricter than type-0 and type-2 is stricter than type-1. Second, they examine all spatial relationship pairs between any two objects in the query image versus those pairs in the image of database. They build type-$i$ subgraph if the pair satisfies type-$i$ constraints. After examining, they find the maximum complete subgraph for each type-$i$ graph. The number of objects in the maximum complete subgraph is the similarity of the query image and the images of database.

The space and time complexity to examine all spatial relationship pairs requires $O(n^2)$, where $n$ is the number of object in an image. Finding maximum complete subgraph is an NP-complete problem [18]. It is a time consuming task. It is not suitable for a large number of icon objects in an image.

## 3. The Spatial Representation Model Using 2D Bε-string

### 3.1. The model of 2D Bε-string

There are many approaches proposed to represent an icon in an image, such as MBR (Minimum Bounding Rectangle) [3,7,8,10], MBE (Minimum Bounding Ellipse) and MBC (Minimum Bounding Circle) [13]. The approach used in the 2D Bε-string is MBR. Conceptually, it is similar to the 2D B-string. However, the 2D Bε-string adopts a quite different idea to address the spatial relationship between two boundary symbols. The 2D B-string uses a **spatial operator** (=) to describe the projection of two boundaries that is **IDENTICAL**. In the 2D Bε-string, we use a **dummy object** to describe the projection of two boundaries that is **DISTINCT**!

We define a 'Dummy Object' in the following way:

***A 'Dummy Object' is not a real object in the original image. It can be specified as any size of space and be memorized as symbol 'ε'.***

The 2D Bε-string with $n$ icons thus can be defined as

$$(u, v) = (d_0x_1d_1x_2d_2\ldots d_{2n-1}x_{2n}d_{2n},\ d_0y_1d_1y_2d_2\ldots d_{2n-1}y_{2n}d_{2n}).$$

Where $d_i$ is a dummy object ε or a null string, $i$ = 0, 1, … $2n$, and $x_i$ and $y_i$ are real icon objects that are either the beginning or the end projected boundaries on the $x$-axis and the $y$-axis, respectively. Set $d_0$ to ε if there is space between the beginning boundary of the leftmost (bottommost) object and the left (bottom) edge of an image. Similarly, set $d_{2n}$ to ε if there is an interval between the end boundary of the rightmost (topmost) object and the right (top) edge of an image. For the rest, set $d_i$ to ε if the boundary projections of $x_i$ and $x_{i+1}$ ($y_i$ and $y_{i+1}$) are different.

The 2D Bε-string showed in Figure 1, for example, is written as $(u, v) = (\varepsilon A_b\varepsilon B_b\varepsilon A_e C_b\varepsilon C_e\varepsilon B_e\varepsilon,\ \varepsilon B_b\varepsilon A_b\varepsilon B_e C_b\varepsilon C_e\varepsilon$

$A_e\varepsilon$). The dummy object $d_3$ is set to a null string because the end boundary of object $A$ and the beginning boundary of object $C$ are projected at the same location on the $x$-axis. A similar case is applied to the end boundary of object $B$ and the beginning boundary of object $C$ on the $y$-axis.

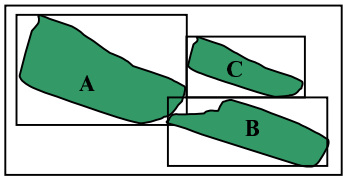


**Figure 1: An image with three objects**

Observably, the 2D Bε-string has the following advantages: ***First***, the object location in original images and symbolic pictures was mapped directly, as showed in Table 1. There is no operator required for representing the spatial relationship between objects. It is intuitively. ***Second***, it does not need to cut the objects of image, which simplifies the construction of image database. And the space complexity for an image with $n$ objects in the worst and best cases is $O(n)$. It requires $4n+1$and $2n+1$ symbols, respectively. ***Third***, it simplifies the similarity retrieval because there is no combination of the result of spatial reasoning required.

**Table 1: Mapping of spatial relationships.**

| | Symbolic picture | Original image |
|---|---|---|
| 1 | $\varepsilon A_b\varepsilon A_e\varepsilon C_b\varepsilon C_e\varepsilon$ | |
| 2 | $\varepsilon A_b\varepsilon A_e C_b\varepsilon C_e\varepsilon$ | |
| 3 | $\varepsilon A_b\varepsilon C_b\varepsilon A_e\varepsilon C_e\varepsilon$ | |
| 4 | $\varepsilon A_b\varepsilon C_b\varepsilon C_e A_e\varepsilon$ | |
| 5 | $\varepsilon A_b\varepsilon C_b\varepsilon C_e\varepsilon A_e\varepsilon$ | |
| 6 | $\varepsilon A_b C_b\varepsilon A_e\varepsilon C_e\varepsilon$ | |
| 7 | $\varepsilon A_b C_b\varepsilon A_e C_e\varepsilon$ | |
| 8 | $\varepsilon A_b C_b\varepsilon C_e\varepsilon A_e\varepsilon$ | |
| 9 | $\varepsilon C_b\varepsilon A_b\varepsilon A_e\varepsilon C_e\varepsilon$ | |
| 10 | $\varepsilon C_b\varepsilon A_b\varepsilon A_e C_e\varepsilon$ | |
| 11 | $\varepsilon C_b\varepsilon A_b\varepsilon C_e\varepsilon A_e\varepsilon$ | |
| 12 | $\varepsilon C_b\varepsilon C_e A_b\varepsilon A_e\varepsilon$ | |
| 13 | $\varepsilon C_b\varepsilon C_e\varepsilon A_b\varepsilon A_e\varepsilon$ | |

A C

### 3.2. Algorithm for constructing a symbolic picture

The algorithm **Convert-2D-Be-String**, showed in Table 2, transforms an original image to a symbolic picture. Lines 1-12 explain the meaning of variables. Lines 14-19 sort the input data by coordinates and object identifiers in ascending order for the $x$-axis and the $y$-axis separately. Lines 21-32 construct the 2D Bε-string on the $x$-axis, and lines 34-45 do the same thing for the $y$-axis.

The time complexity on loops in lines 14-18, 24-30, and 37-43 is $O(n)$, and never exceeds the sorting algorithm called in line 19. If we don't care the sorting algorithm, the space complexity is $O(n)$ too.

**Table 2: Algorithm to construct a symbolic picture.**

**Convert-2D-Be-String ($n$,$C$,$X_b$,$X_e$,$Y_b$,$Y_e$,$x_{max}$,$y_{max}$)**

1. // $n$ …number of objects in an image
2. // $C$ …name of objects, $C=\{c_1, c_2, \ldots, c_n\}$
3. // $X_b$ …beginning boundaries on the $x$-axis, $X_b=\{x_{b1}, x_{b2}, \ldots, x_{bn}\}$
4. // $X_e$ …end boundaries on the $x$-axis, $X_e=\{x_{e1}, x_{e2}, \ldots, x_{en}\}$
5. // $Y_b$ …beginning boundaries on the $y$-axis, $Y_b=\{y_{b1}, y_{b2}, \ldots, y_{bn}\}$
6. // $Y_e$ …end boundaries on the $y$-axis, $Y_e=\{y_{e1}, y_{e2}, \ldots, y_{en}\}$
7. // $x_{max}$ …maximum coordinate on the $x$-axis
8. // $y_{max}$ …maximum coordinate on the $y$-axis
9. // $X_{be}$ …2D Bε-string on the $x$-axis
10. // $Y_{be}$ …2D Bε-string on the $y$-axis
11. // $S$…a sort work for $x$-axis, $S=\{s_i \mid i=1,2,\ldots, 2n\}$
12. // $T$…a sort work for $y$-axis, $T=\{t_i \mid i=1,2,\ldots, 2n\}$
13. // Combine MBR and object identifier as a key
14. for $i$=1 to $n$
15. $\quad s_i \leftarrow x_{bi}c_i$
16. $\quad s_{i+n} \leftarrow x_{ei}c_i$
17. $\quad t_i \leftarrow y_{bi}c_i$
18. $\quad t_{i+n} \leftarrow y_{ei}c_i$
19. Sorting $S$ and $T$ by ascending order
20. // Construct 2D Bε-string on the $x$-axis
21. $X_{be} \leftarrow$ '' // Initialized by a null string
22. if $x_b$ of $s_1 \neq 0$ then // Insert ε at the leftmost?
23. $\quad X_{be} \leftarrow \varepsilon$
24. for $i$=1 to $2n$-1
25. $\quad$ if type of $x$ in $s_i$ is $x_b$ then // Convert to
26. $\quad\quad X_{be} \leftarrow X_{be}c_{bi}$ // boundary symbol
27. $\quad$ else
28. $\quad\quad X_{be} \leftarrow X_{be}c_{ei}$
29. $\quad$ if $x$ of $s_i \neq x$ of $s_{i+1}$ then
30. $\quad\quad X_{be} \leftarrow X_{be}\varepsilon$
31. if $x_e$ of $s_{2n} \neq x_{max}$ then // Insert ε at the rightmost?
32. $\quad X_{be} \leftarrow X_{be}\varepsilon$
33. // Construct 2D Bε-string on the $y$-axis
34. $Y_{be} \leftarrow$ ''
35. if $y_b$ of $t_1 \neq 0$ then // Insert ε at the bottommost?
36. $\quad Y_{be} \leftarrow \varepsilon$
37. for $i$=1 to $2n$-1
38. $\quad$ if type of $y$ in $t_i$ is $y_b$ then // Convert to
39. $\quad\quad Y_{be} \leftarrow Y_{be}c_{bi}$ // boundary symbol
40. $\quad$ else
41. $\quad\quad Y_{be} \leftarrow Y_{be}c_{ei}$
42. $\quad$ if $y$ of $t_i \neq y$ of $t_{i+1}$ then
43. $\quad\quad Y_{be} \leftarrow Y_{be}\varepsilon$
44. if $y_e$ of $t_{2n} \neq y_{max}$ then // Insert ε at the topmost?
45. $\quad Y_{be} \leftarrow Y_{be}\varepsilon$
46. return $X_{be}$, $Y_{be}$

## 4. Image Similarity Retrieval and Evaluation

### 4.1. Algorithms of similarity retrieval

In the similarity assessment, we take care the number of spatial relationships formed by every two objects and appeared in the query image and database image at the same time. In order to find the number of common spatial relationships, we propose an algorithm to find the longest common subsequence (LCS)[5] length from the 2D Bε-strings of a query image and database image. Then

measure the similarity by evaluating this LCS string with respect to the original 2D Bε-strings. The time complexity of the LCS algorithm [5] is $O(mn)$, depends on only the length of the strings.

It is not necessary to examine all the spatial relationships for every two boundary symbols. Because **the LCS string implies that, in query images and database images, all the spatial relationships of every two boundary symbols in LCS string are the same.** So, the similarity can be evaluated in a reasonable time.

This algorithm is shown in Table 3, and named as **2D-Be-LCS-Length**. This algorithm is modified from the LCS algorithm discussed by Cormen et al. [5]. There are two factors to revise the original LCS algorithm. First, we avoid picking dummy objects continuously because only one dummy object sufficiently represents the relative spatial relationship between two boundary symbols. The *if*-statement in line 21 makes this decision. Second, we omit the LCS path recording matrix by evaluating the left and up paths first, as shown in lines 16-19, and evaluating the left-up diagonal path next, as shown in lines 23-24. The LCS path, however, can still be inferred from the matrix recording the LCS length.

**Table 3: Algorithm to calculate the LCS length.**

```
2D-Be-LCS-Length (Q, D)
1.  m←length(Q)
2.  n←length(D)
3.  // Q is a 2D Bε-string of a query image, Q = {q_i | i =
        1, 2, …, m}.
4.  // D is a 2D Bε-string of a database image, D = {d_j | j
        = 1, 2, …, n}.
5.  // W is the LCS-length inferring table, W = {w_i,j | i =
        0, 1, 2, …, m, j = 0, 1, 2, …, n}.
6.  // Initialize the first column of W by zeros.
7.  for i←1 to m do
8.      w_i,0←0
9.  // Initialize the first row of W by zeros.
10. for j←0 to n do
11.     w_0,j←0
12. // Infer each cell until all cells was evaluated.
13. for i = 1 to m do
14.     for j = 1 to n do
15. // Set current cell value.
16.         if |w_i-1,j| ≥ |w_i,j-1| then
17.             w_i,j←w_i-1,j
18.         else
19.             w_i,j←w_i,j-1
20. // Check the symbol q_i, d_j and the last symbol of LCS
        path from left-up diagonal
21.         if (q_i = d_j) and ((q_i ≠ ε) or (w_i-1,j-1 ≥ 0)) then
22. // If all are hold
23.             if (|w_i-1,j-1| +1) > |w_i,j| then
24.                 w_i,j←|w_i-1,j-1| +1
25.             if q_i = ε then
26.                 w_i,j←-w_i,j
27. return W
```

The algorithm **2D-Be-LCS-Length** takes $Q$ and $D$, two 2D Bε-strings, as input parameters and returns the LCS-length inferring table $W$. In a query image with $m$ objects, the maximum length of the 2D Bε-string in each dimension is $4m+1$, including $2m$ boundary symbols and at most $2m+1$ dummy objects. With the same derivation, the maximum length of the 2D Bε-string of a database image with $n$ objects is $4n+1$. The LCS-length inferring table $W$ needs $(1+(4m+1))(1+(4n+1))$ storage units; therefore, the space complexity is $O(mn)$.

In the initialization of the first row and the first column, as shown in lines 7-8 and 10-11 in Table 3, each string symbol must be set once. They will be executed $4m+1$ and $4n+2$ times, respectively. The outer loop, in line 13, examines each row of $W$ $4m+1$ times. The inner loop, in lines 14-26, examines each cell of $W$ $(4m+1)*(4n+1)$ times. Thus, the time complexity is $O((4m+1)+(4n+2)+(4m+1)*(4n+1))$, same as $O(mn)$.

We also give a recursive procedure to print a longest common subsequence string of two 2D Bε-strings. This algorithm is shown in Table 4. The initial invocation is **Print-2D-Be-LCS** ($Q$, $W$, *length*($Q$), *length*($D$)). From the last cell of the LCS-length inferring table $W$, this procedure decreases $i$ and/or $j$ along the directions left and/or up in each recursive call until either $i$ or $j$ reaches zero. Then all symbols of LCS string are printed out in the proper, forward order.

**Table 4: Algorithm to print LCS string.**

```
Print-2D-Be-LCS (Q, W, i, j)
1.  // Q is a 2D Bε-string of query image, Q = {q_i | i = 1,
        2, …, m}.
2.  // W is the LCS-length inferring table induced from
        algorithm in Table 3.
3.  if i = 0 or j = 0 then
4.      return
5.  if |w_i,j| = |w_i-1,j| then
6.      Print-2D-Be-LCS(Q, W, i-1, j)
7.  else if |w_i,j| = |w_i,j-1| then
8.      Print-2D-Be-LCS(Q, W, i, j-1)
9.  else
10.     Print-2D-Be-LCS(Q, W, i-1, j-1)
11.     print q_i
12. return
```

Due to the revision of finding the LCS length from the two 2D Bε-strings in Table 3, we need to compare the LCS string length of the current cell with the string length of up one before printing LCS string symbols. It certainly implies that the LCS paths are induced from up direction if they are the same. The corresponding boundary symbol or dummy object doesn't belong to a symbol of LCS string. We ignore this symbol and continuously induce along the up direction, as lines 5-6. On the other hand, if they are not the same, we need to compare current cell's LCS string length with the left cell's length. If they are the same, the LCS is induced from the left direction. With the same reason as in the preceding case, we also ignore this symbol and continuously induce along the left direction, as lines 7-8. If the LCS is not from the up or left direction, it must be induced from the left-up diagonal direction. The current cell associated with the boundary symbol or dummy

object must be part of the LCS string. After processing all cells on the left/up direction recursively, we print out this symbol, as lines 9-11.

When recursively calling in lines 6, 8 and 10 each time, either $i$ or $j$ is decreased by one. This algorithm can print out all LCS string symbols after $m+n$ times recursive calls are implemented at most. The time complexity is $O(m+n)$.

## 4.2. The similarity evaluation

After obtaining the LCS length and the LCS string of a query image and a database image, we need to evaluate their similarity. Conceptually, the longer the LCS string is, the more two images look similar. For example, string in one dimension of 2D Bε-string is:

Query image 1: $\varepsilon B_b \varepsilon C_b \varepsilon B_e \varepsilon A_b \varepsilon C_e \varepsilon A_e \varepsilon$,

Database image 1: $\varepsilon B_b \varepsilon B_e \varepsilon D_b \varepsilon A_b \varepsilon D_e \varepsilon A_e \varepsilon$,

LCS string 1: $\varepsilon B_b \varepsilon B_e \varepsilon A_b \varepsilon A_e \varepsilon$,

Database image 2: $\varepsilon B_b \varepsilon C_b \varepsilon B_e \varepsilon D_b \varepsilon A_b \varepsilon D_e \varepsilon C_e \varepsilon A_e \varepsilon$,

LCS string 2: $\varepsilon B_b \varepsilon C_b \varepsilon B_e \varepsilon A_b \varepsilon C_e \varepsilon A_e \varepsilon$.

With the same query image1, database image 2 is a better result than database image 1 because its LCS string is longer than the LCS string 1. However, two database images with the same LCS string length do not necessarily lead to the same similarity. For example:

Query image 2: $\varepsilon B_b \varepsilon C_b \varepsilon B_e A_b \varepsilon C_e \varepsilon A_e$,

Database image 3: $\varepsilon B_b \varepsilon B_e \varepsilon D_b \varepsilon A_b \varepsilon D_e \varepsilon A_e \varepsilon$,

LCS string 3: $\varepsilon B_b \varepsilon B_e A_b \varepsilon A_e$,

Database image 4: $\varepsilon B_b \varepsilon D_b \varepsilon B_e A_b \varepsilon D_e \varepsilon A_e \varepsilon$,

LCS string 4: $\varepsilon B_b \varepsilon B_e A_b \varepsilon A_e$.

The database image 4 has a better matching result than the database image 3, because the object B adjoins the object A in the query image 2 and database image 4, but not in the database image 3. Thus, the database image 4 has better similarity than the database image 3.

As analyzed above, we propose an assessment to identify the similarity of the preceding phenomena. Before describing this assessment, we define the following notations for the 2D Bε-string:

- $N$ : **number of objects in a query image;**
- $Q_x$ : **string length along $x$-axis in a query image;**
- $Q_y$ : **string length along $y$-axis in a query image;**
- $L_x$ : **length of LCS string with dummy objects along $x$-axis;**
- $L_y$ : **length of LCS string with dummy objects along $y$-axis;**
- $M_x$ : **length of $L_x$ string without dummy object;**
- $M_y$ : **length of $L_y$ string without dummy object;**
- $D_x$ : **length of boundary symbols with spatial relationships in database image based on $M_x$;**
- $D_y$ : **length of boundary symbols with spatial relationships in database image based on $M_y$;**
- $W_x$ : **similarity weight along $x$-axis, $0 \le W_x \le 1$;**
- $W_y$ : **similarity weight along $y$-axis, $0 \le W_y \le 1$, $W_x + W_y = 1$;**
- $S_x$ : **similarity along $x$-axis, $0 \le S_x \le 1$,**

$$S_x = \begin{cases} 1-(Q_x + D_x - 2L_x)/(4N+1) & \text{if } M_x > 0, \\ 0 & \text{if } M_x = 0; \end{cases}$$

- $S_y$ : **similarity along $y$-axis, $0 \le S_y \le 1$,**

$$S_y = \begin{cases} 1-(Q_y + D_y - 2L_y)/(4N+1) & \text{if } M_y > 0, \\ 0 & \text{if } M_y = 0; \end{cases}$$

- $S$ : **similarity of a query image and a database image, $S = W_x S_x + W_y S_y$, $0 \le S \le 1$.**

The LCS string length $L_x$ and $L_y$ can be obtained by using the algorithm in Table 3. The value of $M_x$ and $M_y$ can be calculated by subtracting the number of dummy objects from the value of $L_x$ and $L_y$, respectively. $W_x$ and $W_y$ can emphasize the importance of similarity in the $x$-axis or $y$-axis.

As mentioned in Section 3.1, an image with $n$ objects has at most $4n+1$ symbols on each dimension of its 2D Bε-string. Among which $2n$ symbols are boundary symbols and the rest $2n+1$ symbols, interspersed among the boundary symbols, are dummy objects. Now let's picture each symbol as a bucket. Thus, an image with $n$ objects will have at most $4n+1$ buckets. If the symbol doesn't exist, the associated bucket is considered as empty. For example, the query image 2 with three objects A, B, and C has at most 4*3+1=13 symbols. The boundary string is $\varepsilon B_b \varepsilon C_b \varepsilon B_e A_b \varepsilon C_e \varepsilon A_e$. Due to the lack of a dummy object ε between $B_e$ and $A_b$, the bucket 7 (Figure 2) is empty. Because the ending boundary of object A is same as the right-edge of the image, the end of boundary string also lacks a dummy object; therefore, the bucket 13 is empty. Another example shown in Figure 3 for the database image 4, the boundary string $\varepsilon B_b \varepsilon D_b \varepsilon B_e A_b \varepsilon D_e \varepsilon A_e \varepsilon$ also has an empty bucket 7.

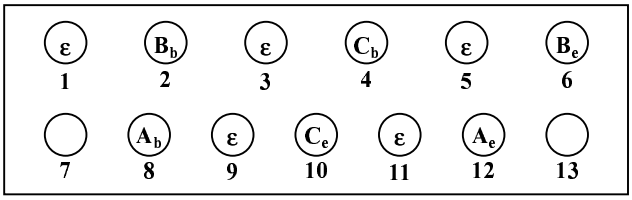


**Figure 2: Symbols in the buckets of query image 2.**

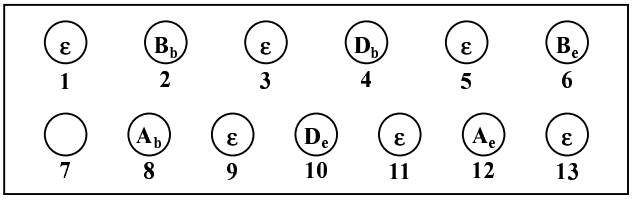


**Figure 3: Symbols in the bucket of database image 4.**

In comparison with the query image in Figure 2 and the database image in Figure 3, we categorize those buckets in query images into four classes and summarize the number of buckets for each class listed in Table 5.

Note that the class IV represented the LCS information of a query image and a database image. Class I and IV together represents the similar portion of both images. The similar portion also can be obtained from the complementarily normalized portion of the total buckets number in the class II and III. The number of symbols in class I and IV along the $x$-axis and $y$-axis are $(4n+1) - (Q_x-L_x+D_x-L_x)$ and $(4n+1) - (Q_y-L_y+D_y-L_y)$, respectively. The similarity on the $x$-axis ($S_x$) is $1-(Q_x+D_x-2L_x) / (4n+1)$ and the similarity on the $y$-axis ($S_y$) is $1-(Q_y+D_y-2L_y) / (4n+1)$. Clearly, the similarity of the entire image is the sum of $S_x$ and $S_y$ with a proper weight, that is, $S=W_xS_x+W_yS_y$.

**Table 5: The classifications of buckets.**

| Class | Descriptions | Number | Buckets |
|---|---|---|---|
| **I** | **There is no dummy object between two boundary symbols in both query image and database image.** | **?** | **7** |
| **II** | **There is no dummy object between two boundary symbols in query image, but has a dummy object of them in database image.** | **$D_x$-$L_x$, $D_y$-$L_y$** | **13** |
| **III** | **Symbols in a query image but not in a database image.** | **$Q_x$-$L_x$, $Q_y$-$L_y$** | **4, 5, 10 and 11** |
| **IV** | **Symbols with same spatial relationship in both query image and database image.** | **$L_x$, $L_y$** | **1, 2, 3, 6, 8, 9 and 12** |

Finally, we explain the detail steps for calculating similarity by an example. The query image in Figure 1 has three objects, so $N$=3. The 2D Bε-string is presented by ($\varepsilon A_b\varepsilon B_b\varepsilon A_e C_b\varepsilon C_e\varepsilon B_e\varepsilon$, $\varepsilon B_b\varepsilon A_b\varepsilon B_e C_b\varepsilon C_e\varepsilon A_e\varepsilon$) and the string length on the $x$-axis and $y$-axis, ($Q_x$, $Q_y$), is (12, 12).

The database image in Figure 4 has a 2D Bε-string ($\varepsilon E_b\varepsilon A_b\varepsilon B_b\varepsilon A_e\varepsilon C_b\varepsilon F_b\varepsilon E_e\varepsilon C_e\varepsilon B_e\varepsilon F_e\varepsilon$, $\varepsilon E_b\varepsilon B_b\varepsilon E_e\varepsilon A_b\varepsilon B_e C_b$ $\varepsilon F_b\varepsilon C_e\varepsilon A_e\varepsilon F_e\varepsilon$). The LCS-length inferring table on the $x$-axis and $y$-axis of a query image and database image with dummy objects are shown in Table 6 and Table 7, respectively. The LCS strings are $\varepsilon A_b\varepsilon B_b\varepsilon A_e C_b\varepsilon C_e\varepsilon B_e\varepsilon$ and $\varepsilon B_b\varepsilon A_b\varepsilon B_e C_b\varepsilon C_e\varepsilon A_e\varepsilon$; thus, the length pair ($L_x$, $L_y$) equals to (12, 12). The value of ($M_x$, $M_y$) can be calculated by subtracting the number of dummy objects in the LCS string from ($L_x$, $L_y$), thus, ($M_x$, $M_y$) equals to (6, 6). The boundary symbols with spatial relationships in a database image based on the strings of $M_x$ and $M_y$ are $\varepsilon A_b\varepsilon B_b\varepsilon A_e\varepsilon C_b\varepsilon C_e\varepsilon B_e\varepsilon$ and $\varepsilon B_b\varepsilon A_b\varepsilon B_e C_b\varepsilon C_e\varepsilon A_e\varepsilon$; thus, ($D_x$, $D_y$) equals to (13,12). Clearly the similarity on the $x$-axis and $y$-axis can be calculated by

$$S_x = 1-(Q_x+D_x-2L_x)/(4N+1) = 1-(12+13-2*12)/(4*3+1) = 1-(25-24)/13 = 1-0.0769 = 0.9231$$

$$S_y = 1-(Q_y+D_y-2L_y)/(4N+1) = 1-(12+12-2*12)/(4*3+1) = 1-(24-24)/13 = 1-0.0 = 1.0$$

The weights along each axis ($W_x$, $W_y$) are given as (0.5, 0.5). Then, for the database image, the similarity is $S$=0.5*0.9231+0.5*1.0=0.9616.

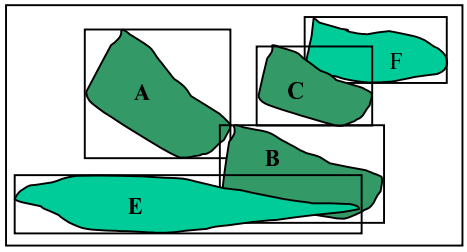


**Figure 4: Database image.**

**Table 6: LCS-length inferring table on the $x$-axis.**

| $W_{ij}$ | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $d_j$ | ε | $E_b$ | ε | $A_b$ | ε | $B_b$ | ε | $A_e$ | ε | $C_b$ | ε | $F_b$ | ε | $E_e$ | ε | $C_e$ | ε | $B_e$ | ε |
| 0 | $q_i$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | ε | 0 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| 2 | $A_b$ | 0 | -1 | -1 | -1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 3 | ε | 0 | -1 | -1 | -1 | 2 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 |
| 4 | $B_b$ | 0 | -1 | -1 | -1 | 2 | -3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 5 | ε | 0 | -1 | -1 | -1 | 2 | -3 | 4 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 |
| 6 | $A_e$ | 0 | -1 | -1 | -1 | 2 | -3 | 4 | -5 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| 7 | $C_b$ | 0 | -1 | -1 | -1 | 2 | -3 | 4 | -5 | 6 | 6 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| 8 | ε | 0 | -1 | -1 | -1 | 2 | -3 | 4 | -5 | 6 | -7 | -7 | -8 | -8 | -8 | -8 | -8 | -8 | -8 | -8 | -8 |
| 9 | $C_e$ | 0 | -1 | -1 | -1 | 2 | -3 | 4 | -5 | 6 | -7 | -7 | -8 | -8 | -8 | -8 | -8 | 9 | 9 | 9 | 9 |
| 10 | ε | 0 | -1 | -1 | -1 | 2 | -3 | 4 | -5 | 6 | -7 | -7 | -8 | -8 | -8 | -8 | -8 | 9 | -10 | -10 | -10 |
| 11 | $B_e$ | 0 | -1 | -1 | -1 | 2 | -3 | 4 | -5 | 6 | -7 | -7 | -8 | -8 | -8 | -8 | -8 | 9 | -10 | 11 | 11 |
| 12 | ε | 0 | -1 | -1 | -1 | 2 | -3 | 4 | -5 | 6 | -7 | -7 | -8 | -8 | -8 | -8 | -8 | 9 | -10 | 11 | -12 |

**Table 7: LCS-length inferring table on the $y$-axis.**

| $W_{ij}$ | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | $d_j$ | ε | $E_b$ | ε | $B_b$ | ε | $E_e$ | ε | $A_b$ | ε | $B_e$ | $C_b$ | ε | $F_b$ | ε | $C_e$ | ε | $A_e$ | ε | $F_e$ | ε |
| 0 | $q_i$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | ε | 0 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |
| 2 | $B_b$ | 0 | -1 | -1 | -1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 3 | ε | 0 | -1 | -1 | -1 | 2 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 | -3 |
| 4 | $A_b$ | 0 | -1 | -1 | -1 | 2 | -3 | -3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 5 | ε | 0 | -1 | -1 | -1 | 2 | -3 | -3 | 4 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 | -5 |
| 6 | $B_e$ | 0 | -1 | -1 | -1 | 2 | -3 | -3 | 4 | -5 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| 7 | $C_b$ | 0 | -1 | -1 | -1 | 2 | -3 | -3 | 4 | -5 | 6 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| 8 | ε | 0 | -1 | -1 | -1 | 2 | -3 | -3 | 4 | -5 | 6 | 7 | -8 | -8 | -8 | -8 | -8 | -8 | -8 | -8 | -8 | -8 |
| 9 | $C_e$ | 0 | -1 | -1 | -1 | 2 | -3 | -3 | 4 | -5 | 6 | 7 | -8 | -8 | -8 | 9 | 9 | 9 | 9 | 9 | 9 | 9 |
| 10 | ε | 0 | -1 | -1 | -1 | 2 | -3 | -3 | 4 | -5 | 6 | 7 | -8 | -8 | -8 | 9 | -10 | -10 | -10 | -10 | -10 | -10 |
| 11 | $A_e$ | 0 | -1 | -1 | -1 | 2 | -3 | -3 | 4 | -5 | 6 | 7 | -8 | -8 | -8 | 9 | -10 | 11 | 11 | 11 | 11 | 11 |
| 12 | ε | 0 | -1 | -1 | -1 | 2 | -3 | -3 | 4 | -5 | 6 | 7 | -8 | -8 | -8 | 9 | -10 | 11 | -12 | -12 | -12 | -12 |

## 4.3. The rotation and reflection of an image

If we consider the rotation (90, 180, 270 degrees clockwise) or reflection (on the $x$-axis or $y$-axis) of an image, we must perform similarity retrieval and evaluation eight times. In 2-D String, 2D G-string, and 2D C-string, we must do a proper string transformation for each dimension each times. It includes that the string may need to be reversed, and a sophisticated formula to transform spatial operators is required [14]. Even thought the 2D B-string, it still needs to recalculate their rank values.

If an image is represented by a 2D Bε-string, then the

similarity retrieval of rotation and reflection for an image becomes very easy. Before evaluation, it needs to reverse the string only, if required. Because the dummy object is not a spatial operator, its meaning is not varied while the image is rotated or reflected.

Assume that $(u, v)$ is a 2D Bε-string with $m$ and $n$ symbols in the $x$-axis and $y$-axis, respectively, that is, $u=\{u_1u_2...u_{m-1}u_m\}$, $v=\{v_1v_2...v_{n-1}v_n\}$. We can define the reversed string $u^{-1}=\{u_mu_{m-1}...u_2u_1\}$ for $u$ and $v^{-1}=\{v_nv_{n-1}...v_2v_1\}$ for $v$. The 2D Bε-strings between the original image and the image after rotation and/or reflection are summarized in Table 8.

**Table 8: The rotation and reflection checklist.**

| | Rotation/reflection | 2D Bε-string |
|---|---|---|
| 1 | Original image | $(u, v)$ |
| 2 | 90 degree clockwise | $(v, u^{-1})$ |
| 3 | 180 degree clockwise | $(u^{-1}, v^{-1})$ |
| 4 | 270 degree clockwise | $(v^{-1}, u)$ |
| 5 | Reflect vs. $x$-axis | $(u, v^{-1})$ |
| 6 | Reflect vs. $y$-axis | $(u^{-1}, v)$ |
| 7 | 90 degree clockwise and reflect vs. $x$-axis | $(v, u)$ |
| 8 | 90 degree clockwise and reflect vs. $y$-axis | $(v^{-1}, u^{-1})$ |

## 5. The Implementation

We have implemented a visual image retrieval system using the approaches of 2D Bε-string spatial representation model and the modified LCS similarity evaluation algorithm. The system architecture is showed in Figure 5.

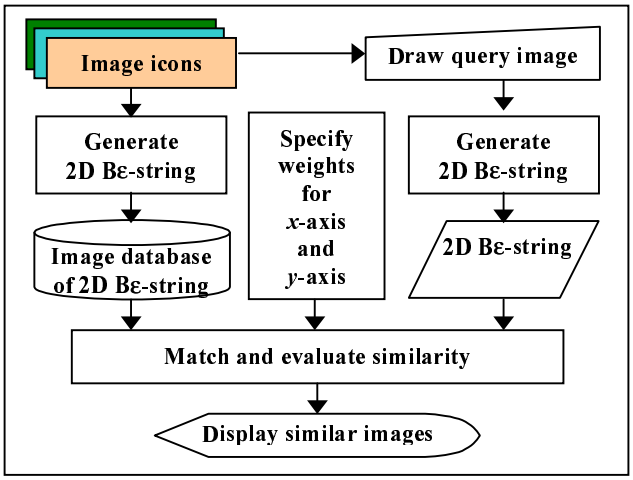


**Figure 5: System architecture.**

After the retrieval system is started, the user can load an image databases from storage and browse them one by one thru the scrolling bar. The user may form the query image by putting a number of available icons in the work area, dragging them to an appropriate location, scaling them to an apposite size (Figure 6), and hit 'Search' button while completing the layout. The system will transform the query image to 2D Bε-string, and compare with the database images. Then the most similar image and its similarity information are displayed (Figure 7). One can switch to the 2D Bε-string (Figure 8) of that image by clicking on '2D Bε-string' tab. The user may browse those images next a lower degree of similarity thru the scrolling bar too.

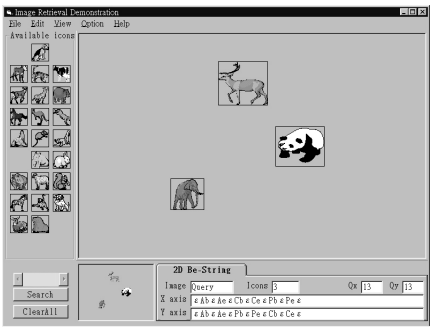

**Figure 6: Arrange icons in the query image.**

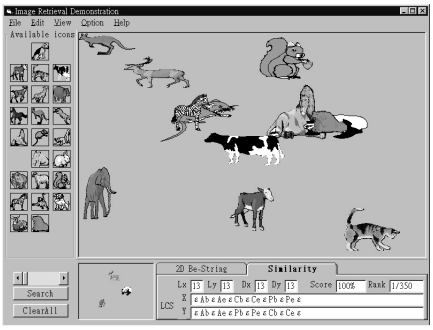

**Figure 7: Show the most similar image with similarity.**

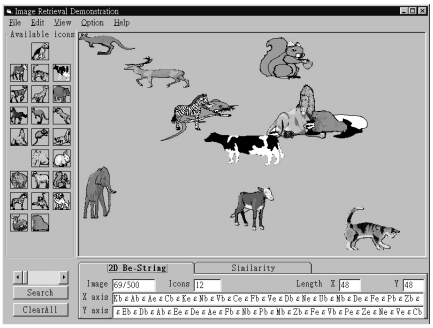

**Figure 8: Show the most similar image with 2D Bε-string.**

## 6. Conclusions and Future Work

A spatial representation model, called "2D Bε-string" is proposed. This model does not need to cut any icon in an image. Instead, an icon object is represented by its MBR boundaries. It depicts the spatial relationship between two boundary symbols by apply a 'dummy object'. It can intuitively represent the spatial relationship associated

with an image. In addition, the 2D Bε-string is formed by the object boundaries directly; thus, an image with $n$ objects has the space complexity $O(n)$.

We introduce an algorithm, named **Convert-2D-Be-String**, to convert an image with MBR coordinate into a symbolic image presented by the 2D Bε-string. The space and time complexity of this algorithm is $O(n)$. We also present a similarity retrieval algorithm, named **2D-Be-LCS-Length**, modified from the LCS algorithm, for the 2D Bε-string representation model. All the space and time complexities of our proposed algorithm are $O(mn)$.

Moreover, we provide an evaluation process, which can evaluate all similarity no matter how the matched LCS string appears or not all query objects or all spatial relationships. For the similarity retrieval of rotation and reflection, our approaches only need to reverse the string and apply the similarity retrieval and evaluation as mentioned above. This process does not need any conversion of spatial operators. It is more efficient and much easier than before.

In comparison with other 2D string methodologies, the 2D Bε-string simplifies the representation of spatial relationships and improves the efficiency of similarity retrieval. Even so, further research is still required.

At first, when an object area in an image differs very much from its MBR area, the similarity is not obvious no matter what to use, 2D Bε-string or other 2D strings. Due to the 2D Bε-string and other 2D strings obscure this information while abstracting. The distance between objects is obscured too. Therefore, how to clarify the size and distance information in the 2D Bε-string representation model is worthy studying.

In addition, we can evaluate the similarity more accurate. The original LCS algorithm does not calculate the number of LCS paths, neither do we here. For a query image from different images of database, even with the same length and content of LCS strings, conceptually, the more number of LCS paths the more similar is. Thus, we suggest further study to take account into the number of LCS paths in our similarity retrieval and evaluation approaches.

It is easy to expand the 2D Bε-string representation model and similarity assessment to retrieve objects in three-dimension case by adding the third dimension's string directly. We would like to integrate the temporal information into the video frame indexing in the near future.

## References


[1] A. Guttman, "R-tree: A Dynamic Index Structure for Spatial Searching", *Proc. ACM SIGMOD Int'l Conf. On the Management of Data*, 1984

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

[3] S. K. Chang, E. Jungert and Y. Li, "Representation and Retrieval of Symbolic Pictures Using Generalized 2D String", Technical Report, University of Pittsburgh, 1988.

[4] H. Samet, "Applications of Spatial Data Structures, Computer Graphics, Image Processing, and GIS." Addison-Wesley Publishing Company, 1989.

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, "Introduction to Algorithms", MIT Press, 1990, pp. 314-319.

[6] N. Beckmann, H. Kriegel, R. Schneider and B. Seeger, "The R*-tree: An Efficient and Robust Access Method for Points and Rectangles", *Proc. ACM SIGMOD Int'l Conf. On the Management of Data,* 1990

[7] S. Y. Lee and F. J. Hsu, "2D C-string: a New Spatial Knowledge Representation for Image Database Systems", *Pattern Recognition,* Vol. 23, 1990, pp. 1077-1087.

[8] S. Y. Lee, M. C. Yang and J. W. Chen, "2D B-string: a Spatial Knowledge Representation for Image Database Systems", *Proc. ICSC'92 Second Int. Computer Sci. Conf.,* 1992, pp. 609-615.

[9] W. Nibalck, R. Barber, W. Wquitz, M. Flickner, E. Glasman, D. P. P. Yanker, C. Faloutsos and G. Taubin, "The QBIC Project: Querying Images by Content Using Color, Texture and Shape", *SPIE*, 1908, 1993.

[10] P. W. Huang and Y. R. Jean, "Using 2D $C^+$-string as Spatial Knowledge Representation for Image Database Systems", *Pattern Recognition,* Vol. 27, No. 9, 1994, pp. 1249-1257.

[11] J. R. Bach, C. F. A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. Jain and C. Shu, "The Virage Image Search Engine: An Open Framework for Image Management", *SPIE*, 2670, 1996.

[12] En-Hui Liang, Sheau-Chuang Wu, "Similarity retrieval of image database based on decomposed objects", Graduate Institute of Information Management, Tamkang University, 1996

[13] En-Hui Liang, Duen-Liue Mou, "A method of computing spatial similarity between images", Graduate Institute of Information Management, Tamkang University, 1997

[14] B. C. Chien, "The Reasoning of Rotation and Reflection in Spatial Database", *Systems, Man, and Cybernetics*, 1998., 1998 IEEE International Conference, Vol. 2, 1998, pp. 1582–1586.

[15] Xiaobo Li and Xiaoqing Qu "Matching Spatial Relations Using DB-tree for Image Retrieval", *Pattern Recognition*, 1998., Proceedings. Fourteenth International Conference, Vol. 2, 1998, pp. 1230-1234.

[16] En-Hui Liang, Guo-Jang You, "Similarity retrieval using String Matching in Image Database Systems", Graduate Institute of Information Management, Tamkang University, 1999

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

[18] Michael Sipser, "Introduction to the Theory of Computation", PWS Publishing Company, 1997, pp. 245-253.