Image indexing and similarity retrieval based on A new Spatial Relation Model

Ying-Hong Wang
inhon@mail.tku.edu.tw
Department of Computer Science and Information Engineering,
TamKang University, Tamsui, Taipei Hsien, 25137, Taiwan, R.O.C.

Abstract

Spatial relation model is important technique for image indexing and retrieval in image or multimedia databases. 2-D strings and its variants are proposed to support the representation of spatial relationship. 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. By applying 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 for those images which partial of icons and/or spatial relationships are similar with the query image. It resolves the problems that the query targets and/or spatial relationships are not certain. Our representation model and similarity evaluation also simply the retrieval progress of linear transformations, include 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 existed in original images. For examples, how the image icons and their characteristics were recognized, how the symbolic image was encoded and constructed, how to index and retrieve these images, how to 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.], the Virage search engine [11.]; (2) by size and location of the image icons, such like R-tree [1.], R*-tree [6.], Quadtree [4.], and Mou's [13.]; (3) by relative position of the icons, such like the 2-D Strings [2.] and its variants [3.,7.,8.,10.,12.,15.,16.,17.]. The third is very suitable for those applications that do not care the acute coordinates of icon objects. For example, 'find all images which icon *A* locates at the left side and icon *B* locates at the right'.

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 straightly 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 $O(n)$ space complexity in the best and worst case, to build an image database using the model we proposed.

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 which all of the icons and their spatial relationships fully accord with the query image can be sifted out, but also for those images which partial of icons and/or spatial relationships are similar with the query image. It resolves the problems that the query targets and/or spatial relationships are not certain. The modified LCS algorithm takes $O(mn)$ space and time complexity, while m and n are the number of icons in a query image and a database image, respectively. It is more simply 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 x- 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 2D Bε-string. A similarity retrieval algorithm, modified from

0-7695-1080-9/01 $10.00 © 2001 IEEE

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 2D Bε-string and modified LCS algorithm. Finally, the concluding remarks and the future works are given in the last section.

2. Related works

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 use a symbolic projection of a picture along x- and y-axis. They define two sets of symbol, V and A. Each symbol in V presents an icon object in an image. A is a set of spatial operators contains {'=', '<', ':'}. A 2D string over V and A is defined by $(u, v) = (x_1y_1x_2y_2...y_{n-1}x_n, x_{p(1)}z_1x_{p(2)}z_2...z_{n-1}x_{p(n)})$, where $x_1x_2...x_n$ is 1-D string over V, $y_1y_2...y_{n-1}$ and $z_1z_2...z_{n-1}$ are 1-D string over A, $x_{p(1)}x_{p(2)}...x_{p(n)}$ is a permutation of $x_1x_2...x_n$.

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 set R_l defines local spatial relationships, that's the projection of two objects is partial overlap. The set R_g defines global spatial relationships, means that the projection of two objects either disjoin or adjoin or located at the same position. 2D G-string unifies the spatial relationship between two cut objects. Only those operators in 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 imporves the problem that there are superfluous cutting objects were generated at 2D G-string cutting progress. But there will be $O(n^2)$ cutting objects in the worst case.

Therefore, The 2D B-string [8.] drops the cutting process, instead, represents an object by two symbols. One stands the begin boundary of that object, another one for the end. 2D B-string also reduces the spatial relationships to single operator '='. It means that two objects have the same boundary projection if '=' is appeared.

The basic similarity retrieval and evaluation idea in 2-D Strings [2.], 2D G-string [3.], 2D C-string [7.] and 2D B-string [8.] are the same. First, they always define three type of similarity, type-i (i = 0, 1, 2). Each is constricted by some conditions. Type-1 is stricter then type-0, type-2 is stricter then type-1. Second, they examine all spatial relationship pairs between any two objects in query image versus pairs in an image of database. Buid 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 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. And finding maximum complete subgraph is an NP-complete problem [18.]. It is a time consuming work. It is not suitable for large number of icon objects in an image.

3. The Spatial Representation Model using 2D Bε-string

There are many approaches were proposed to represent an icon in an image. Such as MBR (Minimum Bounding Rectangle) [3.,7.,8.,10.], MBE (Minimum Bounding Ellipse), MBC (Minimum Bounding Circle) [13.], etc.

3.1 The model of 2D Bε-string

The approach used in 2D Bε-string is MBR too. Conceptually, it is similar with 2D B-string. They present an object by its MBR boundaries and need nothing to be cut. However, 2D Bε-string adopts a quite different idea to state the spatial relationship between two boundary symbols. 2D B-string uses a **spatial operator** (=) to describe the projection of two boundaries is **IDENTICAL**. In 2D Bε-string, we use a **dummy object** to describe the projection of two boundaries is **DISTINCT**!

We define a 'Dummy Object' as following:

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

Then the 2D Bε-string can be defined as

$$(u, v) = (d_0x_1d_1x_2d_2...d_{n-1}x_nd_n, d_0y_1d_1y_2d_2...d_{n-1}y_nd_n).$$

Where d_i is a dummy object ε or a null string, i = 0, 1, ... n, and x_i and y_i are real icon objects, they are either begin or end projected boundaries on x- and y-axis, respectively, i = 1, 2, ... n. To determine d_i, we need to know the maximum size of an image, say X_{max} and Y_{max} for x- and y-axis respectively. Set d_0 to ε if there is a space between the begin boundary of the leftmost (bottommost) object and the left (bottom) edge of image. Similarly, set d_n to ε if there is an interval between the end boundary of the rightmost (topmost) object and the right (top) edge of 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, 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_0 was set to ε because there is a space before begin boundary of object A on x-axis. The dummy object d_6 was set to ε too because there is a space left behind object B on x-axis. But the dummy object d_3 was set to null string because the end boundary of object A and the begin boundary of object C are projected at the same location on x-axis. Similar case is appeared for the end boundary of object B and the begin boundary of object C on y-axis.

Observably, the 2D Bε-string has following advantages:

First, object location in original image and symbolic picture was mapped directly. There is no

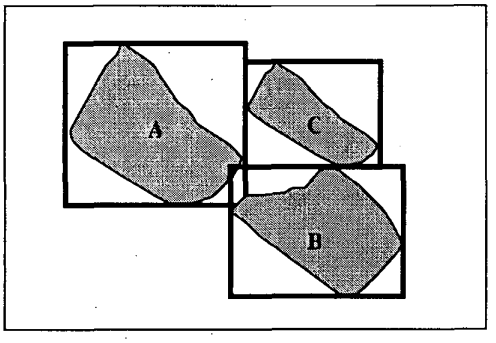

Figure 1: An image with three objects

operator required for representing spatial relationship between objects. It is intuitively.

Second, because it does not need to cut the objects of image. It simplifys the construction of image database. And the space complexity for an image with n objects in the worst and best case is O(n). For the worst case, all boundary projections are distinct and there is a space left in the leftmost, bottommost, rightmost and topmost of an image, it requires $4n+1$ symboles. By constrast, the best case, all boundary projections are identical and exactly fit in an image, it requires $2n+1$ symboles only.

Third, it simplifys the similarity retrieval because there is no combining the results of spatial reasoning required.

3.2 Algorithm for constructing a symbolic picture

By default, before converting an image to a symbolic picture represented by 2D Bε-string, we have abstracted all objects and their MBR coordinates from that image. Then call algorithm **Convert-2D-Be-String**, showed in Algorithm 1, to transform an original image to a symbolic picture. Lines 1-12 explain the meaning of variables used in input parameters and converting process. Lines 14-19 sort the input data by key coordinate and object identifier in ascending order for x- and y-axis separately. Lines 21-32 construct the 2D Bε-string on x-axis, and lines 34-45 construct the 2D Bε-string on y-axis.

The time complexity for algorithm **Convert-2D-Be-String** depends on the sorting algorithm at line 19. Time complexity on loops in line 14-18, 24-30 and 37-43 are $O(n)$, and never exceed the sorting algorithm. Ignore the sorting algorithm, the space complexity is $O(n)$ too.

Algorithm 1 to construct a symbolic picture:

```
Convert-2D-Be-String (n, C, Xb, Xe, Yb, Ye, xmax, ymax)
1.   // n ...number of objects in an image
2.   // C ...object symbols in image, C={c1, c2, ..., cn}
3.   // Xb ...begin boundaries on x-axis, Xb={xb1, xb2, ..., xbn}
4.   // Xe ...end boundaries on x-axis, Xe={xe1, xe2, ..., xen}
5.   // Yb ...begin boundaries on y-axis, Yb={yb1, yb2, ..., ybn}
6.   // Ye ...end boundaries on y-axis, Ye={ye1, ye2, ..., yen}
7.   // xmax ...maximum coordinate on x-axis
8.   // ymax ...maximum coordinate on y-axis
9.   // Xbe ...2D Bε-string on x-axis,
       Xbe={v0x1v1x2v2...vn-1xnvn}, where vi=dummy object ε
       or null string, i =0, 1, ..., n; xi=cbi or cei, identifies the
       projection of beging boundary or end boundary of
       object ci on x-axis, i =1, 2, ..., n
10.  // Ybe ...2D Bε-string on y-axis,
       Ybe={v0y1v1y2v2...vn-1ynvn}, where vi= dummy object ε
       or null string, i =0, 1, ..., n; yi=cbi or cei, identifies the
       projection of beging boundary or end boundary of
       object ci on y-axis, i =1, 2, ..., n
11.  // S...a sort work for x-axis, S={si | si=xbici or xeici, i=1,
       2, ..., 2n}
12.  // T...a sort work for y-axis, T={ti | ti=ybici or yeici, i=1,
       2, ..., 2n}
13.  // Combine MBR coordinate and object identifier as a
       key, sort the input data by ascending order
14.  for i=1 to n
15.     si←xbici
16.     si+n←xeici
17.     ti←ybici
18.     ti+n←yeici
19.  Sorting S and T by ascending order
20   // Construct 2D Bε-string on x-axis
21   Xbe←''                          // Initilized by a null string
22   if xb of s1≠0 then              // Insert ε at the leftmost?
23      Xbe←ε
24   for i=1 to 2n-1
25      if type of x in si is xb then  // convert coordinate to
26         Xbe←Xbecbi                  // boundary symbol
27      else                          // type of x in si is xe
28         Xbe←Xbecei
29      if x of si≠x of si+1 then
30         Xbe←Xbeε
31   if xe of s2n≠xmax then          // Insert ε at the rightmost?
```

```
32      X_be←X_beε
33    // Construct 2D Bε-string on y-axis
34    Y_be←''
35    if y_b of t_1≠0 then              // Insert ε at the
        bottommost?
36      Y_be←ε
37    for i=1 to 2n-1
38      if type of y in t_i is y_b then  // convert corrdinate to
39        Y_be←Y_beC_bi                  // boundary symbol
40      else                           // type of y in t_i is y_e
41      Y_be←Y_beC_ei
42      if y of t_i≠y of t_i+1 then
43        Y_be←Y_beε
44    if y_e of t_2n≠y_max then         // Insert ε at the topmost?
45      Y_be←Y_beε
46    return X_be, Y_be
```

While building an image database of 2D Bε-string, we only require to call algorithm **Convert-2D-Be-String** with the MBR coordinates and object identifiers for each image, and save the results, the 2D Bε-string, to database. Because the 2D Bε-string is an order data, if we save the 2D Bε-string with their MBR coordinates, we can easy find the location to be inserted for a new object and its MBR boundaries using binary search with key MBR coordinates and identifier of the new object. It is easy to determine whether insert a dummy object around this new object boundary or not. When we want to drop an object from a 2D Bε-string symbolic image, we search the dropping object sequentially, delete it directly and eliminate the redundant dummy object if it was found. Image similarity retrieval and evaluation

4. Assessment of Similarity

In section 2, we discuss the symbolic image construction and the assessment of similarity using 2D B-String. In order to decide which type, Type-0, Type-1 or Type-2, of similarity it belongs to, it is necessary to record all spatial relationship for every two objects in database image. However, this assessment does not consider the situation that only partial of query objects are exist in a database image. Moreover, the similarity assessment of Type-0, Type-1 and Type-2 is based on finding maximum complete subgraph. It is rather time-consumed to find a maximum complete subgraph from C_2^m spatial relationships of m objects in query image and C_2^n spatial relationships of n objects in database image.

In this paper, we also use the number of spatial relationships, formed by every two objects and appeared in query image and database image at the same time, as the similarity assesment. But, we do not find a maximum complete subgraph; instead, we find a longest common subsequence (LCS) [5.] between two 2D Bε-strings. And evaluate this LCS string with respect to 2D Bε-strings of query image and databade image.

The intention for finding a LCS string and finding a maximum complete subgraph between two images is similar. They are used to measure a similarity. But the time complexity of LCS algorithm is $O(mn)$, where m, n are the number of objects in query image and database image respectively. The complexity only depends on the length of 2D Bε-string in query image and database image. It is not necessary to examine all the spatial relationships for every two objects. Because "**The LCS string implies that, in query image and database image, all the spatial relationships of every two objects in LCS string are the same.**" So, the similarity can be evaluated in a reasonable time.

4.1 Algorithms of similarity retrieval

In this paper, we propose an algorithm to find a longest common subsequence length from two 2D Bε-strings. This algorithm is shown as follows and named as **2D-Be-LCS-Length**. This algorithm is modified from LCS algorithm in [5.]. There are two factors to revise the original LCS algorithm. The first one is, we avoid LCS from 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 does this evaluation. The second is, we omit the LCS paths recording matrix in original LCS algorithm by evaluating left and up paths first, line 16-19 show this, and evaluating left-up diagonal path next, as showed in line 23-24. The LCS path still can be inferred from the matrix record the LCS length.

Algorithm **2D-Be-LCS-Length** takes two 2D Bε-strings, Q and D, as input parameters, one for query image and the other one for database image. The 2D Bε-string of a query image with m objects along, each dimension has $2m$ boundary symbols and at most $2m+1$ dummy objects. The maximum length of 2D Bε-string in each dimension is $2m+(2m+1)$, that is, $4m+1$. With the same derivation, the maximum length of 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 $O(mn)$.

The initialization of first row and first column in LCS length inferring table W, as showed in line 7-8 and 10-11 of algorithm as follows, 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 showed in algorithm 3. The initial invocation is **Print-2D-Be-LCS** (*Q*, *W*, *length*(*Q*), *length*(*D*)). From the last cell of LCS 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.

Algorithm 2 to find LCS length from two strings:

```
2D-Be-LCS-Length (Q, D)
1.  m←length(Q)
2.  n←length(D)
3.  // Q is a 2D Bε-string of query image, Q = {q_i | q_i ∈ {dummy
        object ε or the boundary symbols of objects in query
        image}, i = 1, 2, ..., m}.
4.  // D is a 2D Bε-string of database image, D = {d_j | d_j ∈ {dummy
        object ε or the boundary symbols of objects in query
        image}, j = 1, 2, ..., n}.
5.  // LCS length is inferred in table W, W = {w_i,j | |w_i,j| is the LCS
        length of string q_1,...,q_i and d_1,...,d_j. If the last symbol of
        LCS string is dummy object ε then w_i,j < 0 else w_i,j ≥ 0,
        where i = 0, 1, 2, ..., m, j = 0, 1, 2, ..., n}.
6.  // Initilize the first column of inferring table W by zeros.
7.  for i←1 to m do
8.      w_i,0←0
9.  // Initilize the first row of inferring table W by zeros.
10. for j←0 to n do
11.     w_0,j←0
12. // From row 1 column 1, infer each cell, row by row and
        column by column, until every cell has been evaluated.
13. for i = 1 to m do
14.     for j = 1 to n do
15. // Set current cell value to the value of left or up cell which
        has maximum absolute 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. // Then check whether the symbol q_i, d_j are the same and at
        least one of q_i and the last symbol of LCS path from
        left-up diagnoal is not dummy object.
21.         if (q_i = d_j) and ((q_i ≠ ε) or (w_i-1,j-1 ≥ 0)) then
22. // If all are hold, then check whether follow the path from
        left-up diagnoal.
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
```

Due to the revision of finding a longest common subsequence length from two 2D Bε-strings before printing LCS string symbol, we need to compare the LCS string length of current cell with the string length of up one. It certainly implies that LCS path is 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 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 left cell's length. If they are the same, the LCS is induced from left direction. With the same reason as preceding case, we also ignore this symbol and continuously induce along left direction, as lines 7-8. If the LCS is not from up nor left direction, it must be induced from left-up diagonal direction. The current cell associated boundary symbol or dummy object must be part of LCS string. After processing all cells on the left/up direction recursively, we print out this symbol, as lines 9-11.

Algorithm 3 to print LCS string of two 2D Bε-strings:

```
Print-2D-Be-LCS (Q, W, i, j)
1.  // Q is a 2D Bε-string of query image, Q = {q_i | q_i ∈ {dummy
        object ε or the boundary symbols of objects in query
        image}, i = 1, 2, ..., m}.
2.  // W is the LCS length inferring table of two 2D Bε-strings,
        and induced from algorithm in Error! Reference source not
        found..
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
```

5. Conclusions and Future works

A spatial representation model is proposed, called 2D Bε-string spatial representation model. This model does not need to cut any icon object in an image, instead an object is represented by its MBR boundaries. It depicts the spatial relationship between two boundary symbols by apply a 'dummy object'. Thus, it can intuitively represent the spatial relationship in an image. In addition, the 2D Bε-string is form by the object boundaries directly; thus, an image with *n* objects only needs storage units between $2n$ and $4n+1$ in each dimension. That is, the space complexity of 2D Bε-string is $O(n)$.

We also introduce an algorithm, named **Convert-2D-Be-String**, to convert an image with MBR coordinate data into a symbolic image presented by 2D Bε-string. The space and time complexity of this algorithm is $O(n)$ when it does not consider the sort requirement.

We also present a similarity retrieval algorithm, named

2D-Be-LCS-Length, which is modified from LCS algorithm for the 2D Be-string representation model. All the space and time complexities of our proposed algorithm are $O(mn)$, where m is the number of icon objects in query image and n is the number of object icons in the image database.

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

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 Effencifent and Robust Access Method for Points and Retangles", *Proce. 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 Engin: 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, "Introcuction to the Theory of Computation", PWS Publishing Company, 1997, pp. 245-253.