SIMILARITY RETRIEVAL OF ICONIC IMAGE DATABASE*

SUIH-YIN LEE,† MAN-KWAN SHAN‡ and WEI-PANG YANG‡

†Institute of Computer Engineering, National Chiao Tung University, Hsinchu, Taiwan 30039, R.O.C.; and ‡Institute of Information Science, National Chiao Tung University, Hsinchu, Taiwan 30039, R.O.C.

(Received 26 July 1988; in revised form 23 November 1988; received for publication 8 December 1988)

Abstract—The perception of spatial relationships among objects in a picture is one of the important selection criteria to discriminate and retrieve the images in an iconic image database system. The data structure called 2D string, proposed by Chang et al., is adopted to represent symbolic pictures. The 2D string preserves the objects’ spatial knowledge embedded in images. Since spatial relationship is a fuzzy concept, the capability of similarity retrieval for the retrieval by subpicture is essential. In this paper, similarity measure based on 2D string longest common subsequence is defined. The algorithm for similarity retrieval is also proposed. Similarity retrieval provides the iconic image database with the distinguishing function different from a conventional database.

Image database Spatial relationship Similarity retrieval 2D string
Longest common subsequence Maximal complete subgraph

1. INTRODUCTION

Recently much attention has been paid to the design of image databases. When retrieving images from a database, one of the most important methods for discriminating among the images is the perception of the objects and the spatial relationships that exist among these objects in the desired image. The capability of assembling queries on the objects and their spatial relationships becomes an important issue of image database design. Most systems of previous approaches provide search capability of simple table look-up of image features and secondary information. Only the pictorial database using Packed R-tree and Intelligent Image Database Systems(IIDS) provide more advanced retrieval capability. Their most important characteristic is that they provide high level object-oriented search rather than search based on the low level image primitives of objects. Above all, the IIDS supports spatial reasoning, flexible image information retrieval, visualization, and traditional image operators. The spatial reasoning is based on a new pictorial data structure, a 2D string, which preserves the objects’ spatial knowledge embedded in images. A picture query can also be specified as a 2D string. The problem of pictorial information retrieval then becomes a problem of 2D subsequence matching. This approach allows an efficient and natural way to construct iconic indexes for pictures.

To retrieve the images according to the spatial relationships, one problem may arise. Spatial relationship is a fuzzy concept and is thus often dependent on human interpretation. Also, the generation of the 2D string is sensitive to the shape, size, and relative position of the objects in the image. Thus, similarity retrieval of images, which is one of the distinguishing functions different from a conventional database, is a necessity.

In this paper, the algorithm for similarity retrieval, based on the 2D string longest common subsequence, is proposed. We first review the 2D string approach of representing symbolic pictures. Section 3 presents the similarity retrieval based on the 2D string longest common subsequence. The conclusions and suggestions for future development are stated in the last section.

2. 2D STRING APPROACH

The approach of iconic indexing by 2D string for spatial query was proposed by Chang et al. First, after preprocessing by image processing and pattern recognition techniques, the objects in the original image are recognized. Then for each object, the objects in an orthogonal relation with respect to other objects are generated. The original picture can be regarded as a symbolic picture. At last, the symbolic picture, which preserves the spatial relationships among objects of the original image, is encoded as a 2D string. The picture query can also be specified as a 2D string. The problem of pictorial information retrieval thus becomes a problem of 2D subsequence matching.

Our concern is how to characterize the spatial relationships of the non-zero sized objects in the symbolic picture. Chang et al. have proposed a method called orthogonal relations to solve this
problem. First, all the objects are enclosed by the minimum enclosing rectangles (MER). In terms of enclosing rectangles of objects, three types of spatial relations among objects can be identified. These are for objects with:

1. nonoverlapping rectangles;
2. partially overlapping rectangles;
3. completely overlapping rectangles.

The case with nonoverlapping rectangles will never cause any problems in describing their mutual spatial relations. The other two cases might sometimes cause problems. The basic idea of orthogonal relations is to regard one of the objects as a “point of view” object (PVO) and then to view the other object in four directions (below, above, left, right). Hence, at least one, or at most four subparts of the other object can be seen from the PVO. The part of the object that actually is seen is in the intersection where the two rectangles overlap, partially or completely. The subobjects segmented by PVO are called ortho-relational objects of the original object. After all the ortho-relational objects have been generated, the objects then can be segmented. The reference point of each segment is the centroid of each orthogonal relation object. All the reference points thus will dominate the spatial relations of objects and will constitute a symbolic picture. The symbolic picture is then converted to the 2D string representation. In the following, the definitions of symbolic pictures and 2D strings are described.

A symbolic picture \( f \) is a mapping \( M \times N \rightarrow W \), where \( M, N \in \{1,2,\ldots,m\} \) are the spatial locations in \( x \)-direction and \( y \)-direction, respectively, \( m \) is the picture size, \( V \) is the set of symbols, and \( W \) is the power set of \( V \). The empty set \( \{\} \) then denotes a null object.

In Fig. 1, from the symbolic picture \( f \), we can see the spatial relationships among the objects. For instance, object \( D \) is on the right of object \( A \) and object \( C \) is on the upper-right of object \( A \).

A 1D string over \( V \) is any string \( v_1 \ldots v_n, n \geq 0 \), where \( v_i \)’s are in \( V \). A 2D string \( (\gamma_x, \gamma_y) \) over \( V \) is defined as

\[
(\gamma_1 f_1 \gamma_2 f_2 \ldots \gamma_{n-1} f_{n-1} f_n, \gamma_1' f_1' f_2' \ldots \gamma'_{n-1} f'_{n-1} f'_n)
\]

where

\[v_1 \ldots v_n \text{ is a 1D string over } V,
\]

\[
p: \{1,\ldots,n\} \rightarrow \{1,\ldots,n\} \text{ is a permutation over } \{1,\ldots,n\},
\]

\[
r_{1x} \ldots r_{(n-1)x} \text{ is a 1D string over } R,
\]

\[
r_{1y} \ldots r_{(n-1)y} \text{ is a 1D string over } R.
\]

As an example, the symbolic picture \( f \) shown in Fig. 2 may be represented as 2D string \((A < B: C = D < E, B: C < A < DE)\). But sometimes, the omission may cause ambiguity if the symbolic picture is reconstructed from the 2D string. For example, given the 2D string \((A < D < BC < A, A < DB < C < A)\), two different pictures can be reconstructed.

Since the symbolic picture is represented by the 2D string, a picture query can also be specified as a 2D string. The problem of pictorial information retrieval then becomes the problem of 2D subsequence matching. Chang et al. (11) defined type-0, type-1 and type-2 2D subsequence as follows.

Definition 1. A string \( \gamma \) is a type-i 1D subsequence of string \( x \), if (1) \( \gamma \) is contained in \( x \), and (2) if \( a_1w_1b_1 \) is a substring of \( \gamma \), \( a_1 \) matches \( a_2 \) in \( x \) and \( b_1 \) matches \( b_2 \) in \( x \), then

\begin{align*}
\text{(type-0)} & \ r(b_2) - r(a_2) \geq r(b_1) - r(a_1) \\
\text{or } & \ r(b_2) - r(a_2) = 0 \\
\text{(type-1)} & \ r(b_2) - r(a_2) \geq r(b_1) - r(a_1) > 0 \\
\text{or } & \ r(b_2) - r(a_2) = r(b_1) - r(a_1) = 0 \\
\text{(type-2)} & \ r(b_2) - r(a_2) = r(b_1) - r(a_1)
\end{align*}

where \( r(x) \), the rank of symbol \( x \), is defined as one plus the number of “<” preceding this symbol \( x \).
Definition 2. Let \((x, x, z_p)\) and \((x, x, z_p)\) be the 2D string representations of symbolic pictures \(f\) and \(f'\), respectively. \((x, x, z_p)\) is a type-1 2D subsequence of \((x, x, z_p)\) if \(z_p\) is a type-1 1D subsequence of \(x\) and \(z_p\) is a type-1 1D subsequence of \(x\). \(f'\) is said to be a type-1 subpicture of \(f\).

The 2D string representations for \(f\), \(f_1\), \(f_2\), \(f_3\) in Fig. 3 are

\[
f: (\overline{A < B < C}, \overline{A < B < C}, \overline{A < B < D}),
\]
\[
f_1: (\overline{A < B}, \overline{A < B}, \overline{A < B, z_2}),
\]
\[
f_2: (\overline{A < C}, \overline{A < C}, \overline{A < C}),
\]
\[
f_3: (\overline{AB < C}, \overline{A < BC < D}).
\]

Fig. 3. Picture matching example\(^{11}\).

Therefore, to determine whether a picture \(f'\) is a type-1 subpicture of \(f\), we need only to determine whether \((x, x, z_p)\) is a type-1 2D subsequence of \((x, x, z_p)\). The picture matching problem thus becomes a 2D string matching problem. The query can be specified graphically by drawing an iconic picture on the screen. The graphical representation can then be translated into the 2D string representation. The query, represented by a 2D string, is matched against the iconic index which is the 2D string representation of a picture in the image database. Those objects the symbols of which match the query 2D string are retrieved.\(^{11}\)

3. SIMILARITY RETRIEVAL

Similarity retrieval is one of the functions of the image database that distinguishes it from a conventional database. The objective is to retrieve the images that are similar to the query image.

The similarity between two patterns or objects can be measured on the basis of the maximum-likelihood or minimum-distance criterion. The similarity between 1D strings based on the minimum-distance criterion has been developed in the technique of syntactic pattern recognition.\(^5\) The distance between two strings is defined in terms of the minimum number of error transformations used to derive one from the other. The similarity between two 1D strings based on the maximum-likelihood is defined in terms of the longest common subsequence between two strings.\(^{12}\)

Since the picture query is processed as the 2D subsequence, we adopt the maximum-likelihood criterion to measure the similarity. Analogous to the type-1 2D subsequence, we define three types of 2D string similarity measure based on the 2D string longest common subsequence.

Definition 3. A 2D string \((x, x, z_p)\) is a common 2D subsequence of two 2D strings \((x, x, z_p)\) and \((x, x, z_p)\) if \((x, x, z_p)\) is a 2D subsequence of \((x, x, z_p)\) and also a 2D subsequence of \((x, x, z_p)\).

Definition 4. A 2D string \((x, x, z_p)\) is a 2D string longest common subsequence (LCS) of 2D strings \((x, x, z_p)\) and \((x, x, z_p)\) if \((x, x, z_p)\) is a common 2D subsequence of maximal length. The length of \((x, x, z_p)\) is defined to be the length of the type-i similarity measure between 2D strings \((x, x, z_p)\) and \((x, x, z_p)\).

Thus, the type-i similarity retrieval is to retrieve the most similar picture the type-i similarity of which is the longest among all the pictures stored in the image database. Some efficient algorithms for the 1D string LCS have been developed.\(^{12}\) The 2D string LCS is somewhat similar to the 1D string LCS, but is more complicated. We first review the 1D string LCS.

Definition 5. String \(\gamma = s_1 s_2 \ldots s_n \) is a subsequence of string \(\alpha = a_1 a_2 \ldots a_m\) if there exists a monotonic function \(F: \{1, 2, \ldots, n\} \rightarrow \{1, 2, \ldots, m\}\) such that \(F(i) = k\) only if \(s_i = a_k\). The monotonic function means that if \(F(i) = k, F(j) = l\), and \(i < j\), then \(k < l\).

In the definition above, the indices \(i\) of \(s_i\) and \(k\) of \(a_i\) can be regarded as the ranks of the symbols \(s_i\) and \(a_i\) respectively, where the rank of symbol \(x\) is the spatial location of symbol \(x\). The monotonic property of the function of a sequence can be viewed as the relative sequencing, with respect to the same 1D direction, between each pair of symbols in \(x\) is the same as that in \(x\).

Definition 6. String \(\gamma\) is a common subsequence of strings \(\alpha\) and \(\beta\) if \(\gamma\) is a subsequence of string \(\alpha\) and also a subsequence of string \(\beta\).

Definition 7. String \(\gamma\) is a longest common subsequence of strings \(\alpha\) and \(\beta\) if \(\gamma\) is a common subsequence of \(\alpha\) and \(\beta\) of maximal length.

For example, string \(\gamma = \text{"LORT"}\) is a subsequence of string \(\alpha = \text{"ALGORITHM"}\). Because there exists a monotonic function \(F\) such that \(F(1) = 2, F(2) = 4, F(3) = 5, F(4) = 7\), where \(s_1 = a_1 = \text{"L"}, s_2 = a_4 = \text{"O"}, s_3 = a_5 = \text{"R"}, s_4 = a_7 = \text{"T"}\). String \(\gamma\) is also a subsequence of string \(\beta = \text{"BELOWART"}\).

The common subsequences of \(\alpha\) and \(\beta\) are \("LOR", "LRT", "LRT", "ORT", "OTT", "RT", "AR", "LOR", "LORT", "ART", "LORT"\). The symbols of all the substrings have the same relative sequencing in \(\alpha\) and \(\beta\). \(Ls\) are left to \(O\)s, \(O\)s are left to \(R\)s, \(R\)s are left to \(T\s\) in both \(\alpha\) and \(\beta\). Because "LORT" is the common subsequence of strings \(\alpha\) and \(\beta\) of maximal length, "LORT" is the longest common subsequence of strings \(\alpha\) and \(\beta\).
To solve the 1D string longest common subsequence problem, there are brute force, dynamic programming, graphic model, geometric approaches and Four-Russian methods. We only describe the graphic model which we follow for the longest common 2D subsequence. The basic idea of the graphic model is the observation that a common subsequence is composed of only matched symbols. Given two 1D strings $\alpha = a_1a_2...a_m$ and $\beta = b_1b_2...b_n$, let entry $M_{ij}$ in the match table $M$ be "1" if $a_i = b_j$, and be "0" if $a_i \neq b_j$. Then only the entries where $M_{ij} = "1"$ can constitute the common subsequence. We call this type of entries matched entries, Now consider each pair of matched entries $M_{ij}$, $M_{ip}$, $M_{pj}$ and $M_{pq}$ can produce a common subsequence if and only if $(i - s)(j - t) > 0$. This is due to the monotonic property.

Consider the given 1D strings $\alpha = "ABCBA"$ and $\beta = "CBACB"$. The match table $M$ in Fig. 4 is computed first. Then only the point "@"s in the table where $M_{ij} = "1"$ may constitute the common subsequence. For each pair of symbol "@"s, only if one of the "@"s is on the lower-right of the other "@", there exists a corresponding common subsequence. There are other cases in which improperly selected "@"s do not correspond to any common subsequence. For example, let entry $M_{it}$ in the x-direction match table be "I" if $a_i = b_t$, and be "0" elsewhere. Then the 2D common subsequence can be constructed from the matched entries. Note that the number of matched entries in $M'$ must be equal to that in $M$.

Now if two dummy vertices $(0, 0)$ and $(\infty, \infty)$ are added, then the longest common subsequence problem may be transformed into the problem of finding a longest path from $(0, 0)$ to $(\infty, \infty)$ on the directed graph. The algorithm of topological sorting in scheduling can be used and takes $O(n \log n)$ time complexity, where $n$ is the number of vertices in the directed graph.

The algorithm for the 2D string LCS is somewhat similar to that of 1D string LCS. To simplify the problem of 2D string LCS, we first consider the case in which each spatial location contains only one symbol. In this case, the spatial relation symbols "<" and "=" of the 2D string can be deleted without affecting the symbolic picture for which the 2D string stands. We call this type of 2D string simple 2D string. For example, the simple 2D string of the 2D string $(A < B < C < D, C < A < D < B, 3142)$ is $(ABC, CADB, 3142)$. Then we define the simple 2D subsequence.

**Definition 8.** A simple 2D string $(x, y, \gamma_p) = (v_1v_2...v_n, v_{p(1)}v_{p(2)}...v_{p(m)}, p(1)p(2)...p(n))$ is a simple 2D subsequence of 2D string $(x, x, x_p) = (a_1a_2...a_m, a_{q(1)}a_{q(2)}...a_{q(m)}, q(1)q(2)...q(m))$, $m \geq n$, if there exist two monotonic functions, $F_1: \{1, 2, ..., n\} \rightarrow \{1, 2, ..., m\}$, and $F_2: \{p(1), p(2), ..., p(n)\} \rightarrow \{q(1), q(2), ..., q(m)\}$ such that $F_1(i) = j$ only if $a_i = a_j$, $F_2(p(i)) = q(j)$ only if $v_{p(i)} = v_{q(j)}$. Here, the monotonic function means that if function $F(i) = u, F(j) = v$, and $i < j$, then $u < v$.

For example, let $(AC, CA, 21)$ and $(ACB, CBA, 231)$ be two simple 2D strings where $v_1 = A, v_2 = C, a_1 = A, a_2 = A, a_3 = B, v_{p(1)} = C, v_{p(2)} = A, a_{q(1)} = C, a_{q(2)} = B, a_{q(3)} = A$. There exist two monotonic functions $F_1$ and $F_2$ such that $F_1(1) = 1, F_1(2) = 2, F_2(p(1)) = q(1), and F_2(p(2)) = q(3)$. $(AC, CA)$ is a 2D subsequence of $(ACB, CBA)$. The definitions of simple 2D common subsequence and simple 2D string longest common subsequence are analogous to those of 2D string in definitions 3 and 4.

How may one solve the problem of simple 2D string LCS? We follow the graph model approach of the 1D string LCS. Similar to the observation of the graph model for the 1D string LCS, only matched symbols can constitute the 2D common subsequence. But for each pair of symbols, there are two indices, x-index and y-index to consider in constructing a common subsequence.

Consider two given simple 2D strings $(x, x, x_p) = (a_1a_2...a_m, a_{q(1)}a_{q(2)}...a_{q(m)}, q(1)q(2)...q(m))$, and $(\beta, \beta, \beta_p) = (b_1b_2...b_n, b_{q(1)}b_{q(2)}...b_{q(m)}, q(1)q(2)...q(m))$. Let entry $M_{ij}$ in the x-direction match table be "1" if $a_i = b_j$, and "0" elsewhere. Also, let entry $M_{kl}$ in the y-direction match table be "1" if $a_{q(k)} = b_{q(l)}$, and "0" otherwise. Then the 2D common subsequence can be constructed from the matched entries. Note that the number of matched entries in $M'$ must be equal to that in $M$.

The next step is to decide which pairs of matched
entries correspond to the common subsequence. Similar to the 1D string LCS, we check each pair of matched entries in the ascending order. $M^x_i$ and $M^y_i$ correspond to a 1D (in the x-direction) common subsequence if $(i - s) * (j - t) > 0$. This is due to the monotonic property of function $F_1$ in definition 8. But, we must also check the y-direction. Let $k, l, u, v$ be the index values such that $p(k) = i, q(l) = j, p(u) = s, q(v) = t$. In fact, $k, l, u, v$ are the index values of $a_i, b_j, a_s, b_t$ in the y-strings, respectively. Then $M^x_i$ and $M^y_i$ correspond to a 1D (in the y-direction) common subsequence if $(k - u) * (l - v) > 0$. This is also due to the monotonic property of function $F_2$. So, in general, let $a_i = b_j = \varnothing_1 = a_p(k), b_j = a_q(l), a_i = b_t = a_p(u), a_s = b_t = a_p(v)$.

The simple 2D string LCS described in graph-theoretic terms is given as follows.

1. $(i, j, k, l)$ is a vertex in the graph if $a_i = b_j$, where $i, j$ are the index values of the $i$th, $j$th symbols of the $x$-strings of the 2D strings $(\alpha_x, \gamma_x, \gamma_p)$ and $(\beta_x, \gamma_y, \gamma_p)$, $k, l$ are the index values in the $y$-strings such that $p(k) = i, q(l) = j$.

2. Let $(i, j, k, l), (s, t, u, v)$ be two vertices. There is an edge between $(i, j, k, l)$ and $(s, t, u, v)$ if and only if $(i - s) * (j - t) > 0$, and $(k - u) * (l - v) > 0$.

3. A common subsequence is a set of vertices in which each pair of the vertices has an edge between them. In other words, each complete subgraph corresponds to a common subsequence.

4. The simple 2D string LCS is a complete subgraph containing the maximal number of symbols, i.e. the maximal complete subgraph (clique).

Notice the major distinctions between the graph models of 1D and 2D subsequences. Both the semantic interpretations of the 1D LCS and the 2D LCS can be viewed as finding the maximal number of symbols with same relative sequencing. The only exception is that the relative sequencing of 1D string is one dimensional, whereas that of 2D string is two dimensional. The 1D subsequence possesses the transitive property. For example, if symbol $X$ is left to symbol $Y$, and $Y$ is left to symbol $Z$, then $X$ is left to $Z$. So, the dynamic property could be utilized to solve the 1D string LCS. But the transitive property does not hold for the 2D subsequence. For example, if symbol $X$ is north-eastern to symbol $Y$ and $Y$ is north-western to symbol $Z$, we cannot tell whether $X$ is north-eastern or north-western to $Z$. The absence of transitivity of 2D strings makes the problem of 2D string LCS more complicated. The algorithm for the simple 2D string LCS is proposed as below.

Algorithm: simple 2D string LCS 
input: Two simple augmented 2D strings $(\alpha_x, \gamma_x, \gamma_p)$ and $(\beta_x, \gamma_y, \gamma_p)$
output: The simple 2D string LCS $(\gamma_x, \gamma_y, \gamma_p)$

begin

$M =$ length of the string $\alpha_x$
$N =$ length of the string $\beta_x$
for $i := 1$ to $M$
do
find the index value $k$ such that $p(k) = i$
end for

for $j := 1$ to $N$
do
find the index value $l$ such that $q(l) = j$
end for

for $i := 1$ to $M$
do
for $j := 1$ to $N$
do
if the $i$th symbol $a_i$ of $\alpha_x$ = the $j$th symbol $b_j$ of $\beta_x$
then record as a vertex $(i, j, k, l)$
end for
end for

for each pair of vertices $(i, j, k, l), (s, t, u, v)$
if $(i - s) * (j - t) > 0$ and $(k - u) * (l - v) > 0$
then construct an edge between them
do
for
for each pair of vertices $(i, j, k, l), (s, t, u, v)$
if $(i - s) * (j - t) > 0$ and $(k - u) * (l - v) > 0$
then find the maximal complete subgraphs (cliques) of this undirected graph
output $(\gamma_x, \gamma_y, \gamma_p)$ by deleting those symbols which do not constitute the clique
end for
end for

end-for
end-for

find the maximal complete subgraphs (cliques) of this undirected graph
output $(\gamma_x, \gamma_y, \gamma_p)$ by deleting those symbols which do not constitute the clique
end for
end-for

end of Algorithm Simple-2D-Strings-LCS

The algorithm for generating the maximal complete subgraphs (cliques) developed by Bronn is shown below. It is expressed as a recursive program.

Algorithm: generating the clique 
input: An undirected graph $G = (V, E)$, where $V$ is the set of vertexes, $E$ is the set of edges
output: A set $S$ of vertexes constituting a clique

begin
initialize $S$ as an empty set
call subroutine clique $(V, \phi)$
end

subroutine clique $(N, D)$
begin
if $N \cap D = \phi$
then output $S$, which is a clique
else
if $N \neq \phi$
then select one vertex $f$ from $N$
call subroutine explore $(f)$
while $N \cap (V - \text{Adj}(f)) \neq \phi$
do
$v$ is vertex in $N \cap (V - \text{Adj}(f))$
call subroutine explore $(v)$
end-while
end-if
end-if
end-of-subroutine-explore

subroutine explore $(u)$
begin
let $N$ be $N - \{u\}$
let $S$ be $S \cup \{u\}$
call subroutine clique $(N \cap \text{Adj}(u), D \cap \text{Adj}(u))$
let $S$ be $S - \{u\}$
let $D$ be $D \cup \{u\}$
end-of-subroutine-explore
end-of-Algorithm Generating-the-Clique
Having solved the simple 2D string LCS, the type-i 2D string LCS can also be solved. It is noted that the index of each symbol can be regarded as the rank of each symbol. To solve the type-i 2D string LCS, only the condition for connecting the edges of the undirected graph need to be modified. The major difference between algorithms simple-2D-string-LCS and 2D-string-LCS lies in the condition for constructing the edges.

**Algorithm: 2D-string-LCS**

**Input:** Two 2D strings ($\beta_x, \beta_y, \beta_p$) and ($\beta_x, \beta_y, \beta_p$)  
**Output:** The 2D string LCS ($\gamma_x, \gamma_y, \gamma_p$)

begin  
\( M = \text{length of the string } \beta_x \)  
\( N = \text{length of the string } \beta_y \)  
for \( i = 1 \) to \( M \) do  
\( \) find the index value \( k \) such that \( p(k) = i \)  
end for  
for \( j = 1 \) to \( N \) do  
\( \) find the index value \( l \) such that \( q(l) = j \)  
end for  
for \( i = 1 \) to \( M \) do  
\( \) for \( j = 1 \) to \( N \) do  
\( \) if the \( i \)-th symbol \( a_i \) of \( \beta_x \) matches the \( j \)-th symbol \( b_j \) of \( \beta_y \)  
\( \) record as a vertex \((i, j, k, l)\)  
end for  
end for  
for each pair of the vertexes \((i, j, k, l), (s, t, u, v)\)  
\( \) \(c_1 = \text{rank}(s) - \text{rank}(i)\)  
\( \) \(c_2 = \text{rank}(t) - \text{rank}(j)\)  
\( \) \(c_3 = \text{rank}(u) - \text{rank}(k)\)  
\( \) \(c_4 = \text{rank}(v) - \text{rank}(l)\)  
end-case
end-for  
for each pair of the vertexes \((i, j, k, l), (s, t, u, v)\)  
\( \) \(\text{case}\)  
\( \) \(\text{type-0: if } (c_1 \cdot c_2 \geq 0) \) and \( (c_3 \cdot c_4 \geq 0)\)  
\( \) then connect the edge \(\{ij\}\)  
end-case  
for each pair of the vertexes \((i, j, k, l), (s, t, u, v)\)  
\( \) \(\text{case}\)  
\( \) \(\text{type-1: if } ((c_1 \cdot c_2 > 0) \) or \( (c_1 = 0 \) and \( c_2 = 0)\) \) and \( (c_3 \cdot c_4 > 0) \) or \( (c_3 = 0 \) and \( c_4 = 0)\)  
\( \) then connect the edge \(\{ij\}\)  
end-case  
for each pair of the vertexes \((i, j, k, l), (s, t, u, v)\)  
\( \) \(\text{case}\)  
\( \) \(\text{type-2: if } (c_1 = c_2) \) and \( (c_3 = c_4)\)  
\( \) then connect the edge \(\{ij\}\)  
end-case
end-for  
find the maximal complete subgraphs (cliques) of this undirected graph  
output \((\gamma_x, \gamma_y, \gamma_p)\) by deleting those symbols which do not constitute the clique  
end-of-Algorithm

Fig. 5 shows two sample symbolic pictures \(f_1, f_2\) for 2D LCS. 

(2) Find the x-rank and y-rank of each symbol in \((\alpha_x, \alpha_y, \alpha_p)\) and \((\beta_x, \beta_y, \beta_p)\). Expressed as rank strings are \((122345, 425341)\) and \((122345, 253214)\). 

(3) Then construct the match tables in which each entry represents the vertex. Listed below are the vertices with each vertex associated with the symbol and its x-ranks and y-ranks: \((A, 1, 2, 4, 3), (B, 2, 3, 3, 1), (C, 3, 1, 3, 5), (D, 5, 5, 4, 3), (E, 4, 4, 4, 1), (F, 5, 5, 4, 2)\). The match tables for the x-coordinate and y-coordinate are shown in Fig. 6(a) and Fig. 6(b). 

(4) For each pair of the vertices, check the condition for connecting the edge

**Type-0:** \(AB, AG, AF, BC, BF, BG, DE, FG\).  
(Fig. 7)

**Type-1:** \(AB, AF, BF, DE\). (Fig. 8)

**Type-2:** \(AB\). (Fig. 9).

(5) Find the maximal complete subgraphs.

Type-0: edges \(AB, AG, AF, BC, BF, BG, DE, FG\) constitute the clique. Edge \(DE\) constitutes another clique. So, the most similar subpicture consists of symbols \(A, B, G, F\).

Type-1: edges \(AB, AF, BF\) constitute the clique.

**Fig. 6(a).** X-coordinate match table \(M^x\) of \(f_1, f_2\).  

**Fig. 6(b).** Y-coordinate match table \(M^y\) of \(f_1, f_2\).
Similarity retrieval provides an iconic image database with the distinguishing function different from a conventional database. In this paper, the similarity measure for symbolic pictures based on the 2D string longest common subsequence is defined. The algorithm to compute the 2D string longest common subsequence is also proposed. The 2D string LCS problem is transformed to the maximal complete subgraph (clique) problem. The clique problem takes nonpolynomial time complexity, same is the time complexity for the problem of 2D string LCS. This is due to the absence of the transitive property of spatial relationships among 2D objects. More efficient algorithms for similarity retrieval will be the subject of future research. At the same time, we note that the performance of the 2D string iconic indexing depends on the abstraction from segmented images to symbolic pictures. More powerful iconic indexing and abstraction techniques for symbolic representation are worth further study.

Acknowledgements—The authors would like to express their gratitude to the referees whose valuable comments and suggestions led to considerable improvement in the presentation of this material. Special thanks are extended to Professor S. K. Chang of University of Pittsburgh for his inspiration and providing some illustrations.

REFERENCES


About the Author—SUH-YIN LEE was born on 1 August 1950 in Taiwan, R.O.C. She received B.S.E.E. degree from National Chiao Tung University, Taiwan in 1972, and M.S. degree in Computer Science from University of Washington, Seattle Washington in 1975. During 1975–1976, she enrolled as a Ph.D. student in the Computer Science Department of University of Illinois, Urbana–Champaign Illinois. She joined the faculty of the Department of Computer Engineering at National Chiao Tung University in 1976, where she continued her Ph.D. program and received the Ph.D. degree in Electronic Engineering in 1982. She is now an associate professor in the Department of Computer Engineering of Chiao Tung University. Her current research interests include image database, office automation, and computer network. Dr Lee is a member of Phi Tau Phi.

About the Author—MAN-KWAN SHAN was born on 7 March 1963 in Taichung, Taiwan, R.O.C. He received the B.S. degree in computer engineering and the M.S. degree in information science both from National Chiao Tung University, Hsinchu, Taiwan in 1986 and 1988, respectively. His current research interests are in the areas of database management system, pattern recognition, and computational geometry.

About the Author—WEI-PANG YANG was born on 17 May 1950 in Hualien, Taiwan, R.O.C. He received the B.S. degree in Mathematics from National Taiwan Normal University in 1974, and the M.S. and Ph.D. degrees from National Chiao Tung University in 1979 and 1984, respectively, both in Computer Engineering. Since 1979, he has been on the faculty of the Department of Computer Engineering at National Chiao Tung University, Hsinchu, Taiwan. In the academic year 1985–1986 he was a visiting scholar at Harvard University. From 1986 to 1987 he was the Director of computer center of National Chiao Tung University. In August 1988 he joined the Department of Information Science of the same University and currently acts as the Head of the department. His research interests include database theory, extensible database systems, and expert database systems. Dr Yang is an associate editor of Computer Quarterly, and a member of IEEE, ACM and Phi Tau Phi society. He was elected an Outstanding Instructor in the College of Engineering of National Chiao Tung University in 1988. He was the winner of the 1988 Acer Long Term Award for outstanding M.S. Thesis Supervision.