JOURNAL OF VISUAL COMMUNICATION AND IMAGE REPRESENTATION
Vol. 9, No. 1, March, pp. 87–100, 1998
ARTICLE NO. VC980377

Similarity Retrieval by 2D C-Trees Matching in Image Databases

Fang-Jung Hsu* and Suh-Yin Lee

Institute of Computer Science and Information Engineering, National Chiao Tung University, HsinChu, Taiwan

and

Bao-Shuh Lin

Computer & Communication Research Laboratories, Industrial Technology Research Institute, HsinChu, Taiwan

Received June 18, 1997; accepted February 26, 1998

The image retrieval based on spatial content is an attracting task in many image database applications. The 2D strings provide a natural way of constructing spatial indexing for images and support effective picture query. Nevertheless, the 2D string is deficient in describing the spatial knowledge of nonzero sized objects with overlapping. In this paper, we use an ordered labeled tree, a 2D C-tree, to be the spatial representation for images and propose the tree-matching algorithm for similarity retrieval. The distance between 2D C-trees is used to measure the similarity of images. The proposed tree comparison algorithm is also modified to compute the partial tree distance for subpicture query. Experimental results for verifying the effectiveness of similarity retrieval by 2D C-trees matching are presented. © 1998 Academic Press

1. INTRODUCTION

Content-based image retrieval plays a principal activity in many application areas, such as picture archiving and communication systems, geographic information systems, biomedical, education, and home entertainment systems [1]. In a content-based image retrieval system, it is required to effectively and efficiently retrieve information from the image repositories. Such a system helps users retrieve relevant images based on their content [8, 20]. Current approaches [6, 9, 18, 21] to content-based image retrieval differ in terms of image features extracted, the level of abstraction manifested in the features, and the desired degree of domain independence. There are two major categories of features: primitive and logical. Primitive, or low level, image features such as object colors [6] and boundaries can be extracted automatically or semi-automatically. Logical features are abstract representations of images at various levels of detail. Some logical features such as spatial-location and spatial-relation [18] may be synthesized from primitive features, whereas others can only be obtained through considerable human involvement. Spatial constraint is a significant logical feature and is our focus in this article.

The intelligent image database system (IIDS) [4] provides high-level object-oriented search and supports spatial query. The spatial reasoning is based on a data to structure called 2D string [3] which preserves the objects' spatial knowledge embedded in images. Each symbolic picture can be represented by a 2D string and a picture query can also be specified by a 2D string. The problem of image retrieval then becomes a problem of 2D string subsequence matching [15]. Lee and Hsu [13] proposed 2D C-string representation for nonzero sized objects with a set of spatial operators and a more efficient cutting mechanism. All the spatial relations among objects with efficient segmentation are preserved with 2D C-string representation. The problems of how to infer the spatial relations between pictorial objects from a given 2D C-string in spatial reasoning and similarity retrieval are solved by using the ranking mechanism [14].

In general, similarity retrieval is needed when users cannot express queries in a precise way. The target of similarity retrieval for images is to retrieve the images that are most similar to the query image. The similarity between two patterns or pictures can be measured on the basis of the maximum-likelihood or minimum-distance criterion [12]. The similarity based upon the minimum-distance criterion has been proposed using the techniques of 2D string matching defined in terms of longest common subsequence [3]. However, the 2D string representation is deficient in describing the spatial knowledge of the nonzero sized objects with overlapping. The similarity retrieval based on 2D

* Corresponding author.
E-mail: fjhsu@info4.csie.nctu.edu.tw.

1047-3203/98 $25.00

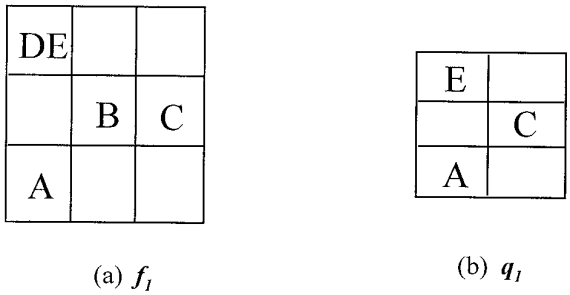
C-strings investigated by Lee and Hsu instead adopts the maximum-likelihood approach in terms of maximum degree of object-pairs clique [14]. All the spatial relationships between object pairs, which is $O(N^2)$ for N objects in an image, need to be reasoned first. The algorithm for similarity retrieval based on 2D C-strings actually finds a maximum clique and becomes an NP-complete problem.

In this paper, we use a transformed structure of the 2D C-string, called 2D C-tree [10], to be the spatial representation for images and propose the tree-matching algorithm for similarity retrieval. The 2D C-tree is an ordered labeled tree, which preserves the complete spatial knowledge among objects without spatial operators. The structural tree representation plays a significant role in retrieving images by tree-matching. We briefly review the 2D string indexing approach in the next section. In Section 3 the basic structure of a 2D C-tree and a sample image representation are introduced. The metric for tree distance computation is defined in Section 4. Then we propose a specific tree-matching algorithm to solve the problem of image retrieval in Section 5. The image retrieval algorithm is modified to compute the partial tree distance for subpicture query. This work is explored in Section 6. Simulation results for verifying the effectiveness of similarity retrieval by 2D C-trees matching are presented in Section 7. Also, an experimental project applying the proposed algorithms to video information system is described in Section 8. Finally, conclusions are summarized in the last section.

2. 2D STRING INDEXING APPROACHES

Given a physical image at the pixel level, the objects and their relative positions within the image can be extracted by using various image processing and understanding techniques [17]. Although this task is computationally expensive, it is performed only at the time of inserting images into the database. Moreover, this task may be carried out in an automated fashion or in a human-assisted semi-automated fashion, depending on the domain and the complexity of the images. A symbolic image is then obtained by associating a name with each of the domain objects thus identified. An image composed of a set of graphic icons that represents the symbolic objects is named an iconic image. The idea of representing physical images by iconic images is similar to the representation of documents by index terms in bibliographic information systems. We use the terms *image, symbolic image,* and *iconic image* interchangeably in this article.

The 2D string approach for spatial indexing was initially proposed by Chang *et al.* [3] to represent iconic images. Three spatial relation operators "<," "=," and ":" are employed in 2D strings. The operator "<" denotes the "left-right" or "below-above" spatial relation. The operator "=" denotes the "at the same spatial location as" relation. The operator ":" denotes the "in the same set as" relation. The symbolic picture f_1 in Fig. 1a may be represented as the 2D string **(A = D:E < B < C, A < B = C < D:E)** or as **(A = DE < B < C, A < B = C < DE)**, where the symbol ":" can be omitted and is omitted.



(a) f_1 (b) q_1

FIG. 1. A symbolic image and a query sketch.

The 2D string representation is also suitable for formulating picture queries. In fact, we can imagine that the query can be specified graphically, by drawing an iconic image on the screen of a computer. The graphic representation, called an *icon sketch*, can be translated into 2D string representation. For example, we may want to retrieve images satisfying a certain icon sketch q_1 as in Fig. 1b. Then q_1 can be translated into the 2D string **(A = E < C, A < C < E)**. This query string is a substring of the 2D string representation of the example image f_1. The problem of image retrieval then becomes the problem of 2D string subsequence matching.

However, the spatial operators of 2D strings are not sufficient to give a complete description of spatial knowledge for images of arbitrary complexity. The 2D G-string representation were proposed to handle more types of relations between pictorial objects [2], but they are not economic for complex images in terms of storage space efficiency and navigation complexity. Lee and Hsu [13] proposed 2D C-string representation with a set of spatial operators and a more efficient cutting mechanism. They employed a characteristic set of spatial operators illustrated in Table 1 to give a complete description for images of arbitrary complexity.

Basically, the 2D C-string approach performs a cut to handle the cases of objects with partly overlapping. The *global* operators "<" and "|," which are employed in the original 2D string approach, handle the cases of nonoverlapping. The extended operators "=," "[," "%," and "]," called the *local* operators, and a pair of separators "( )" handle the cases of overlapping. The picture f_2 in Fig. 2a is similar to f_1 in Fig. 1a, except that the objects in f_2 are nonzero sized objects as opposed to point objects in f_1. The 2D C-string representation of the picture f_2 is **(A]D[E|B|C, A|B%C|D%E)**. It is noted that all the objects in f_2 keep

TABLE 1
The Definition of Characteristic Spatial Operators

Notation	Condition	Meaning
$A < B$	$A_e < B_b$	A disjoints B.
$A \mid B$	$A_e = B_b$	A is edge to edge with B.
$A = B$	$A_b = B_b$, and $A_e = B_e$	A is the same as B.
$A [B$	$A_b = B_b$, and $A_e > B_e$	A contains B and they have the same begin-bound.
$A] B$	$A_b < B_b$, and $A_e = B_e$	A contains B and they have the same end-bound.
$A \% B$	$A_b < B_b$, and $A_e > B_e$	A contains B and they do not have the same bound.
A / B	$A_b < B_b < A_e < B_e$	A is partly overlapping with B.

Note. The notations A_b and A_e (B_b and B_e) denote the values of begin- and end-bounds of objects A and B, respectively.

intact without cutting because the case of partly overlapping does not happen.

The 2D C-string is efficient in the representation and manipulation of images, but it is not suitable in image retrieval based on 2D string subsequence matching. For example, we use a query sketch q_2 as in Fig. 2b, which is a subpicture of f_2. The 2D C-string of the query image, **(A%E < C, A < C < E)** of q_2, is quite different in the format from the 2D C-string of f_2, due to the spatial operators. The string q_2 is not a substring of the string f_2 any longer. The operators are needed to handle the global and local relations among symbolic objects in a 2D C-string and cannot be omitted.

Although the inference of the spatial relations between objects from a given 2D C-string in spatial reasoning can be solved by using the ranking mechanism [14], the computation of object ranks in a 2D C-string is somewhat complicated. Moreover, all the spatial relationships of objects pairs, which is $O(N^2)$ for N objects in an image, are required to be reasoned first by adopting the 2D longest common subsequence algorithm [15]. The algorithm for similarity retrieval actually finds a maximum clique of the corresponding association graph and becomes an NP-complete problem although there are some polynomial time algorithms for the average case. Therefore we explore a more efficient representation and a matching algorithm to solve the problem of image similar retrieval.

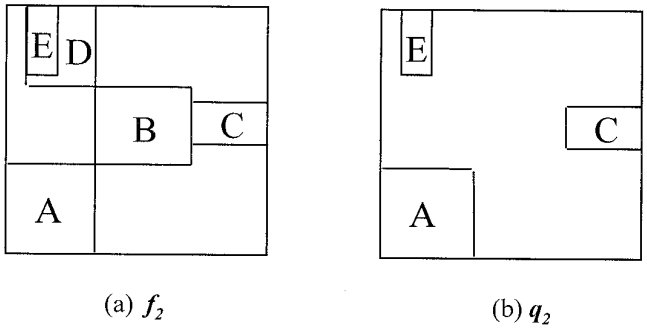


FIG. 2. A symbolic image with nonzero sized objects and a query sketch.

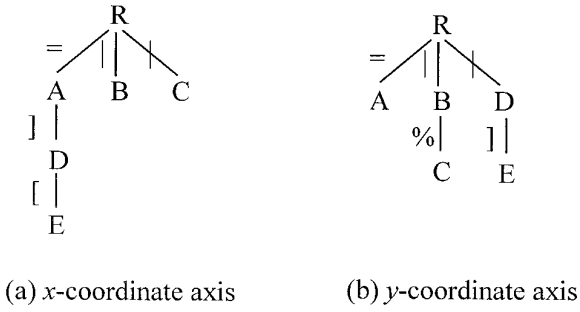


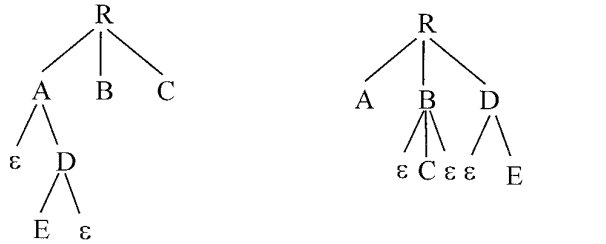
FIG. 3. The signed 2D C-trees of image f_2.

3. 2D C-TREE

The 2D C-tree is an ordered labeled tree. We first introduce the basic structure of a 2D C-tree. The 2D C-tree representation still employs the sparse cutting mechanism to handle the case of symbolic images with partly overlapping objects [10]. The cutting mechanism performs only essential cuttings to get rid of the ambiguity incurred due to partly overlapping. After cutting, an image is partitioned to some portions between two cuttings. All the portions are sequentially linked to a root, **R**, which is initialized to represent the margin or boundary of the area covered by a given image.

The original 2D C-tree representation, called the signed 2D C-tree, is proposed with associated spatial operators. Each node with label, or symbol name, represents an object in the image. The link connecting two nodes, called the signed link, is signed with the relation operator. For the ordered subtree rooted at node S with n immediate descendants in the ordering $s_1, s_2, \ldots, s_n$, S, being the parent, actually contains the *local body* consisting of all its immediate child-nodes $s_1, s_2, \ldots, s_n$. The relation operator between node S and its first child-node s_1 is surely a *local* operator that indicates the ensemble relationship between S and the *local body* consisting of all its child-nodes $s_1, s_2, \ldots, s_n$. The relation operators between node S and other child-nodes s_i ($2 \le i \le n$) are definitely *global* operators that indicate the sibling relation between the child-node s_i ($2 \le i \le n$) and the prior child-node s_{i-1} of node S. The signed 2D C-trees of f_2 are constructed as shown in Fig. 3.

However, a tree with signed links is somewhat unusual for general applications. The *empty-node* and *set-node* are employed in order to remove the relation operator from the signed link according to the basic definition of the



(a) f_{2X} on x-coordinate axis (b) f_{2Y} on y-coordinate axis

FIG. 4. The General 2D C-trees of image f_2.

operators. An *empty-node* is a pseudo node which is labeled "ε" and can be of various sizes. The relation operator of the signed link can be removed by inserting some suitable *empty-nodes*. When the relation operators are stripped off from a signed 2D C-tree, each node of the transformed tree has at least two child-nodes, except the node that originally connects to its single child-node by the "=" operator. The "=" operator possesses the commutation law, which is different from other relation operators of the 2D C-tree. The objects which are connected with the "=" operator have the same begin-bound and end-bound. For the reasoning of spatial relationship among the nodes of the 2D C-tree, a special *set-node* is introduced for treating a set of lineage that each node has single child-node. A *set-node* is a multilabel node consisting of objects that have the same begin-bound and end-bound. The detailed transformation rules are investigated in [10]. The sample symbolic image f_2 in Fig. 2a is represented in a General 2D C-tree as shown in Fig. 4.

4. TREE METRIC

Ordered labeled trees are trees whose nodes are labeled and in which the left to right ordering among siblings is significant. The distance and/or similarity of such trees have many applications in computer vision, pattern recognition, programming compilation, and natural language processing [7]. The distance between two ordered trees is considered to be the weighted number of editing operations required to transform one tree to another. Many algorithms have been developed for ordered labeled tree matching and comparison [23]. Currently the best algorithm for computing the editing distance was presented by Zhang and Shasha [22]. In this section we introduce the distance metric between trees to be the basis for presenting an elegant tree-matching algorithm in image retrieval.

Three kinds of editing operations of a labeled tree [23] are considered and illustrated in Fig. 5. Relabeling a node s means changing the label on s. Deleting a node s means making the children of s become the children of the parent of s and then removing s. Inserting is the complement of deleting. Inserting s as a child of r will make s become the parent of a consecutive subsequence of the current children of r.

These editing operations can be represented as $\alpha \to \beta$, where α is either Λ (null) or a label in tree $\mathcal{T}_1$ and β is either Λ or a label in tree $\mathcal{T}_2$. We call $\alpha \to \beta$ a relabel operation if $\alpha \neq \Lambda$ and $\beta \neq \Lambda$, a delete operation if $\beta = \Lambda$, and an insert operation if $\alpha = \Lambda$. Let Δ be a cost function which assigns a nonnegative real number, referred as $\Delta(\alpha \to \beta)$, to each editing operation $\alpha \to \beta$. The cost can vary in different operations on different nodes. For example, a higher node in a tree has a greater weight than a lower one. Nevertheless, the cost of each editing operation on any node is set equal for simplicity in this paper.

The cost function Δ of each editing operation is constrained to be a distance metric [5]. That is,

(i) $\Delta(\alpha \to \beta) \geq 0$, $\Delta(\alpha \to \alpha) = 0$, *positivity*;
(ii) $\Delta(\alpha \to \beta) = \Delta(\beta \to \alpha)$, *symmetry*;
(iii) $\Delta(\alpha \to \gamma) \leq \Delta(\alpha \to \beta) + \Delta(\beta \to \gamma)$, *triangle inequality.*

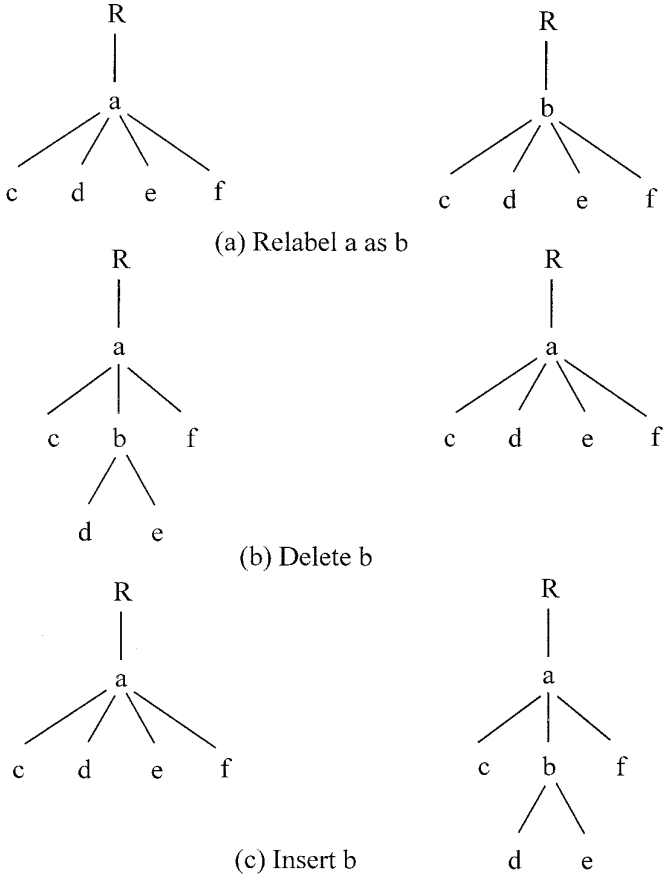


FIG. 5. Three editing operations on labeled tree.

Let E be a sequence $e_1, e_2, \ldots, e_k$ of editing operations. An E-derivation from tree A to tree B is a sequence of trees $A_0, A_1, \ldots, A_k$ such that $A = A_0$, $B = A_k$, and $A_{i-1} \rightarrow A_i$, via editing operation e_i for $1 \leq i \leq k$. Then the cost function Δ can be extended to the sequence of editing operations by letting

$$\Delta(E) = \sum_{i=1}^{|E|} \Delta(e_i). \quad (1)$$

The editing distance between two trees is defined as minimum cost of the editing sequence that transforms one tree to the other. Formally the editing distance between trees $\mathscr{T}_1$ and $\mathscr{T}_2$ is defined as

$$\delta(\mathscr{T}_1, \mathscr{T}_2) = min\{\Delta(E) \mid E \text{ is an editing sequence from } \mathscr{T}_1 \text{ to } \mathscr{T}_2\}. \quad (2)$$

The editing sequence can be treated as a mapping that is a graphical specification of editing operations applied to the nodes in the two ordered trees. Suppose that we have a numbering mechanism, for example, the postorder numbering for a tree. Let $\mathscr{T}[i]$ be the ith node of tree $\mathscr{T}$ in the postorder numbering. Formally, we define a triple (M, $\mathscr{T}_1, \mathscr{T}_2$) to be a mapping from $\mathscr{T}_1$ to $\mathscr{T}_2$, where mapping M is the set of integer pairs (i, j) satisfying:

(1) $1 \leq i \leq |\mathscr{T}_1|$, $1 \leq j \leq |\mathscr{T}_2|$;

(2) For two pairs of (i_1, j_1) and (i_2, j_2) in M,

(a) $i_1 = i_2$ if and only if $j_1 = j_2$ (one-to-one),

(b) $\mathscr{T}_1[i_1]$ is to the left of $\mathscr{T}_1[i_2]$ if and only if $\mathscr{T}_2[j_1]$ is to the left of $\mathscr{T}_2[j_2]$ (sibling order preserved),

(c) $\mathscr{T}_1[i_1]$ is an ancestor of $\mathscr{T}_1[i_2]$ if and only if $\mathscr{T}_2[j_1]$ is an ancestor of $\mathscr{T}_2[j_2]$ (ancestor order preserved).

(3) Let $lca(i_1, i_2)$ represent the least common ancestor node of i_1 and i_2. For three pairs of (i_1, j_1), (i_2, j_2), and (i_3, j_3) in M, $\mathscr{T}_1[lca(i_1, i_2)]$ is a proper ancestor of $\mathscr{T}_1[i_3]$ if and only if $\mathscr{T}_2[lca(j_1, j_2)]$ is a proper ancestor of $\mathscr{T}_2[j_3]$.

Let M be a mapping from $\mathscr{T}_1$ to $\mathscr{T}_2$. Let I and J be the sets of unmatched nodes in $\mathscr{T}_1$ and $\mathscr{T}_2$, respectively. We will use M instead of (M, $\mathscr{T}_1$, $\mathscr{T}_2$) if there is no confusion. Then we can define the cost of M:

$$\Delta(M) = \sum_{(i,j)\in M} \Delta(\mathscr{T}_1[i] \rightarrow \mathscr{T}_2[j]) + \sum_{i\in I} \Delta(\mathscr{T}_1[i] \rightarrow \Lambda) + \sum_{j\in J} \Delta(\Lambda \rightarrow \mathscr{T}_2[j]). \quad (3)$$

Hence,

$$\delta(\mathscr{T}_1, \mathscr{T}_2) = min\{\Delta(M) \mid M \text{ is a mapping from } \mathscr{T}_1 \text{ to } \mathscr{T}_2\}. \quad (4)$$

5. STRUCTURAL IMAGE RETRIEVAL

Now, we begin to introduce the tree matching algorithm for image retrieval. The 2D C-tree is an ordered labeled tree. Since the nodes of a 2D C-tree may be *empty-nodes* or *set-nodes*, we must make a small but significant modification of the tree-matching algorithm developed by Zhang and Shasha [22].

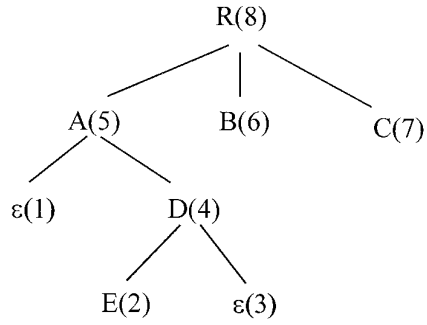


FIG. 6. The 2D C-tree f_{2X} of Fig. 4a with postorder numbering.

Suppose that A is a node in tree $\mathscr{T}_1$. N(A) denotes the number of labels of node A. If A is an *empty-node*, A has a special label "ε" and N(A) is one. If A is a *set-node*, N(A) must be more than one. For an editing operation $A \rightarrow B$, where B is a node in tree $\mathscr{T}_2$, the cost function needs to be re-examined:

(1) The cost of a delete operation $A \rightarrow \Lambda$, $\Delta(A \rightarrow \Lambda)$, is defined as N(A). That is, $\Delta(A \rightarrow \Lambda) = N(A)$.

(2) The cost of an insert operation $\Lambda \rightarrow B$, $\Delta(\Lambda \rightarrow B)$, is defined as N(B). That is, $\Delta(\Lambda \rightarrow B) = N(B)$.

(3) The cost of a relabel operation $A \rightarrow B$, $\Delta(A \rightarrow B)$, is defined as the larger of two numbers N(A/B) and N(B/A). Let N(A/B) represent the number of labels of node A which are differentiated from those of node B. That is, $\Delta(A \rightarrow B) = max\{N(A/B), N(B/A)\}$. If one of them is an *empty-node*, the cost is the number of labels of the other node.

These cost functions defined above are still under the constraints of distance metric. In the following some notations on trees are illustrated in Fig. 6 using the tree f_{2X} of Fig. 4a with postorder numbering in the parenthesis as an example:

(1) $\mathscr{T}[i]$. The ith node in the tree $\mathscr{T}$ according to the left-to-right postorder numbering. (Ex. The label of $\mathscr{T}[2]$ is E.)

(2) $\theta(i)$. The number of the leftmost leaf descendant of the subtree rooted at $\mathscr{T}[i]$. When $\mathscr{T}[i]$ is a leaf node, $\theta(i) = i$. (Ex. $\theta(4) = 2$; i.e., E is the leftmost leaf descendant of D.)

(3) $\partial(i)$. The depth of $\mathscr{T}[i]$; it is the number of nodes on the path from the root of tree $\mathscr{T}$ to node $\mathscr{T}[i]$, excluding $\mathscr{T}[i]$. (Ex. $\partial(2) = 3$; i.e., the depth of E is 3.)

(4) P(i). The set of all the predecessors of $\mathscr{T}[i]$. Also,

$P^k(i)$ denotes the kth level predecessor of $\mathscr{T}[i]$, where the level is counted from node $\mathscr{T}[i]$ backward to the root. $P(i) = \{P^k(i) \mid 1 \le k \le \partial(i)\}$. (Ex. $P(2) = \{4, 5, 8\}$; i.e., the predecessors of E are D, A, and R.)

(5) $\mathscr{T}[i \,..\, j]$. An ordered subforest of tree $\mathscr{T}$ induced by the nodes numbered from i to j inclusive. If $i > j$, then $\mathscr{T}[i \,..\, j] = \varnothing$. (Ex. $\mathscr{T}[2 \,..\, 6]$ includes E, ε, D, A, and B.)

(6) *Forest*(i). An ordered subforest $\mathscr{T}[1 \,..\, i]$. (Ex. *Forest*(4) includes ε, E, ε, and D.)

(7) *Tree*(i). A subtree of $\mathscr{T}$ rooted at $\mathscr{T}[i]$. $\mathscr{T}[\theta(i) \,..\, i]$ will be referred to as *Tree*(i). (Ex. *Tree*(4) is equivalent to $\mathscr{T}[2 \,..\, 4]$.)

(8) *Size*(i). The number of nodes in *Tree*(i). (Ex. *Size*(4) = 3.)

(9) *ForestDist*($i' \,..\, i, j' \,..\, j$). The distance between two subforests $\mathscr{T}_1[i' \,..\, i]$ in $\mathscr{T}_1$ and $\mathscr{T}_2[j' \,..\, j]$ in $\mathscr{T}_2$. We use an abbreviated notation *ForestDist*(i, j) for the distance between $\mathscr{T}_1[1 \,..\, i]$ and $\mathscr{T}_2[1 \,..\, j]$ (see below).

(10) TreeDist(i, j). The distance between the subtree *Tree*(i) rooted at i in $\mathscr{T}_1$ and the subtree *Tree*(j) rooted at j in $\mathscr{T}_2$ (see below).

The following three lemmas are necessary for the tree distance computation algorithm.

LEMMA 1. *Let* $i^p \in P(i)$ *and* $j^p \in P(j)$. *Then*

(i) $$\textit{ForestDist}(\varnothing, \varnothing) = 0. \tag{5}$$

(ii) $$\textit{ForestDist}(\theta(i^p) \,..\, i, \varnothing) = \textit{ForestDist}(\theta(i^p) \,..\, i-1, \varnothing) + \Delta(\mathscr{T}_1[i] \to \Lambda). \tag{6}$$

(iii) $$\textit{ForestDist}(\varnothing, \theta(j^p) \,..\, j) = \textit{ForestDist}(\varnothing, \theta(j^p) \,..\, j-1) + \Delta(\Lambda \to \mathscr{T}_2[j]). \tag{7}$$

Case (i) requires no editing operation and is assigned 0 for initialization. In (ii), the distance corresponds to the cost of deleting a node $\mathscr{T}_1[i]$ from a forest $\mathscr{T}_1[\theta(i^p) \,..\, i]$. The forest $\mathscr{T}_1[\theta(i^p) \,..\, i]$ is led by the leftmost leaf node of a tree *Tree*(i^p) containing node $\mathscr{T}_1[i]$ in $\mathscr{T}_1$ and concluded at $\mathscr{T}_1[i]$. In (iii), the distance corresponds to the cost of inserting a node $\mathscr{T}_2[j]$ into a forest $\mathscr{T}_2[\theta(j^p) \,..\, j-1]$ in $\mathscr{T}_2$.

LEMMA 2. $$\textit{ForestDist}(\theta(i) \,..\, i, \theta(j) \,..\, j) = \textit{TreeDist}(i, j) = \textit{min}\{$$

$$\begin{aligned}
&\textit{ForestDist}(\theta(i) \,..\, i-1, \theta(j) \,..\, j) + \Delta(\mathscr{T}_1[i] \to \Lambda),\\
&\textit{ForestDist}(\theta(i) \,..\, i, \theta(j) \,..\, j-1) + \Delta(\Lambda \to \mathscr{T}_2[j]),\\
&\textit{ForestDist}(\theta(i) \,..\, i-1, \theta(j) \,..\, j-1) + \Delta(\mathscr{T}_1[i] \to \mathscr{T}_2[j])\}.
\end{aligned} \tag{8}$$

Lemma 2 computes the distance between two subtrees rooted at $\mathscr{T}_1[i]$ in $\mathscr{T}_1$ and $\mathscr{T}_2[j]$ in $\mathscr{T}_2$, respectively. Consider the mapping of two roots, $\mathscr{T}_1[i]$ and $\mathscr{T}_2[j]$. The distance between *Tree*(i) of $\mathscr{T}_1$ and *Tree*(j) of $\mathscr{T}_2$ is the minimum of the three possible editing mapping costs: delete $\mathscr{T}_1[i]$ from $\mathscr{T}_1$, insert $\mathscr{T}_2[j]$ from $\mathscr{T}_2$, or relabel $\mathscr{T}_1[i]$ as $\mathscr{T}_2[j]$.

LEMMA 3. *Let* $i^p \in P(i)$ *and* $j^p \in P(j)$. *If* $\theta(i^p) \ne \theta(i)$, *or* $\theta(j^p) \ne \theta(j)$, *then*

$$\begin{aligned}
&\textit{ForestDist}(\theta(i^p) \,..\, i, \theta(j^p) \,..\, j) = \textit{min}\{\\
&\quad \textit{ForestDist}(\theta(i^p) \,..\, i-1, \theta(j^p) \,..\, j) + \Delta(\mathscr{T}_1[i] \to \Lambda),\\
&\quad \textit{ForestDist}(\theta(i^p) \,..\, i, \theta(j^p) \,..\, j-1) + \Delta(\Lambda \to \mathscr{T}_2[j]),\\
&\quad \textit{ForestDist}(\theta(i^p) \,..\, \theta(i)-1, \theta(j^p) \,..\, \theta(j)-1)\\
&\qquad + \textit{TreeDist}(i, j)\}.
\end{aligned} \tag{9}$$

Lemma 3 considers the distance between two forests. The forest $\mathscr{T}_1[\theta(i^p) \,..\, i]$ ($\mathscr{T}_2[\theta(j^p) \,..\, j]$) is led by the leftmost leaf node of a tree *Tree*(i^p) (*Tree*(j^p)) containing the considered node $\mathscr{T}_1[i]$ ($\mathscr{T}_2[j]$) and concluded at $\mathscr{T}_1[i]$ ($\mathscr{T}_2[j]$). The distance between $\mathscr{T}_1[(\theta(i^p) \,..\, i]$ of $\mathscr{T}_1$ and $\mathscr{T}_2[(\theta(j^p) \,..\, j]$ of $\mathscr{T}_2$ is the minimum of the three possible editing mapping costs: delete $\mathscr{T}_1[i]$ from $\mathscr{T}_1$, insert $\mathscr{T}_2[j]$ into $\mathscr{T}_2$, or substitute the subtree *Tree*(i) of $\mathscr{T}_1$ by *Tree*(j) of $\mathscr{T}_2$.

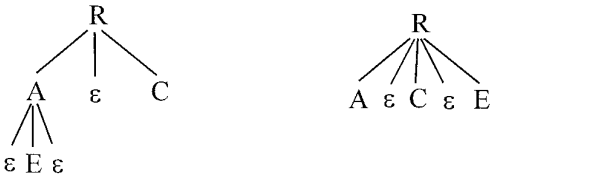
For proofs of Lemmas 1, 2, and 3, refer to [22].

The algorithm to compute the tree distance uses a dynamic programming style [16]. From Lemma 3 we observe that to compute *TreeDist*(i^p, j^p) we need in advance almost all values of *TreeDist*(i, j) where $\mathscr{T}_1[i^p]$ is the root of a subtree containing $\mathscr{T}_1[i]$ and $\mathscr{T}_2[j^p]$ is the root of a subtree containing $\mathscr{T}_2[j]$. This suggests a bottom-up procedure for computing all subtree pairs.

ALGORITHM 1. The computation of *TreeDist*(x, y).

Input: Two subtrees, *Tree*(x) rooted at x in tree $\mathscr{T}_1$ and *Tree*(y) rooted at y in tree $\mathscr{T}_2$.
Output: The distance *TreeDist*(x, y).

```
Begin
  ForestDist(∅, ∅) = 0;
  for i := θ(x) to x
    ForestDist(θ(x) .. i, ∅) = ForestDist(θ(x) .. i − 1, ∅) + N(i);
  for j := θ(y) to y
    ForestDist(∅, θ(y) .. j) = ForestDist(∅, θ(y) .. j − 1) + N(j);
  for i := θ(x) to x
    for j := θ(y) to y
      if θ(i) = θ(x) and θ(j) = θ(y), then
          ForestDist(θ(x) .. i, θ(y) .. j) = min{
            ForestDist(θ(x) .. i − 1, θ(y) .. j) + N(i),
            ForestDist(θ(x) .. i, θ(y) .. j − 1) + N(j),
            ForestDist(θ(x) .. i − 1, θ(y) .. j − 1) +
            max{N(A/B), N(B/A)}};
          TreeDist(i, j) = ForestDist(θ(x) .. i, θ(y) .. j);
        else
          ForestDist(θ(x) .. i, θ(y) .. j) = min{
            ForestDist(θ(x) .. i − 1, θ(y) .. j) + N(i),
            ForestDist(θ(x) .. i, θ(y) .. j − 1) + N(j),
            ForestDist(θ(x) .. θ(i) − 1, θ(y) .. θ(j) −
            1) + TreeDist(i, j)};
End;
```



(a) q_{2X} on x-coordinate axis (b) q_{2Y} on y-coordinate axis

FIG. 7. The 2D C-trees of query sketch q_2.

For two 2D C-trees, $\mathcal{T}_1$ and $\mathcal{T}_2$, rooted at R_1 and R_2, respectively, the distance between them, denoted by $\delta(\mathcal{T}_1, \mathcal{T}_2)$, is computed as the value of $TreeDist(R_1, R_2)$. We can use the algorithm to compute the distance of 2D C-trees for solving the picture query problem. Consider two pictures, P_1 and P_2. Two 2D C-tree representations of P_1 (P_2), $\mathcal{T}_{1x}$ ($\mathcal{T}_{2x}$), and $\mathcal{T}_{1Y}$ ($\mathcal{T}_{2Y}$) along x-coordinate and y-coordinate, respectively, are constructed. We define the distance between P_1 and P_2 as follows.

DEFINITION 1. The distance between two pictures P_1 and P_2, $\delta(P_1, P_2)$, is $\delta(\mathcal{T}_{1x}, \mathcal{T}_{2x}) * \delta(\mathcal{T}_{1Y}, \mathcal{T}_{2Y})$. If $\delta(\mathcal{T}_{1x}, \mathcal{T}_{2x})$ is zero, then define $\delta(P_1, P_2) = \delta(\mathcal{T}_{1Y}, \mathcal{T}_{2Y})$. On the contrary, if $\delta(\mathcal{T}_{1Y}, \mathcal{T}_{2Y})$ is zero, then define $\delta(P_1, P_2) = \delta(\mathcal{T}_{1X}, \mathcal{T}_{2X})$.

We use the example picture f_2 in Fig. 2a and query sketch q_2 in Fig. 2b to demonstrate the computation of picture distance. The 2D C-trees of f_2 and q_2 along x-coordinate axis are in Figs. 4a and 7a correspondingly. The editing distance between these two trees is the cost of editing operations required to transform f_{2x} to q_{2x}. At least two editing operations are needed. That is, $\Delta(D \to \Lambda)$ and $\Delta(B \to \varepsilon)$. So the tree distance of $\delta(f_{2x}, q_{2x})$ is 2. Three delete operations, $\Delta(B \to \Lambda)$, $\Delta(D \to \Lambda)$, and $\Delta(\varepsilon \to \Lambda)$, are needed between f_{2Y} in Fig. 4b and q_{2Y} in Fig. 7b along y-coordinate. That is, the cost of $\delta(f_{2Y}, q_{2Y})$ is 3. Finally, the distance of $\delta(f_2, q_2)$ is 6.

Moreover, in [22] Zhang and Shasha had defined an LR_keyroots set of tree $\mathcal{T}$, $LR\_keyroots(\mathcal{T})$, to efficiently reduce the computation time of tree distance. The complexity of the algorithm is $O(|\mathcal{T}_1| * |\mathcal{T}_2| * min(\text{depth}(\mathcal{T}_1), \text{leaves}(\mathcal{T}_1)) * min(\text{depth}(\mathcal{T}_2), \text{leaves}(\mathcal{T}_2)))$. Let $\text{depth}(\mathcal{T}_1)$ denote the depth of the tree $\mathcal{T}_1$ and $\text{leaves}(\mathcal{T}_1)$ denote the number of leaf nodes of the tree $\mathcal{T}_1$. In general, the fast algorithm takes $O(n^4)$ for computing the editing distance between two trees consisting of n nodes. The parallel algorithm is the time of complexity $O(|\mathcal{T}_1| * |\mathcal{T}_2|)$ by using $O(min(|\mathcal{T}_1|, |\mathcal{T}_2|) * \text{leaves}(\mathcal{T}_1) * \text{leaves}(\mathcal{T}_2))$ processors.

While all the tree distances between the query image and the images in the database have been computed, the most similar image can be obtained. Suppose that there are n images in the database, $P_1, P_2, \ldots, P_n$, and a query image Q. The most similar image(s) to Q is

$$\{P_i \mid \delta(P_i, Q) \text{ is the minimum of } \delta(P_k, Q), 1 \le k \le n\}.$$

6. SUBPICTURE QUERY

Subpicture query is useful when a user cannot express queries in a precise way [5]. Sometimes we may ask "please retrieve images that contain this specific subpicture," or "I want some images that have some part like this query sketch." For example, the query image q_2 in Fig. 2b is a subpicture of f_2 in Fig. 2a. An approximate-tree-by-example (ATBE) system [19] developed by Wang *et al.* manipulates the inexact query by approximate tree matching. But the cutting and pruning operations that remove all the descendants of a node are somehow not suitable for subpicture query. The tree distance computation algorithm proposed in the previous section not only can support a simple measure for similarity retrieval, but also it can be modified for a subpicture query. In essence, we adopt the tree-matching algorithm and modify the cost functions of editing operations as required.

(1) The cost of delete operation is weighted zero. Deleting a symbol from a *reference* image means that this symbol does not appear in the *query* image. For subpicture query, the symbols existing in the *reference* image may not be expressed in the *query* image or specified subpicture. In such a case, the superfluous symbols in *reference* image can be ignored on purpose when they do not appear in the query image and are allowed to delete with zero cost. The cost of editing operation $A \to \Lambda$ is weighted zero; i.e., $\Delta(A \to \Lambda) = 0$.

(2) The cost of an insert operation $\Lambda \to B$, i.e., $\Delta(\Lambda \to B)$, is not changed because all the symbols of the query image should be considered. That is, $\Delta(\Lambda \to B) = N(B)$.

(3) The cost of a relabel operation $A \to B$, i.e., $\Delta(A \to B)$, is slightly changed and is defined as the number of unmatched symbols in B, which do not appear in A. That is, $\Delta(A \to B) = N(B/A)$. One special case is for $B = \varepsilon$. The cost of $\Delta(A \to \varepsilon)$ is redefined to be 0 because the symbol(s) in A can be viewed as an empty-node in B.

Obviously, the newly defined cost functions, called partial cost functions, of editing operations do not obey the symmetry constraint of distance metric. Although the delete operation is not the inverse function of insert operation any more, the constraint of a distance metric is not our major concern for subpicture query. The partial cost functions directly affect computation of the tree distance, based upon the lemmas in the previous section. In Lemma 1, the second sentence $ForestDist(\theta(i^p) \ldots i, \varnothing)$ is always zero because $\Delta(\mathcal{T}_1[i] \to \Lambda) = 0$. In Lemma 2 and Lemma 3, the

second value of the first statement in the minimum group, i.e., $\Delta(\mathscr{T}_1[i] \rightarrow \Lambda)$, is removed because the delete operation is weighted zero also. Consequently, the algorithm of partial tree distance is modified as follows.

ALGORITHM 2. The computation of *PartialTreeDist* (x, y).

Input: Two subtrees, *Tree*(x) rooted at x in tree $\mathscr{T}_1$ and *Tree*(y) rooted at y in tree $\mathscr{T}_2$.
Output: The distance *PartialTreeDist*(x, y).

```
Begin
  ForestDist(∅, ∅) = 0;
  for i := θ(x) to x
    ForestDist(θ(x) .. i, ∅) = 0;
  for j := θ(y) to y
    ForestDist(∅, θ(y) .. j) = ForestDist(∅, θ(y) ..
    j − 1) + N(j);
  for i := θ(x) to x
    for j := θ(y) to y
      if θ(i) = θ(x) and θ(j) = θ(y), then
          ForestDist(θ(x) .. i, θ(y) .. j) = min{
            ForestDist(θ(x) .. i − 1, θ(y) .. j),
            ForestDist(θ(x) .. i, θ(y) .. j − 1) + N(j),
            ForestDist(θ(x) .. i − 1, θ(y) .. j − 1) +
            N(j/i)};
          PartialTreeDist(i, j) = ForestDist(θ(x) .. i,
          θ(y) .. j);
        else
          ForestDist(θ(x) .. i, θ(y) .. j) = min{
            ForestDist(θ(x) .. i − 1, θ(y) .. j),
            ForestDist(θ(x) .. i, θ(y) .. j − 1) + N(j),
            ForestDist(θ(x) .. θ(i) − 1, θ(y) .. θ(j) −
            1) + PartialTreeDist(i, j)};
End;
```

Then, we can use the partial tree-matching algorithm to compute the distance of 2D C-trees for solving the subpicture query problem. Let $\gamma(\mathscr{T}_1, \mathscr{T}_2)$ represent the partial tree distance between trees $\mathscr{T}_1$ and $\mathscr{T}_2$. The partial distance between $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$ is defined as follows.

DEFINITION 2. The partial distance between two pictures $\boldsymbol{P_1}$ and $\boldsymbol{P_2}$, $\gamma(\boldsymbol{P_1}, \boldsymbol{P_2})$, is $\gamma(\mathscr{T}_{1x}, \mathscr{T}_{2x}) * \gamma(\mathscr{T}_{1Y}, \mathscr{T}_{2Y})$. If $\gamma(\mathscr{T}_{1x}, \mathscr{T}_{2x})$ is zero, then define $\gamma(\boldsymbol{P_1}, \boldsymbol{P_2}) = \gamma(\mathscr{T}_{1Y}, \mathscr{T}_{2Y})$. On the contrary, if $\gamma(\mathscr{T}_{1Y}, \mathscr{T}_{2Y})$ is zero, then define $\gamma(\boldsymbol{P_1}, \boldsymbol{P_2}) = \gamma(\mathscr{T}_{1X}, \mathscr{T}_{2X})$.

We use the example picture $\boldsymbol{f_2}$ in Fig. 2a and query sketch $\boldsymbol{q_2}$ in Fig. 2b to demonstrate the computation of partial distance. The 2D C-trees of $\boldsymbol{f_2}$ and $\boldsymbol{q_2}$ are in Fig. 4 and Fig. 7, correspondingly. For computing the partial tree distance between $\boldsymbol{f_{2x}}$ and $\boldsymbol{q_{2x}}$, the first editing $\Delta(D \rightarrow \Lambda)$ is a delete operation having zero weight. The second editing $\Delta(B \rightarrow \varepsilon)$ is a special case of relabel operation also weighted zero. So the tree distance of $\gamma(\boldsymbol{f_{2x}}, \boldsymbol{q_{2x}})$ is 0 for x-coordinate direction. The costs of three delete operations, $\Delta(B \rightarrow \Lambda)$, $\Delta(D \rightarrow \Lambda)$, and $\Delta(\varepsilon \rightarrow \Lambda)$, needed for transforming from $\boldsymbol{f_{2Y}}$ to $\boldsymbol{q_{2Y}}$ along y-coordinate are all weighted zero. That is, the cost of $\gamma(\boldsymbol{f_{2Y}}, \boldsymbol{q_{2Y}})$ is 0 also. Finally, the distance of $\gamma(\boldsymbol{f_2}, \boldsymbol{q_2})$ is 0. It means that $\boldsymbol{q_2}$ is a subpicture of $\boldsymbol{f_2}$ with distance zero.

Analogously, the most similar image(s) that contains a query subpicture $\boldsymbol{Q}$ from $\boldsymbol{P_1}, \boldsymbol{P_2}, \ldots,$ and $\boldsymbol{P_n}$ is

$$\{\boldsymbol{P_i} \mid \gamma(\boldsymbol{P_i}, \boldsymbol{Q}) \text{ is the minimum of } \gamma(\boldsymbol{P_k}, \boldsymbol{Q}), 1 \leq k \leq n\}.$$

7. SIMULATION RESULTS

For verifying the effectiveness of similarity retrieval by 2D C-trees matching, a test consisting of 10 simulation pictures is evaluated. A symbolic image with random spatial relationship among objects can be generated by random generation of quadruple-values. Table 2 shows 10 random generated objects A through J with the bounds on x-axis and y-axis, respectively.

We construct 10 simulation pictures, $P_1, P_2, \ldots,$ and P_{10}. Without loss of generality, assume P_1 contains single object, the first object (A). P_2 contains the first two objects (A and B), and so on. The tenth picture P_{10} contains all 10 objects. Fig. 8 depicts the symbolic images P_3 and P_4. It is interesting to find out that P_3 is a subpicture of P_4. It could be foreseen that a picture with less objects is a subpicture of a picture with more objects in this experiment; i.e., P_i is always a subpicture of P_j, for $i \leq j$.

Then these 10 pictures are represented in 2D C-trees, respectively. The 2D C-trees of P_3, referred to as $\mathscr{T}_{3X}$ and $\mathscr{T}_{3Y}$, are shown in Fig. 9 and the 2D C-trees of P_4 in Fig. 10.

For illustrating the computations of tree distances among these 10 pictures, the 2D C-tree representation is expressed by a recursive sentence $\alpha(\alpha_1\alpha_2\alpha_3 \ldots . \alpha_n)$ for a node α, which has n immediate descendants in the ordering $\alpha_1, \alpha_2, \alpha_3, \ldots,$ and α_n. For example, $\mathscr{T}_{3X}$ in Fig. 9 is represented

TABLE 2
The Simulation Data of Ten Objects

Label	A	B	C	D	E	F	G	H	I	J
x-axis	131, 347	165, 358	31, 641	192, 572	5, 358	80, 346	119, 470	213, 584	420, 610	364, 600
y-axis	167, 415	20, 332	217, 363	9, 241	36, 156	431, 466	208, 509	48, 50	355, 467	14, 545

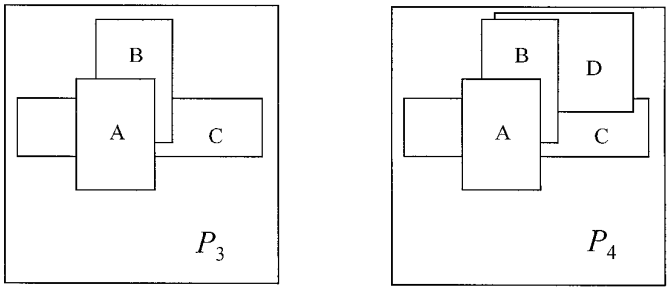


FIG. 8. P_3 is a subpicture of P_4.

as $\boldsymbol{R}$(C(εA(εB)Bε)). Note that $\boldsymbol{R}$, ε and the bracket [ ] denote the root of tree, an empty-node and a set-node, respectively. The 2D C-trees of the 10 simulation pictures are constructed and listed in Fig. 11.

In the sample database, there are 10 generated pictures, $P_1, P_2, \ldots,$ and P_{10}, as *reference* pictures. We also use P_i, $1 \le i \le 10$, as *query* picture to validate the correctness of the matching algorithm. Listed in Table 3 is the exact query and we compute the tree distance δ(*reference*, *query*). Table 4 shows the subpicture query and we compute the partial tree distance γ(*reference*, *query*).

There are some interesting observations in the simulation results:

(1) Table 3 shows that the tree distance computation strictly obeys the constraints of distance metric. That is, for any P_i, P_j, and P_k,

(i) $\delta(P_i, P_j) \ge 0$, and $\delta(P_i, P_i) = 0$ (*positivity*),

(ii) $\delta(P_i, P_j) = \delta(P_j, P_i)$ (*symmetry*),

(iii) $\delta(P_i, P_k) \le \delta(P_i, P_j) + \delta(P_j, P_k)$ (*triangle inequality*).

(2) For any P_j,

(i) if $i < j$, then $\delta(P_i, P_j) < \delta(P_{i-1}, P_j)$;

(ii) if $k > j$, then $\delta(P_k, P_j) < \delta(P_{k+1}, P_j)$.

P_j contains the first j objects. P_i contains the first i objects and is a subset of objects of P_j if $i < j$. P_{i-1} contains all the objects of P_i excluding the ith object. The distance between P_i and P_j is always smaller than the distance between P_{i-1} and P_j. For two pictures P_k and P_{k+1} containing more objects than P_j, the distance between P_k and P_j is always smaller than the distance between P_{k+1} and P_j. The above statements confirm that the computation of tree distances is suitable for measuring the similarity between two pictures. The smaller the distance between two pictures is, the more similar the two pictures are.

(3) It seems apparent that the partial tree distances in Table 4 are not symmetric. The values of the lower-triangle

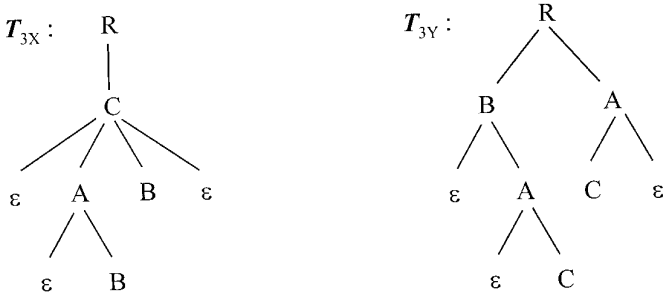


FIG. 9. The 2D C-trees of P_3.

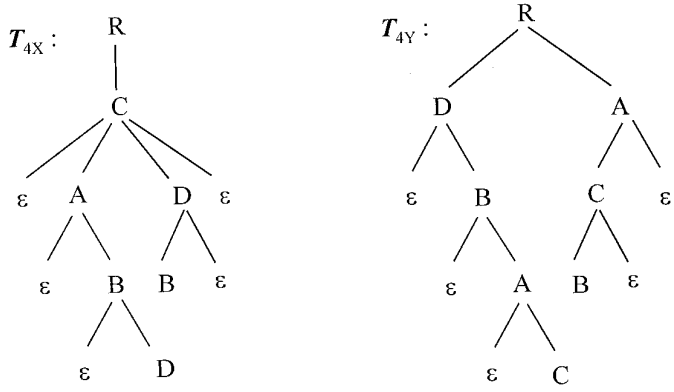


FIG. 10. The 2D C-trees of P_4.

$\mathfrak{I}_{1X}$: R(A)

$\mathfrak{I}_{1Y}$: R(A)

$\mathfrak{I}_{2X}$: R(A(εB)B)

$\mathfrak{I}_{2Y}$: R(B(εA)A)

$\mathfrak{I}_{3X}$: R(C(εA(εB)Bε))

$\mathfrak{I}_{3Y}$: R(B(εA(εC))A(Cε))

$\mathfrak{I}_{4X}$: R(C(εA(εB(εD))D(Bε)ε))

$\mathfrak{I}_{4Y}$: R(D(εB(εA(εC)))A(C(Bε)ε))

$\mathfrak{I}_{5X}$: R(E(εC(εA(εB(εD))[BD]))C(Dε))

$\mathfrak{I}_{5Y}$: R(D(εB(εEεA(εC)))A(C(Bε)ε))

$\mathfrak{I}_{6X}$: R(E(εC(εF(εA(εB(εD)))[BD](Aε)))C(Dε))

$\mathfrak{I}_{6Y}$: R(D(εB(εEεA(εC)))A(C(Bε)ε)εF)

$\mathfrak{I}_{7X}$: R(E(εC(εF(εG(εA(εB(εD))))[GBD](Aε)))C(D(Gε)ε))

$\mathfrak{I}_{7Y}$: R(D(εB(εEεA(εG(εC))))G(A(C(Bε)ε)εFε))

$\mathfrak{I}_{8X}$: R(E(εC(εF(εG(εA(εB(εD(εH)))))[GBDH](Aε)))C(H(D(Gε)ε)ε))

$\mathfrak{I}_{8Y}$: R(D(εB(εE(εHε)εA(εG(εC))))G(A(C(Bε)ε)εFε))

$\mathfrak{I}_{9X}$: R(E(εC(εF(εG(εA(εB(εD(εH)))))[GBDH](Aε)))C(H(D(G(εI)I)I)Iε))

$\mathfrak{I}_{9Y}$: R(D(εB(εE(εHε)εA(εG(εC))))G(A(C(BεI)I)I(εFε)ε))

$\mathfrak{I}_{10X}$:R(E(εC(εF(εG(εA(εB(εD(εH)))))[GBDH](Aε)))C(H(D(G(εI))[JI])[IJ])I(Jε)ε))

$\mathfrak{I}_{10Y}$:R(D(εJ(εB(εE(εHε)εA(εG(εC)))))J(G(A(C(BεI)I)I(εFε)ε)ε))

FIG. 11. The 2D C-tree representations of 10 simulation pictures.

TABLE 3
The Tree Distance Result of Comparing Ten Simulation Pictures

δ Query	Reference									
	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
P_1	0	9	42	110	169	255	418	594	750	1044
P_2	9	0	12	56	100	168	304	456	594	858
P_3	42	12	0	16	54	96	204	330	475	713
P_4	110	56	16	0	16	40	120	220	345	551
P_5	169	100	54	16	0	8	54	126	221	391
P_6	255	168	96	40	8	0	20	70	143	285
P_7	418	304	204	120	54	20	0	15	63	165
P_8	594	456	330	220	126	70	15	0	20	88
P_9	750	594	475	345	221	143	63	20	0	24
P_{10}	1044	858	713	551	391	285	165	88	24	0

are almost the same as those of the tree distances in Table 3. And the values of the upper-triangle are almost zero because the costs of delete operations are weighted zero. Basically, the partial tree distance computation obeys the distance metric except symmetry constraint due to the partial cost functions defined. Note that some very small nonzero values appearing in the upper-triangle happen when the cutting causes some objects being segmented.

(4) The value in the upper-triangle of Table 4 represents the partial tree distance between one (*reference*) picture with more objects and another (*query*) picture with less objects, i.e., $\gamma(P_j, P_i)$ for two pictures P_j and P_i, $j > i$. Since this value is zero or very closer to zero, P_i is viewed as a subpicture of P_j. For example, $\gamma(P_4, P_3) = 0$ implies that P_3 is a subpicture of P_4 with zero cost. The result complies with the fact of the simulation.

The above evidences validate the accountability of our tree-matching algorithms and the effectiveness of similarity retrieval by 2D C-trees matching.

8. PROTOTYPE SYSTEM

We apply the above mechanisms to implement an interactive video information system in our experimental project [11]. We capture 48 streams from ''The Lion King'' cartoon produced by The Walt Disney Company and store the video data in AVI file format. Each stream takes about 99 s and consists of about 1500 frames. Some *key image* frames are identified in a human-assisted fashion for each of the video streams. Notes that this work can be benefited from the motion analysis of recorded scene. These *key images* become representative of the streams. There are 351 *key images* in our experiment and some are listed in Fig. 12. For this popular animation, 78 roles are chosen to be the objects, which are also extracted in human-assisted fashion. These objects are represented by a set of designed icons in the system. The objects and their bounding rectangles within images are also extracted after capturing the image from the source video. Each image containing about five objects in average is constructed into two 2D C-trees

TABLE 4
The Partial Tree Distance Result of Comparing Ten Simulation Pictures

γ Query	Reference									
	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	P_{10}
P_1	0	0	0	0	0	0	0	0	0	0
P_2	9	0	0	0	0	0	0	0	0	0
P_3	42	12	0	0	1	1	1	1	1	1
P_4	110	56	16	0	3	2	2	2	2	2
P_5	169	100	48	12	0	0	0	0	1	1
P_6	255	168	96	36	8	0	0	0	1	1
P_7	418	304	204	112	54	20	0	0	1	1
P_8	594	456	330	209	126	70	15	0	2	2
P_9	750	594	475	345	221	143	63	20	0	0
P_{10}	1044	858	713	532	391	285	165	88	24	0

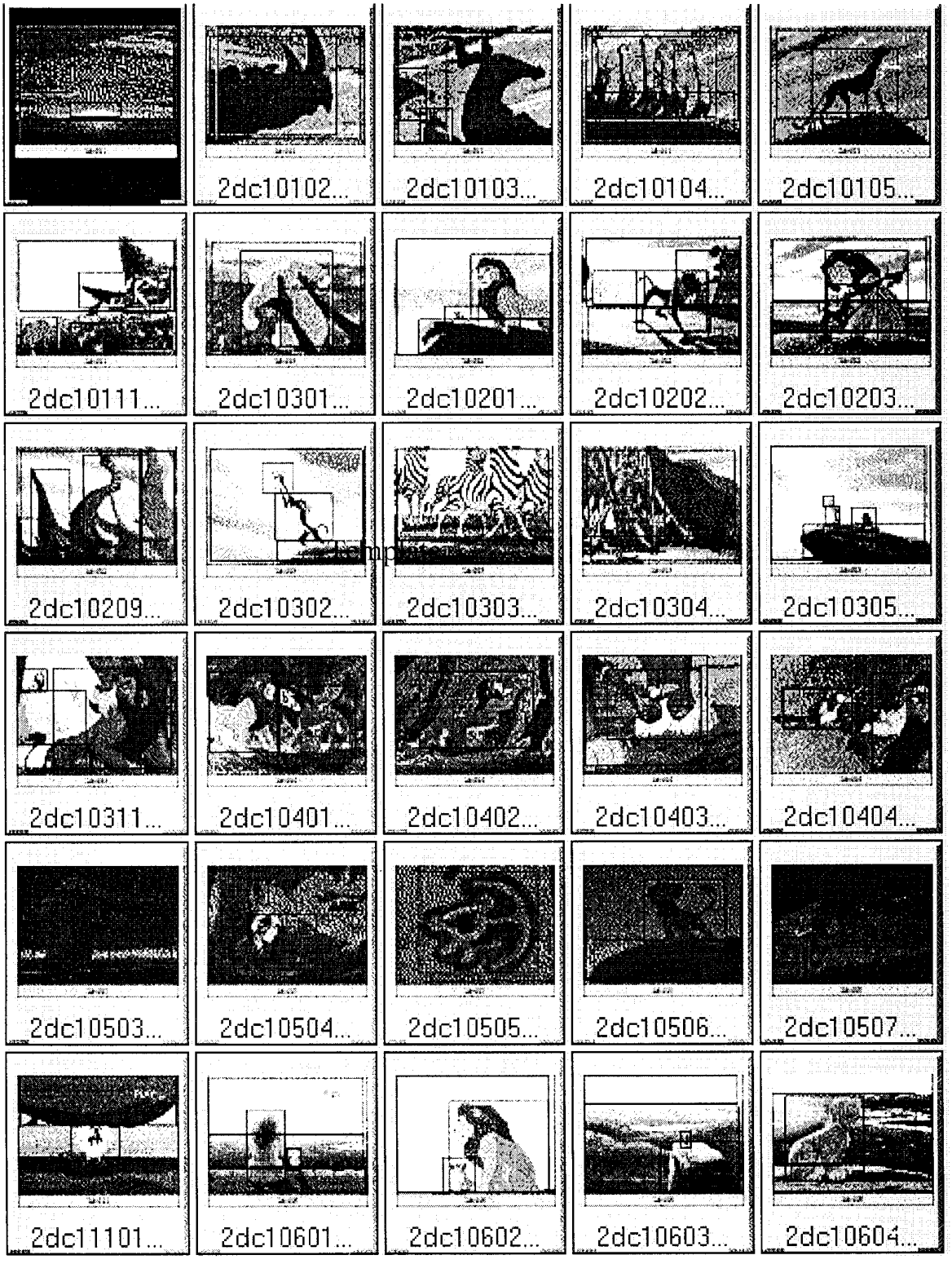


FIG. 12. Some key frames of the 48 streams collected.

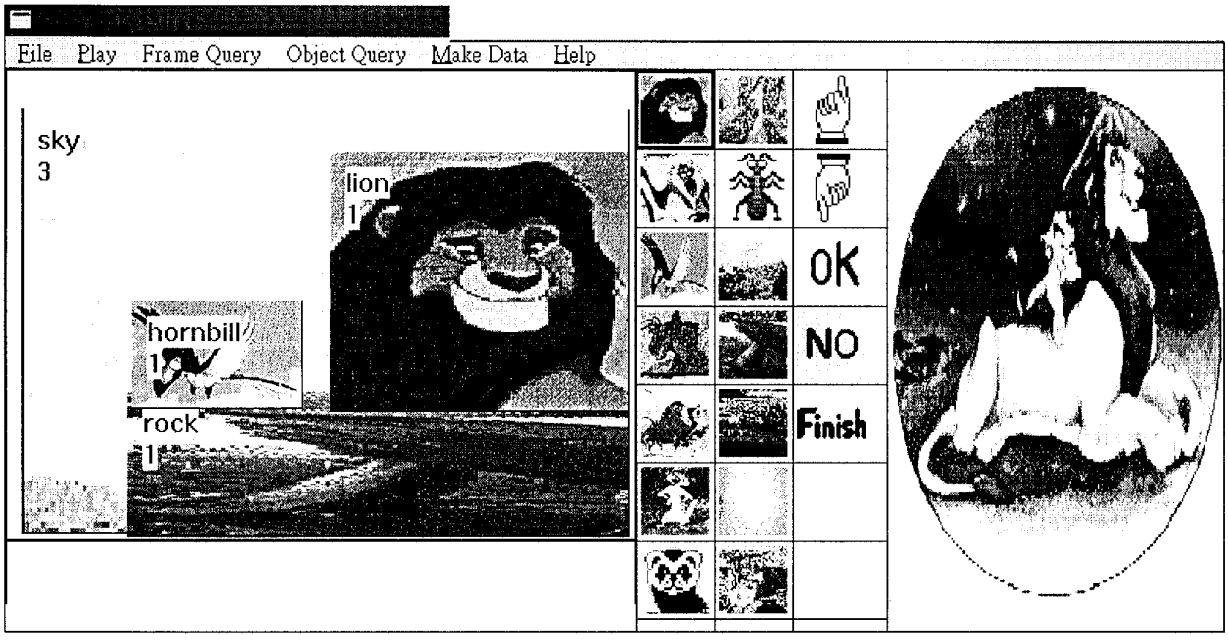


FIG. 13. An example query image.

along x- and y-axis directions independently and these 2D C-trees are stored associated with the source AVI file. The number of objects within each image implies the number of nodes in its corresponding 2D C-trees.

The system supports single image query and frame sequence query. The system allows users to draw a query image by assembling object icons designed or use a beforehand query template consisting of a sequence of query frames. An example query image is shown in Fig. 13. Two 2D C-trees of each image in the query sequence is constructed first. Then, for each stream of video database, we compute the partial distances between the stream and the query sequence. An approximate sequence matching (ASM) mechanism can compute the subsequence matching distance. The stream with minimum distance represents the most similar stream for the query sequence. A result of the query template in Fig. 13 is shown in Fig. 14. Our initial results validate the effectiveness of similarity retrieval by 2D C-trees matching.

9. CONCLUSIONS

Similarity retrieval is one of the attracting functions of an image database system that distinguishes it from traditional database systems. The goal is to retrieve the images that are similar to the query image. The similarity retrieval based upon the minimum-distance criterion had been proposed in the techniques of 2D string matching defined in terms of longest common subsequence. However, the 2D string representation is deficient in describing the spatial knowledge of the nonzero sized objects with overlapping. The similarity retrieval of images using 2D C-strings adopted the maximum-likelihood approach defined in terms of maximum degree of object-pairs clique. The algorithm for similarity retrieval based on 2D C-strings actually finds a maximum clique and becomes an NP-complete problem though polynomial algorithm for average case is available.

In this paper, we use an ordered labeled tree, 2D C-tree, to be the spatial representation for an image and propose the tree-matching algorithm for similarity retrieval. The algorithm provides a simple fast comparison for computing tree distance among images. The computation of distance between 2D C-trees can be used to measure the similarity of images with spatial constraint. This approach provides an effective and efficient mechanism for similarity retrieval in image databases. The tree distance comparison algorithm is also modified to compute the partial tree distance for subpicture query. We also validate the accountability of our tree-matching algorithms for similarity retrieval by simulation results. Moreover, the methodology of similarity retrieval is utilized in video sequence matching in our video information retrieval project being executed.

APPENDIX: LIST OF SYMBOLS

A_b	the begin-bound of object A
A_e	the end-bound of object A
R	the root of a 2D C-tree

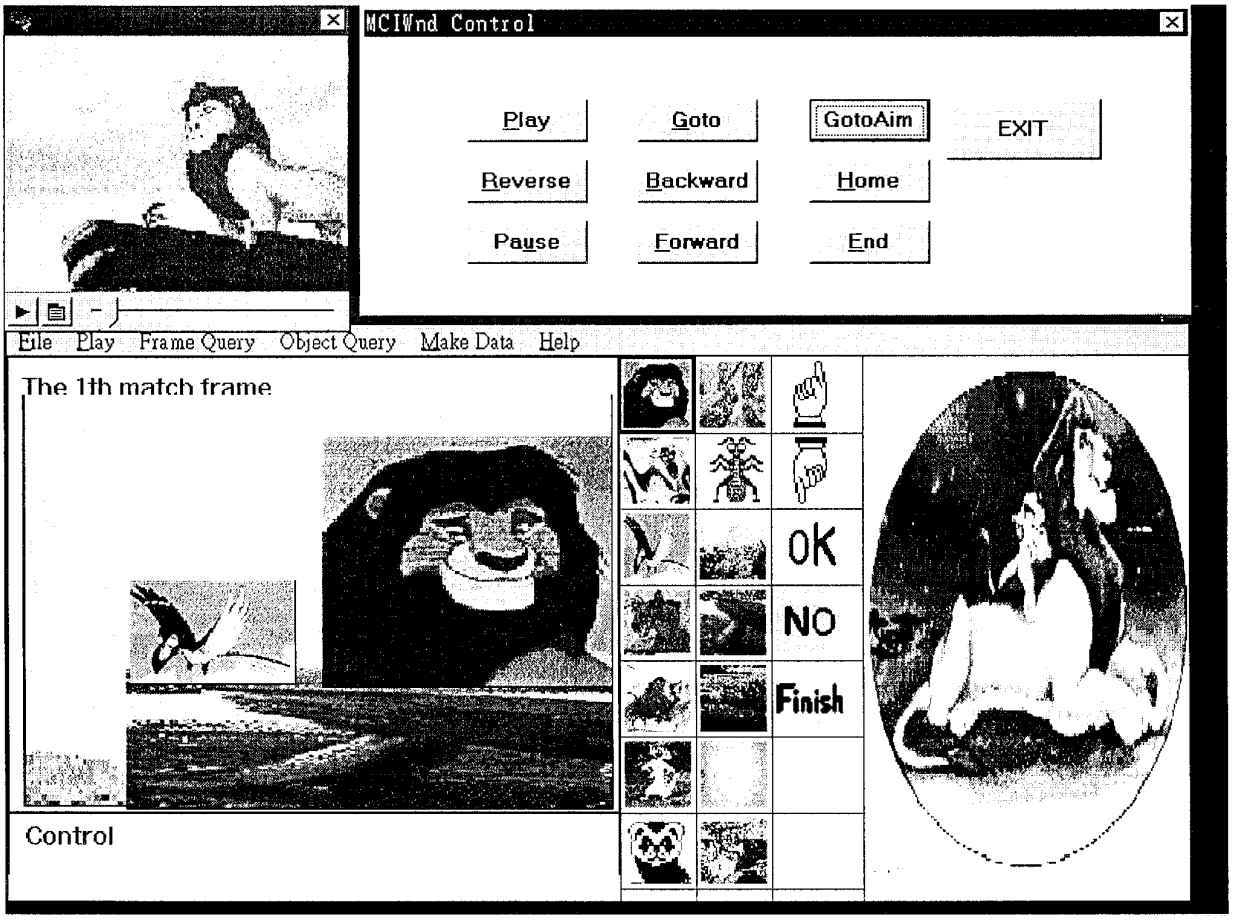


FIG. 14. A result of the query template in Fig. 13.

s_i	the ith immediate descendant of node S
ε	empty-node
$\mathscr{T}$	a rooted tree
$\mathscr{T}$[i]	the ith node of tree $\mathscr{T}$ in the postorder numbering
$e_1 \ldots e_k$	the editing operations
Δ	the cost function of editing operation
$\delta(\mathscr{T}_1, \mathscr{T}_2)$	the tree distance between tree $\mathscr{T}_1$ and tree $\mathscr{T}_2$
lca(i, j)	the least common ancestor node of $\mathscr{T}$[i] and $\mathscr{T}$[j]
N($\mathscr{T}$)	the number of symbols in tree $\mathscr{T}$
θ(i)	the postorder number of the leftmost leaf descendant in a subtree rooted at $\mathscr{T}$[i]
∂(i)	the depth of $\mathscr{T}$[i]
P(i)	the set of predecessors of $\mathscr{T}$[i]
P^k(i)	the kth level predecessor of $\mathscr{T}$[i]
$\mathscr{T}$[i . . j]	the ordered subforest of tree $\mathscr{T}$ induced by the nodes numbered from $\mathscr{T}$[i] to $\mathscr{T}$[j] inclusive
Forest(i)	an ordered subforest $\mathscr{T}$[1 . . i]
Tree(i)	a subtree of $\mathscr{T}$ rooted at $\mathscr{T}$[i]
Size(i)	the number of nodes in *Tree*(i)
$\lvert\mathscr{T}\rvert$	the number of nodes in the tree $\mathscr{T}$
depth($\mathscr{T}$)	the depth of the tree $\mathscr{T}$
leaves($\mathscr{T}$)	the number of leaf nodes in the tree $\mathscr{T}$
P_i	the ith picture in the database
$\delta(P_1, P_2)$	the tree distance between two pictures P_1 and P_2
$\gamma(P_1, P_2)$	the partial distance between two pictures P_1 and P_2

REFERENCES

1. S. K. Chang, *Principles of Pictorial Information Systems Design*, Prentice–Hall, Englewood Cliffs, NJ, 1989.
2. S. K. Chang, E. Jungert, and Y. Li, Representation and retrieval of symbolic pictures using generalized 2D strings, in *SPIE Proc. on Visual Communications and Image Processing, Philadelphia, 1989*, pp. 1360–1372.
3. S. K. Chang, Q. Y. Shi, and C. W. Yan, Iconic indexing by 2-D strings, *IEEE Trans. Pattern Anal. Mach. Intell.* **PAMI-9,** 1987, 413–428.
4. S. K. Chang, C. W. Yan, D. C. Dimitrof, and T. Arndt, Intelligent image database system, *IEEE Trans. Software Eng.* **14,** 1988, 681–688.

5. P. Ciaccia, F. Rabitti, and P. Zezula, Similarity search in multimedia database systems, in *Proceedings, The First International Conference on Visual Information Systems (VISUAL'96), Melbourne, Australia, 1996,* pp. 107–115.
6. M. Flickner, H. Sawhney, W. Niblack J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, Query by image and video content: The QBIC system, *IEEE Comput.* **44,** 1995, 23–32.
7. K. S. Fu, *Syntactic Pattern Recognition and Application,* Prentice–Hall, Englewood Cliffs, NJ, 1982.
8. V. N. Gudivada and V. V. Raghavan, Content-based image retrieval systems, *IEEE Comput.* **44,** 1995, 18–22.
9. A. Gupta, T. Weymouth, and R. Jain, Semantic queries with pictures: The VIMSYS model, in *Proceedings, The 17th International Conference on Very Large Data Bases, Barcelona, Spain, 1991,* pp. 69–79.
10. F. J. Hsu, S. Y. Lee, and P. S. Lin, 2D C-tree spatial representation for iconic image, in *Proceedings, The 2nd International Conference on Visual Information Systems, San Diego, CA, 1997,* pp. 287–294.
11. F. J. Hsu, S. Y. Lee, and P. S. Lin, Video data indexing by 2D C-trees, *J. Vis. Lang. Comput.,* submitted.
12. H. V. Jagadish, A. O. Mendelzon, and T. Milo, Similarity-based queries, in *Proceedings, The 14th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, San Jose, CA, 1995,* pp. 36–45.
13. S. Y. Lee and F. J. Hsu, 2D C-string: A new spatial knowledge representation for image database systems, *Pattern Recognit.* **23,** 1990, 1077–1087.
14. S. Y. Lee and F. J. Hsu, Spatial reasoning and similarity retrieval of images using 2D C-string knowledge representation, *Pattern Recognit.* **25,** 1992, 305–318.
15. S. Y. Lee, M. K. Shan, and W. P. Yang, Similarity retrieval of iconic image database, *Pattern Recognit.* **22,** 1989, 675–682.
16. U. Manber, *Introduction To Algorithms: A Creative Approach,* Addison–Wesley, Reading, MA, 1989.
17. W. Rickert, Extracting area objects from raster image data, *IEEE Comput. Graphics Appl.* **13,** 1993, 68–73.
18. A. Soffer and H. Samet, Pictorial queries by image similarity, in *Proceedings, The 13th International Conference on Pattern Recognition, Vienna, Austria, 1996,* pp. 114–119.
19. J. S. Wang, K. Zhang, K. Jeong, and D. Shasha, A system for approximate tree matching, *IEEE Trans. Knowledge Data Eng.* **6,** 1994, 559–571.
20. K. Wakimoto, M. Shima, S. Tanaka, and A. Maeda, Content-based retrieval applied to drawing image database, *SPIE* **1908,** 1993, 74–84.
21. J. K. Wu, A. D. Narasimhalu, B. M. Mehtre, C. P. Lam, and Y. J. Gao, CORE: A content-based retrieval engine for multimedia information systems, *ACM Multimedia Syst.* **3,** 1995, 25–41.
22. K. Zhang and D. Shasha, Simple fast algorithms for the editing distance between trees and related problems, *SIAM J. Comput.* **18,** 1989, 1245–1262.
23. K. Zhang, Algorithms for the constrained editing distance between ordered labeled trees and related problems, *Pattern Recognit.* **28,** 1995, 465–474.

FANG-JUNG HSU received his B.S. degree in computer science from the Soochow University and his M.S. degree in information science from the Chiao Tung University, Taiwan in 1982 and 1989, respectively. He is currently a Ph.D. candidate in computer science and information engineering at Chiao Tung University. He has been a senior engineer at the Computer Communication Research Laboratories (CCL) of the Industrial Technology Research Institute (ITRI), Taiwan, since 1984. His current research interests include multimedia information systems, image/spatial databases, object-oriented databases.

SUH-YIN LEE received her B.S.E.E. degree from the National Chiao Tung University, Taiwan in 1972 and her M.S. degree in computer science from the University of Washington, Seattle in 1975. She joined the faculty of the Department of Computer Engineering at Chiao Tung University in 1976 and received the Ph.D. degree in electronic engineering there in 1982. Dr. Lee is now a professor in the Department of Computer Science and Information Engineering at Chiao Tung University. She chaired the department from 1991 to 1993. Her current research interests include multimedia information systems, object-oriented databases, image/spatial databases, and computer networks. Dr. Lee is a member of Phi Tau Phi, the ACM, and the IEEE Computer Society.

BAO-SHUH LIN (S'76-M'79-SM'89) received the Ph.D. degree in computer science from the University of Illinois, Urbana, IL in 1980. He had been working as an R&D Manager of computer communication-related projects for AT&T Bell Laboratories, Racal Data Communications, Boeing, and Teknekron Communication Systems. He is currently the Deputy General Director of the Computer Communication Research Laboratories (CCL) of the Industrial Technology Research Institute (ITRI), Taiwan, R.O.C. He is the director of many advanced projects in CCL, including high-performance computer systems, multimedia systems, Chinese information technologies, and advanced information technologies. Dr. Lin is a senior member of IEEE Computer Society.