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

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

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.

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

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
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 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 can be translated into 2D string $A = B < C$, where the symbol ”;” can be omitted and is omitted.

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 $A = E < C$, where the symbol ”;” can be omitted and is omitted.

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 ”<,” “>,” ”=,” ”;”, 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 $(A|D|E|B|C, A|B|%|D|E)$. It is noted that all the objects in $f_2$ keep
TABLE 1

<table>
<thead>
<tr>
<th>Notation</th>
<th>Condition</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>A &lt; B</td>
<td>A&lt; B</td>
<td>A disjoints B.</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>A = B</td>
</tr>
<tr>
<td>A = B</td>
<td>A = B, and A = B</td>
<td>A is the same as B.</td>
</tr>
<tr>
<td>A [ B</td>
<td>A = B, and A &gt; B</td>
<td>A contains B and they have the same begin-bound.</td>
</tr>
<tr>
<td>A ] B</td>
<td>A &lt; B, and A = B</td>
<td>A contains B and they have the same end-bound.</td>
</tr>
<tr>
<td>A % B</td>
<td>A &lt; B, and A &gt; B</td>
<td>A contains B and they do not have the same bound.</td>
</tr>
<tr>
<td>A / B</td>
<td>A &lt; B &lt; A &lt; B</td>
<td>A is partly overlapping with B.</td>
</tr>
</tbody>
</table>

Note: The notations A< and A= (B< and B=) 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 q2 as in Fig. 2b, which is a subpicture of f2. The 2D C-string of the query image, (A%E < C, A < C < E) of q2, is quite different in the format from the 2D C-string of f2, due to the spatial operators. The string q2 is not a substring of the string f2 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.

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 s1, s2, ..., sn, S, being the parent, actually contains the local body consisting of all its immediate child-nodes s1, s2, ..., sn. The relation operator between node S and its first child-node s1 is surely a local operator that indicates the ensemble relationship between S and the local body consisting of all its child-nodes s1, s2, ..., sn. The relation operators between node S and other child-nodes si (2 ≤ i ≤ n) are definitely global operators that indicate the sibling relation between the child-node si (2 ≤ i ≤ n) and the prior child-node si−1 of node S. The signed 2D C-trees of f2 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
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 $α → β$, where $α$ is either $Λ$ (null) or a label in tree $T_1$ and $β$ is either $Λ$ or a label in tree $T_2$. We call $α → β$ a relabel operation if $α ≠ Λ$ and $β ≠ Λ$, a delete operation if $β = Λ$, and an insert operation if $α = Λ$. Let $Δ$ be a cost function which assigns a nonnegative real number, referred as $Δ(α → β)$, to each editing operation $α → β$. 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,

1. $Δ(α → β) ≥ 0$, $Δ(α → α) = 0$, positivity;
2. $Δ(α → β) = Δ(β → α)$, symmetry;
3. $Δ(α → γ) ≤ Δ(α → β) + Δ(β → γ)$, triangle inequality.

![FIG. 5. Three editing operations on labeled tree.](image-url)
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 $\Delta$ can be extended to the sequence of editing operations by letting

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

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 $\mathcal{T}_1$ and $\mathcal{T}_2$ is defined as

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

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 $\mathcal{T}[i]$ be the $i$th node of tree $\mathcal{T}$ in the postorder numbering. Formally, we define a triple $(M, \mathcal{T}_1, \mathcal{T}_2)$ to be a mapping from $\mathcal{T}_1$ to $\mathcal{T}_2$, where mapping $M$ is the set of integer pairs $(i, j)$ satisfying:

1. $1 \leq i \leq |\mathcal{T}_1|, 1 \leq j \leq |\mathcal{T}_2|$;
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. $\mathcal{T}_1[i_1]$ is to the left of $\mathcal{T}_1[j_1]$ if and only if $\mathcal{T}_2[j_1]$ is to the left of $\mathcal{T}_2[i_1]$ (sibling order preserved),
   c. $\mathcal{T}_1[i_1]$ is an ancestor of $\mathcal{T}_1[i_2]$ if and only if $\mathcal{T}_2[j_1]$ is an ancestor of $\mathcal{T}_2[j_2]$ (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 $M$, $\mathcal{T}_1[lca(i_1, i_2)]$ is a proper ancestor of $\mathcal{T}_1[i_3]$ if and only if $\mathcal{T}_2[lca(j_1, j_2)]$ is a proper ancestor of $\mathcal{T}_2[j_3]$.

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

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

Hence,

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

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].

Suppose that $A$ is a node in tree $\mathcal{T}$, $N(A)$ denotes the number of labels of node $A$. If $A$ is an empty-node, $A$ has a special label "e" 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 $\mathcal{T}_2$, 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 $A \rightarrow B$, $\Delta(A \rightarrow B)$, is defined as $N(B)$. That is, $\Delta(A \rightarrow B) = N(B)$.
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 as an example:

1. $\mathcal{T}[i]$. The $i$th node in the tree $\mathcal{T}$ according to the left-to-right postorder numbering. (Ex. The label of $\mathcal{T}[2]$ is E.)
2. $\theta(i)$. The number of the leftmost leaf descendant of the subtree rooted at $\mathcal{T}[i]$. When $\mathcal{T}[i]$ is a leaf node, $\theta(i) = 1$. (Ex. $\theta(4) = 2$; i.e., $E$ is the leftmost leaf descendant of $D$.)
3. $\vartheta(i)$. The depth of $\mathcal{T}[i]$; it is the number of nodes on the path from the root of tree $\mathcal{T}$ to node $\mathcal{T}[i]$, excluding $\mathcal{T}[i]$. (Ex. $\vartheta(2) = 3$; i.e., the depth of $E$ is 3.)
4. $\pi(i)$. The set of all the predecessors of $\mathcal{T}[i]$. Also,
\(P^k(i)\) denotes the \(k\)th level predecessor of \(T[i]\), where the level is counted from node \(T[i]\) backward to the root. 
\(P(i) = \{P^k(i) \mid 1 \leq k \leq \delta(i)\}.\) (Ex. \(P(2) = \{4, 5, 8\};\) i.e., the predecessors of \(E\) are \(D, A,\) and \(B\).)

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

(6) \(ForestDist\). An ordered subforest \(T\{1..i\}\). (Ex. \(ForestDist(4)\) includes \(e, E, e,\) and \(D\).)

(7) \(TreeDist\). A subtree of \(T\) rooted at \(T[i]\). \(T\{i..j\}\) will be referred to as \(Tree(i)\). (Ex. \(Tree(4)\) is equivalent to \(T\{2..4\}\).)

(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 \(T[i'..i]\) in \(T_1\) and \(T[j'..j]\) in \(T_2\). We use an abbreviated notation \(ForestDist(i, j)\) for the distance between \(T\{1..i\}\) and \(T\{1..j\}\) (see below).

(10) \(TreeDist(i, j)\). The distance between the subtree \(Tree(i)\) rooted at \(i\) in \(T_1\) and the subtree \(Tree(j)\) rooted at \(j\) in \(T_2\) (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) \(ForestDist(\emptyset, \emptyset) = 0.\)

(ii) \(ForestDist(\emptyset, i) = ForestDist(\emptyset, i - 1, \emptyset) + \Delta(T[i] \rightarrow \Lambda).\)

(iii) \(ForestDist(\emptyset, \emptyset, j) = ForestDist(\emptyset, \emptyset, j - 1) + \Delta(\Lambda \rightarrow T[j]).\)

Case (i) requires no editing operation and is assigned 0 for initialization. In (ii), the distance corresponds to the cost of deleting a node \(T[i]\) from a forest \(T[\emptyset, i]\) i.e., The forest \(T[\emptyset, i]\) is led by the leftmost leaf node of a tree \(Tree(i)\). (9) is referred to as \(TreeDist(i, j)\).

\(\text{LEMMA 2.}\) \(ForestDist(\emptyset, \emptyset, j) = TreeDist(i, j) = \min\)

\(ForestDist(\emptyset, i - 1, \emptyset, j) + \Delta(T[i] \rightarrow \Lambda),\)

\(ForestDist(\emptyset, i, \emptyset, j - 1) + \Delta(\Lambda \rightarrow T[j]),\)

\(ForestDist(\emptyset, i - 1, \emptyset, j - 1) + \Delta(T[i] \rightarrow T[j]).\)

\(\text{LEMMA 3.}\) Let \(i^p \in P(i)\) and \(j^p \in P(j)\). If \(\theta(i^p) \neq \theta(i)\), or \(\theta(j^p) \neq \theta(j)\), then

\(ForestDist(\emptyset, \emptyset, i, \emptyset, j) = \min\)

\(ForestDist(\emptyset, i - 1, \emptyset, j) + \Delta(T[i] \rightarrow \Lambda),\)

\(ForestDist(\emptyset, i, \emptyset, j - 1) + \Delta(\Lambda \rightarrow T[j]),\)

\(ForestDist(\emptyset, i - 1, \emptyset, j - 1) + \Delta(T[i] \rightarrow T[j]);\)

End.
DEFINITION 1. The distance between two pictures \( P_1 \) and \( P_2 \), denoted by \( \delta(P_1, P_2) \), is computed as the value of \( \text{TreeDist}(R_1, R_2) \). 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 \( P_1 \) and \( P_2 \), \( \mathcal{T}_1 \) and \( \mathcal{T}_2 \), along x-coordinate and y-coordinate, respectively, are constructed. We define the distance between \( P_1 \) and \( P_2 \) as follows.

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 \( f_2 \) to \( q_2 \). At least two editing operations are needed. That is, \( \Delta(D \rightarrow \lambda) \) and \( \Delta(B \rightarrow \varepsilon) \). So the tree distance of \( \delta(f_2, q_2) \) is 2. Three delete operations, \( \Delta(B \rightarrow \lambda), \Delta(D \rightarrow \lambda), \) and \( \Delta(\varepsilon \rightarrow \lambda) \), are needed between \( f_2 \) and \( q_2 \) along y-coordinate. That is, the cost of \( \delta(f_2, q_2) \) is 2. Finally, the distance of \( \delta(f_2, q_2) \) is 3.

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 efficiency of the algorithm is \( O(|\mathcal{T}| \times |\mathcal{T}|) \) * \( \min(\text{depth}(\mathcal{T}_1), \text{leaves}(\mathcal{T}_1)) \) * \( \min(\text{depth}(\mathcal{T}_2), \text{leaves}(\mathcal{T}_2)) \)). Let depth(\( \mathcal{T} \)) denote the depth of the tree \( \mathcal{T} \) and leaves(\( \mathcal{T} \)) denote the number of leaf nodes of the tree \( \mathcal{T} \). In general, the fast algorithm takes \( O(n^2) \) for computing the editing distance between two trees consisting of \( n \) nodes. The parallel algorithm is the time of complexity \( O(|\mathcal{T}_1| \times |\mathcal{T}_2|) \) by using \( O(min(|\mathcal{T}_1|, |\mathcal{T}_2|) \times \text{leaves}(\mathcal{T}_1) \times \text{leaves}(\mathcal{T}_2)) \) 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

\[
[P_1 | \delta(P_i, Q)] \text{ is the minimum of } \delta(P_k, Q), 1 \leq k \leq n.
\]

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 superflious 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 \( A \rightarrow \lambda \) is weighted zero; i.e., \( \Delta(A \rightarrow \lambda) = 0 \).

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

3. The cost of a relabel operation \( A \rightarrow B \), i.e., \( \Delta(A \rightarrow B) \), is slightly changed and is defined as the number of unmatched symbols in \( B \), which do not appear in \( A \). That is, \( \Delta(A \rightarrow B) = N(B/A) \). One special case is for \( B = \varepsilon \). The cost of \( \Delta(A \rightarrow \varepsilon) \) 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 \( \text{ForestDist}(\theta(\mathcal{T}) \ldots i, \emptyset) \) is always zero because \( \Delta(\mathcal{T}[i] \rightarrow \lambda) = 0 \). In Lemma 2 and Lemma 3, the
second value of the first statement in the minimum group, i.e., \( \Delta(\mathcal{T}_i[1] \rightarrow \Lambda) \), 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 \(\mathcal{T}_1\) and Tree\((y)\) rooted at \(y\) in tree \(\mathcal{T}_2\).

**Output:** The distance PartialTreeDist\((x, y)\).

\[
\begin{align*}
\text{ForestDist}(\varnothing, \varnothing) & = 0; \\
\text{for } i := \theta(x) \text{ to } x & \text{ begin} \\
\text{ForestDist}(\theta(x) \ldots i, \varnothing) & = 0; \\
\text{for } j := \theta(y) \text{ to } y & \text{ begin} \\
\text{ForestDist}(\varnothing, \theta(y) \ldots j) & = \text{ForestDist}(\varnothing, \theta(y) \ldots j - 1) + N(j); \\
\text{for } i := \theta(x) \text{ to } x & \text{ begin} \\
\text{for } j := \theta(x) \text{ to } y & \text{ begin} \\
\text{if } \theta(i) = \theta(x) \text{ and } \theta(j) = \theta(y) & \text{ then} \\
\text{ForestDist}(\theta(x) \ldots i, \theta(y) \ldots j) & = \min\{ \\
\text{ForestDist}(\theta(x) \ldots i - 1, \theta(y) \ldots j), \\
\text{ForestDist}(\theta(x) \ldots i, \theta(y) \ldots j - 1) + N(j), \\
\text{ForestDist}(\theta(x) \ldots i, \theta(y) \ldots j - 1) + \text{PartialTreeDist}(i, j)\}; \\
\text{else} & \text{ end} \\
\text{else} & \text{ end} \\
\text{end} \\
\end{align*}
\]

We use the example picture \(f_2\) in Fig. 2a and query sketch \(q_2\) in Fig. 2b to demonstrate the computation of partial distance. The 2D C-trees of \(f_2\) and \(q_2\) are in Fig. 4 and Fig. 7, correspondingly. For computing the partial tree distance between \(f_2\) and \(q_2\), 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 \(\gamma(f_2, q_2)\) 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 \(f_2\) to \(q_2\) along \(y\)-coordinate are all weighted zero. That is, the cost of \(\gamma(f_2, q_2)\) is 0 also. Finally, the distance of \(\gamma(f_2, q_2)\) is 0. It means that \(q_2\) is a subpicture of \(f_2\) with distance zero.

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

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

**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., \(P_3\) is always a subpicture of \(P_i\), for \(i \leq j\).

Then these 10 pictures are represented in 2D C-trees, respectively. The 2D C-trees of \(P_i\), referred to as \(\mathcal{T}_i\) and \(\mathcal{T}_{XY}\), are shown in Fig. 9 and the 2D C-trees of \(P_{10}\) 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 \(a_0(a_1a_2a_3 \ldots a_n)\) for a node \(a\), which has \(n\) immediate descendants in the ordering \(a_1, a_2, a_3, \ldots, a_n\). For example, \(\mathcal{T}_{XY}\) in Fig. 9 is represented

**Table 2**

The Simulation Data of Ten Objects

<table>
<thead>
<tr>
<th>Label</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
<tbody>
<tr>
<td>x-axis</td>
<td>131</td>
<td>347</td>
<td>165</td>
<td>358</td>
<td>31</td>
<td>641</td>
<td>192</td>
<td>572</td>
<td>5</td>
<td>358</td>
</tr>
<tr>
<td></td>
<td>192</td>
<td>572</td>
<td>5</td>
<td>358</td>
<td>192</td>
<td>572</td>
<td>5</td>
<td>358</td>
<td>192</td>
<td>572</td>
</tr>
<tr>
<td>y-axis</td>
<td>167</td>
<td>415</td>
<td>20</td>
<td>332</td>
<td>217</td>
<td>363</td>
<td>9</td>
<td>241</td>
<td>36</td>
<td>156</td>
</tr>
<tr>
<td></td>
<td>217</td>
<td>363</td>
<td>9</td>
<td>241</td>
<td>217</td>
<td>363</td>
<td>9</td>
<td>241</td>
<td>217</td>
<td>363</td>
</tr>
</tbody>
</table>

For illustrating the computations of tree distances among these 10 pictures, the 2D C-tree representation is expressed by a recursive sentence \(a_0(a_1a_2a_3 \ldots a_n)\) for a node \(a\), which has \(n\) immediate descendants in the ordering \(a_1, a_2, a_3, \ldots, a_n\). For example, \(\mathcal{T}_{XY}\) in Fig. 9 is represented.
FIG. 8. $P_3$ is a subpicture of $P_4$.

The 2D C-trees of $P_4$ as $R(C(\varepsilon A(eB)B)e)$. Note that $R$, $\varepsilon$ 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, $P_1, P_2, \ldots, P_{10}$, as reference pictures. We also use $P_i, 1 \leq i \leq 10$, 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 $\delta$ (reference, query). Table 4 shows the subpicture query and we compute the partial tree distance $\gamma$(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 $P_k$,

$\begin{align*}
(i) & \quad \delta(P_i, P_j) \geq 0, \text{ and } \delta(P_i, P_i) = 0 \text{ (positivity)}, \\
(ii) & \quad \delta(P_i, P_j) = \delta(P_j, P_i) \text{ (symmetry)}, \\
(iii) & \quad \delta(P_i, P_k) \leq \delta(P_i, P_j) + \delta(P_j, P_k) \text{ (triangle inequality)}. 
\end{align*}$

(2) For any $P_j$,

$\begin{align*}
(i) & \quad \text{if } i < j, \text{ then } \delta(P_i, P_j) < \delta(P_{i-1}, P_j); \\
(ii) & \quad \text{if } k > j, \text{ then } \delta(P_k, P_j) < \delta(P_{k-1}, P_j). 
\end{align*}$

$P_i$ contains the first $j$ objects. $P_j$ 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 $i$th 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 $P_1$, the distance between $P_k$ and $P_j$ is always smaller than the distance between $P_k$ and $P_{k+1}$. 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

$\begin{align*}
\sum_{i=1}^{10} d(P_i, P_j), \\
\sum_{j=1}^{10} d(P_i, P_j), \\
\sum_{k=1}^{10} d(P_k, P_j), \\
\sum_{l=1}^{10} d(P_l, P_j), \\
\sum_{m=1}^{10} d(P_m, P_j), \\
\sum_{n=1}^{10} d(P_n, P_j), \\
\sum_{p=1}^{10} d(P_p, P_j), \\
\sum_{q=1}^{10} d(P_q, P_j), \\
\sum_{r=1}^{10} d(P_r, P_j), \\
\sum_{s=1}^{10} d(P_s, P_j). 
\end{align*}$

FIG. 10. The 2D C-trees of $P_3$.

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

FIG. 11. The 2D C-tree representations of 10 simulation pictures.
TABLE 3
The Tree Distance Result of Comparing Ten Simulation Pictures

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

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 non-zero 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, $P_j$ is viewed as a subpicture of $P_i$. 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 images 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

<table>
<thead>
<tr>
<th>Reference</th>
<th>$P_1$</th>
<th>$P_2$</th>
<th>$P_3$</th>
<th>$P_4$</th>
<th>$P_5$</th>
<th>$P_6$</th>
<th>$P_7$</th>
<th>$P_8$</th>
<th>$P_9$</th>
<th>$P_{10}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$P_2$</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$P_3$</td>
<td>42</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$P_4$</td>
<td>110</td>
<td>56</td>
<td>16</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$P_5$</td>
<td>169</td>
<td>100</td>
<td>48</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$P_6$</td>
<td>255</td>
<td>168</td>
<td>96</td>
<td>36</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$P_7$</td>
<td>418</td>
<td>304</td>
<td>204</td>
<td>112</td>
<td>54</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>$P_8$</td>
<td>594</td>
<td>456</td>
<td>330</td>
<td>209</td>
<td>126</td>
<td>70</td>
<td>15</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>$P_9$</td>
<td>750</td>
<td>594</td>
<td>475</td>
<td>345</td>
<td>221</td>
<td>143</td>
<td>63</td>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$P_{10}$</td>
<td>1044</td>
<td>858</td>
<td>713</td>
<td>551</td>
<td>391</td>
<td>285</td>
<td>165</td>
<td>88</td>
<td>24</td>
<td>0</td>
</tr>
</tbody>
</table>
FIG. 12. Some key frames of the 48 streams collected.
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 $i$th immediate descendant of node $S$
- $e$: empty-node
- $\mathcal{T}$: a rooted tree
- $\mathcal{T}[i]$: the $i$th node of tree $\mathcal{T}$ in the postorder numbering
- $e_1 \ldots e_k$: the editing operations
- $\Delta$: the cost function of editing operation
- $\delta(\mathcal{T}_1, \mathcal{T}_2)$: the tree distance between tree $\mathcal{T}_1$ and tree $\mathcal{T}_2$
- $\text{lca}(i, j)$: the least common ancestor node of $\mathcal{T}[i]$ and $\mathcal{T}[j]$
- $\text{N}(\mathcal{T})$: the number of symbols in tree $\mathcal{T}$
- $\theta(i)$: the postorder number of the leftmost leaf descendant in a subtree rooted at $\mathcal{T}[i]$
- $\delta([i])$: the depth of $\mathcal{T}[i]$
- $P(i)$: the set of predecessors of $\mathcal{T}[i]$
- $P^k(i)$: the $k$th level predecessor of $\mathcal{T}[i]$
- $\mathcal{T}[i \ldots j]$: the ordered subforest of tree $\mathcal{T}$ induced by the nodes numbered from $\mathcal{T}[i]$ to $\mathcal{T}[j]$ inclusive
- $\text{Forest}(i)$: an ordered subforest $\mathcal{T}[1 \ldots i]$
- $\text{Tree}(i)$: a subtree of $\mathcal{T}$ rooted at $\mathcal{T}[i]$
- $\text{Size}(i)$: the number of nodes in $\text{Tree}(i)$
- $|\mathcal{T}|$: the number of nodes in the tree $\mathcal{T}$
- $\text{depth}(\mathcal{T})$: the depth of the tree $\mathcal{T}$
- $\text{leaves}(\mathcal{T})$: the number of leaf nodes in the tree $\mathcal{T}$
- $P_i$: the $i$th 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


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.