International Journal of Image and Graphics
Vol. 3, No. 1 (2003) 31–53
© World Scientific Publishing Company

SIMILARITY-BASED SUBSEQUENCE SEARCH IN IMAGE SEQUENCE DATABASES*

SANGHYUN PARK

Department of Computer Science and Engineering
Pohang University of Science and Technology (POSTECH)
San 31, Hyoja-dong, Pohang 790-784, Korea
sanghyun@postech.ac.kr

WESLEY W. CHU

Department of Computer Science, University of California, Los Angeles,
Los Angeles, CA 90095, USA
wwc@cs.ucla.edu

Received 15 June 2002
Revised 8 October 2002

This paper proposes an indexing technique for fast retrieval of similar image subsequences using the multi-dimensional time warping distance. The time warping distance is a more suitable similarity measure as compared to the L_p distance in many applications where sequences may be of different lengths and/or different sampling rates. Our indexing scheme employs a disk-based suffix tree as an index structure and uses a lower-bound distance function to filter out dissimilar subsequence without false dismissals. It applies the normalization for an easier control of relative weighting of feature dimensions and the discretization to compress the index tree. Experiments on medical and synthetic image sequences verified that the proposed method significantly outperforms the naïve method and scales well in a large volume of image sequence databases.

Keywords: Similarity search; Image sequence database; Time warping; Suffix tree.

1. Introduction

An image sequence database is a set of image sequences, each of which is an ordered list of images. A series of lung images of a patient, a set of picture images taken by panorama cameras, and consecutive frames of video clips are the typical examples of image sequences.

Similarity search is an operation that finds sequences or subsequences whose changing patterns are similar to that of a given query sequence.[1–3] Similarity search is of growing importance in many new applications such as data mining, data warehousing and digital image/video libraries.[4,5] Especially in the medical domain, a

*This research is supported in part by NIC/NIH Grant #442511-33780.

search for patients with similar temporal characteristics can augment the process of patient care by providing physicians with the insight into the treatment of previous patients with similar medical conditions. For example, an oncologist might search the database for patients with similar tumor evolution patterns to find the optimal course of treatment.

Similarity search is classified into whole matching and subsequence matching.[1] Assuming that all the data and query sequences have the same length, whole matching searches for the data sequences similar to a query sequence. Subsequence matching searches for the subsequences, contained in data sequences, which are similar to a query sequence of arbitrary length.

The naïve method for similarity search reads each image sequence or subsequence sequentially from the database and computes its distance to a query image sequence. This method is simple but suffers from severe performance degradation when the database is large. Therefore an effective indexing scheme is essential as a scalable solution for similarity search.

Finding a similarity measure for sequences is not easy because sequences that are qualitatively the same may be quantitatively different. Therefore the previous approaches often fail to retrieve some of the similar data sequences when employing only the Euclidean distance as a similarity measure. Thus recent work on similarity search tends to support various types of transformations such as scaling,[2,6] shifting,[2,6] normalization,[7,8] and time warping.[9–13]

Time warping[9,14] is a transformation that allows any sequence element to replicate itself as many times as needed without extra costs. The time warping distance is defined as the smallest distance between two sequences transformed by time warping. While the Euclidean distance can be used only when two sequences compared are of the same length, the time warping distance can be applied to any two sequences of arbitrary lengths. Therefore the time warping distance fits well with the image sequence databases where sequences are of different lengths or having different interval lengths between elements.

For the efficient processing of similarity search, most of the previous approaches[1–3] map each sequence or subsequence into a multi-dimensional point and then compute L_p distance metric between multi-dimensional points to filter out dissimilar sequences in the index space. Yi *et al.*[13] claimed that the multi-dimensional indexes, assuming the triangular inequality, directly or indirectly cause *false dismissal* in similarity search when their underlying distance functions do not satisfy the triangular inequality. False dismissal[1,2] is defined as a miss in a part of the final query result. Yi *et al.*[13] proved that the time warping distance does not satisfy the triangular inequality. Therefore the multi-dimensional indexes that assume the triangular inequality could not work with the time warping distance in many applications that do not permit false dismissal.

In our earlier work,[11] we suggested an efficient subsequence matching approach with the time warping distance as a similarity measure, assuming that each sequence element has a single numeric value. Our approach employed a suffix tree,[15] (which does not presume the triangular inequality) as an index structure, and applied the discretization of element values in order to reduce the index size. While traversing the suffix tree, our method computed the lower-bound distance to retrieve all the similar subsequences without any false dismissal.

This paper extends our earlier work[11] to handle the similarity-based subsequence search in the image sequences, where multiple numeric values (i.e. feature vector) represent a single element. The suffix tree is employed again as an index structure and the idea of the discretization is applied once more for index compression. The challenges of this extension are: (1) how to define the multi-dimensional time warping distance metric with the consideration of relative weighting, (2) how to discretize a multi-dimensional element into a single symbol, and (3) how to define a lower-bound multi-dimensional time warping distance metric which can be employed by an index traversal algorithm to filter out dissimilar subsequences without false dismissal.

Section 2 describes the notations used in this paper and presents the multi-dimensional time warping distance metric. Sections 3 and 4 present the indexing and the query processing algorithms of the proposed indexing scheme. The proposed method is applied to the medical database system in Sec. 5 and then evaluated in terms of performance and scalability in Sec. 6. Section 7 describes the related work and Sec. 8 concludes the paper.

2. Definition

This section first describes the notations used in this paper and then gives the definitions of various distance functions. Finally it presents the formal definition of the target problem we are going to solve.

2.1. *Notation*

We use the notation $X = \langle X[1], \ldots, X[n] \rangle$ for a single dimensional sequence with n elements. $X[i]$ denotes the ith element of X and $|X|$ denotes the number of elements in X. $|X|$ also represents the length of X. $X[i:j]$ is a subsequence of X containing elements in positions i through j. $X[i:-]$ is a subsequence of X starting at the ith element position and ending at the last element position. That is, $X[i:-] = X[i:|X|]$. $X[i:-]$ also represents the suffix of X starting at the ith position. $\langle\,\rangle$ denotes an empty sequence.

Sequences of numeric elements can be converted into sequences of discrete symbols by the discretization. X^{C} denotes a sequence of discrete symbols converted from X. Then, $X^{\mathrm{C}}[i]$ is the ith symbol of X^{C} and $|X^{\mathrm{C}}|$ is the number of elements in X^{C}. The meanings of $X^{\mathrm{C}}[i:j]$ and $X^{\mathrm{C}}[i:-]$ are analogous to those of $X[i:j]$ and $X[i:-]$.

We use the bold font for multi-dimensional sequences. That is, $\boldsymbol{X} = \langle \boldsymbol{X}[1], \ldots, \boldsymbol{X}[n] \rangle$ represents a multi-dimensional sequence with n elements. $\boldsymbol{X}[i] = (\boldsymbol{X}[i][1], \ldots, \boldsymbol{X}[i][k])$ denotes the ith element of $\boldsymbol{X}$ with k feature values. We assume

January 10, 2003 16:8 WSPC/164-IJIG 00090

that every element of multi-dimensional sequences has the same number of feature values. Notice that an image sequence is an instance of a multi-dimensional sequence.

2.2. *Distance function*

2.2.1. *L_p distance*

The L_p distance function has been widely used to measure the similarity of any two sequences X and Y. L_1 is the Manhattan distance, L_2 is the Euclidean distance, and L_∞ is the maximum distance in any pair of elements.[16] The L_p function requires the two sequences to be compared and also similar in length.

$$L_p(X, Y) = \left(\sum_{i=1}^{|X|} |X[i] - Y[i]|^p \right)^{1/p}, \quad 1 \le p \le \infty\,.$$

2.2.2. *Time warping distance*

In general, finding a similarity measure for sequences is not easy because sequences that are qualitatively the same may be quantitatively different. First, the sequences may be of different lengths, making it difficult or impossible to embed them in a metric space and then using the Euclidean distance to measure their similarity. Second, the sampling rates of the sequences may be different, making similarity measures such as cross-correlation useless. In the area of speech recognition,[14] this problem has been approached using a similarity measure, called the time warping distance.[9,14]

Time warping is a generalization of classical algorithms for comparing discrete sequences with continuous values.[14] To find the minimum difference between two sequences, time warping enables each element of a sequence to match one or more neighboring elements of another sequence.

Definition 1. Given any two sequences X and Y, the time warping distance D_{tw} is defined recursively as follows[14]:

$$D_{\text{tw}}(\langle\rangle, \langle\rangle) = 0\,,$$

$$D_{\text{tw}}(X, \langle\rangle) = D_{\text{tw}}(\langle\rangle, Y) = \infty\,,$$

$$D_{\text{tw}}(X, Y) = |X[1] - Y[1]| + \min \begin{cases} D_{\text{tw}}(X, Y[2:-]) \\ D_{\text{tw}}(X[2:-], Y) \\ D_{\text{tw}}(X[2:-], Y[2:-])\,. \end{cases}$$

$D_{\text{tw}}(X, Y)$ can be efficiently calculated using a dynamic programming technique[9] based on the recurrence relation. The dynamic programming algorithm[9] fills in the table T of cumulative distances as the computation proceeds. The final

Table 1. Cumulative distance table for $X = \langle 4, 5, 6, 7, 6, 6\rangle$ and $Y = \langle 3, 4, 3\rangle$.

row6	**6**	16	11	*12*
row5	**6**	13	9	10
row4	**7**	10	7	8
row3	**6**	6	4	5
row2	**5**	3	2	3
row1	**4**	1	1	2
	X/Y	**3**	**4**	**3**
		column1	column2	column3

cumulative distance, $T[|X|][|Y|]$, is the minimum distance between X and Y, and the matching of elements can be traced backward in the table by choosing the previous cells with the lowest cumulative distance. This distance computation has the complexity $O(|X|\,|Y|)$. Table 1 shows the cumulative distance table for two sequences $X = \langle 4, 5, 6, 7, 6, 6\rangle$ and $Y = \langle 3, 4, 3\rangle$. Here $D_{\text{tw}}(X, Y) = 12$ because $T[|X|][|Y|] = T[6][3] = 12$.

2.2.3. *Multi-dimensional time warping distance*

Before proposing the multi-dimensional time warping distance function, let us consider the weighted distance function D_{mbase} for any two multi-dimensional elements $\boldsymbol{X}[i]$ and $\boldsymbol{Y}[j]$ with k features:

$$D_{\text{mbase}}(\boldsymbol{X}[i], \boldsymbol{Y}[j]) = \sum_{h=1}^{k} W_h * |\boldsymbol{X}[i][h] - \boldsymbol{Y}[j][h]|\,.$$

Here W_h is the weight of the hth feature dimension. With D_{mbase} as a distance metric for any two elements, the multi-dimensional time warping distance for any two multi-dimensional sequences $\boldsymbol{X}$ and $\boldsymbol{Y}$ is defined as follows:

Definition 2. For any two multi-dimensional sequences, $\boldsymbol{X}$ and $\boldsymbol{Y}$, the multi-dimensional time warping distance between $\boldsymbol{X}$ and $\boldsymbol{Y}$ is defined as follows:

$$D_{\text{mtw}}(\langle\,\rangle, \langle\,\rangle) = 0\,,$$

$$D_{\text{mtw}}(\boldsymbol{X}, \langle\,\rangle) = D_{\text{mtw}}(\langle\,\rangle, \boldsymbol{Y}) = \infty\,,$$

$$D_{\text{mtw}}(\boldsymbol{X}, \boldsymbol{Y}) = D_{\text{mbase}}(\boldsymbol{X}[1], \boldsymbol{Y}[1]) + \min \begin{cases} D_{\text{mtw}}(\boldsymbol{X}, \boldsymbol{Y}[2:-]) \\ D_{\text{mtw}}(\boldsymbol{X}[2:-], \boldsymbol{Y}) \\ D_{\text{mtw}}(\boldsymbol{X}[2:-], \boldsymbol{Y}[2:-])\,. \end{cases}$$

$D_{\text{mtw}}(\boldsymbol{X}, \boldsymbol{Y})$ can be calculated with the computation complexity $O(k|\boldsymbol{X}|\,|\boldsymbol{Y}|)$ using a dynamic programming technique[9] based on the recurrence relation.

2.3. *Problem definition*

The primary goal of this paper is to propose an efficient indexing technique for similarity-based subsequence search in image sequence databases. The formal definition of the problem is as follows:

Given a set of image data sequences of arbitrary lengths, an image query sequence $\boldsymbol{q}$, and a distance tolerance ε, find those data subsequences $\boldsymbol{x}$ which satisfy $D_{\mathrm{mtw}}(\boldsymbol{x}, \boldsymbol{q}) \leq \varepsilon$.

Additional types of queries include the nearest neighbor queries (e.g. "find the five image sequences most similar to a given image sequence") and the "all pairs" queries (e.g. "report all pairs of image sequences that are within the distance ε from each other"). Both types of queries can be handled by our approach using a branch-and-bound algorithm[17] together with a spatial join algorithm.[18]

3. Indexing

The indexing step builds an index from a set of image sequences. As shown in Fig. 1, this step is further divided into the pre-processing and the index construction. The pre-processing step transforms a set of raw image sequences into a set of discrete symbol sequences through the successive processes of segmentation, feature extraction, normalization, and discretization. The index construction step builds a disk-based suffix tree incrementally from a set of discrete symbol sequences by performing a series of binary merges of suffix trees.

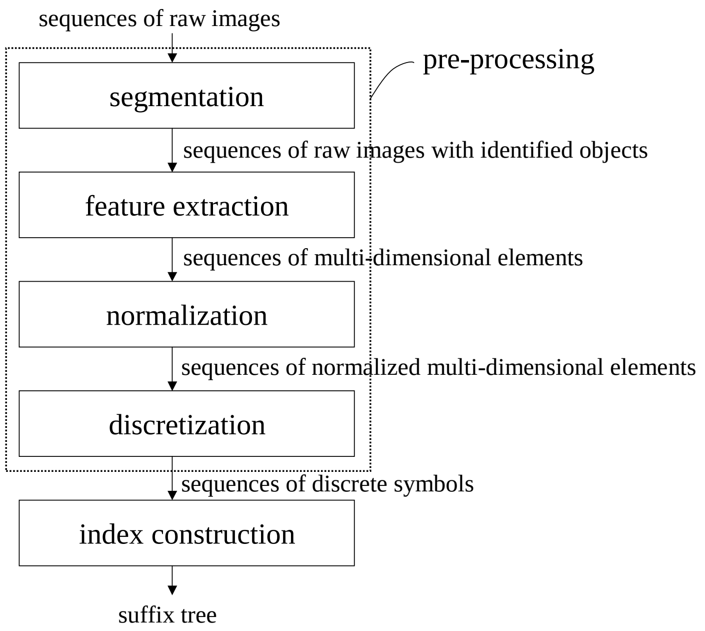

sequences of raw images
pre-processing
segmentation
sequences of raw images with identified objects
feature extraction
sequences of multi-dimensional elements
normalization
sequences of normalized multi-dimensional elements
discretization
sequences of discrete symbols
index construction
suffix tree

Fig. 1. The steps for building an index from a set of raw image sequences.

January 10, 2003 16:8 WSPC/164-IJIG 00090

3.1. *Segmentation and feature extraction*

The segmentation is to detect the boundaries of objects from the background and the feature extraction is to compute a representative feature vector from the identified objects. As an example, let us consider a sequence of brain images containing a tumor. We may extract the following three features from each tumor object: (*location*, *size*, *perimeter*). Then, the image sequence $\boldsymbol{X}$ is denoted as $\langle(\mathit{location}[1], \mathit{size}[1], \mathit{perimeter}[1]), \ldots, (\mathit{location}[n], \mathit{size}[n], \mathit{perimeter}[n])\rangle$ where n is the number of images in $\boldsymbol{X}$.

Since the detailed description on segmentation and feature extraction is beyond the scope of this paper, we omit the further discussion on them. Interested readers may refer to the Refs. 19–22.

3.2. *Normalization*

The purpose of the normalization process is to make it easier for users to assign or control the relative weighting of the feature dimensions. Without normalization, the feature dimensions with higher average values may have more influence in determining the similarity of any two sequences. Thus the normalization enables every feature dimension to have the same data value distribution (i.e. normal distribution).

3.3. *Discretization*

The discretization process converts a feature vector into the corresponding symbol in order to make the index structure compact and thus accelerate the query processing. For discretization, we will first generate a set of categories from a set of normalized multi-dimensional elements. Using the categorization method of multiple-attribute type abstraction hierarchy (MTAH),[23] we classify similar multi-dimensional elements into the same category and then assign a unique symbol to each category. MTAH is a data-driven multiple-level categorization hierarchy that uses *relaxation error* as a goodness measure of categories. MTAH has the following benefits: (1) The algorithm considers both value and frequency distributions; therefore it is more accurate than the equal-length interval categorization, and (2) MTAH is easier to implement as compared to the maximum-entropy categorization method. A category from the k-dimensional elements is represented as $C = ([C[1]\cdot\min, C[1]\cdot\max], [C[2]\cdot\min, C[2]\cdot\max], \ldots, [C[k]\cdot\min, C[k]\cdot\max])$. After obtaining a set of categories, we convert each element into the symbol of the corresponding category. We use the notation $\boldsymbol{X}^{\mathrm{C}}$ to denote the sequence discretized from $\boldsymbol{X}$.

3.4. *Index construction*

Once we have converted the sequences of raw images into sequences of symbols, we propose to use the suffix tree as an index structure for fast subsequence matching.

The suffix tree has the benefits such that (1) it is a good structure especially for subsequence matching since all possible suffixes of the given sequence are maintained in the tree, and (2) it does not assume any geometry or any distance function. Thus it guarantees the absence of false dismissals even with the time warping distance if the distance metric used in the index space is a lower-bound function of the time warping distance.

Let us present the definition and the internal structure of the suffix tree. A *trie* is an indexing structure used for indexing a set of keywords. A *suffix trie*[15] is a trie whose set of keywords comprises the suffixes of sequences. Nodes with a single outgoing edge can be collapsed, yielding the structure known as the *suffix tree.*[15] Each suffix of a sequence is represented by a leaf node. More specifically, $X[i:-]$ is expressed by a leaf node labeled with $(ID(X), i)$, where $ID(X)$ is the identifier of X and i is the offset from which the suffix starts. The edges are labeled with subsequences such that the concatenation of the edge labels on the path from the root to the leaf $(ID(X), i)$ becomes $X[i:-]$. The concatenation of the edge labels on the path from the root to the internal node, N, represents the longest common prefix of the suffixes represented by the leaf nodes under N. We use the notation $label(N_1, N_2)$ for the concatenated labels on the path from N_1 to N_2.

To build the suffix tree from multiple sequences, we use an incremental disk-based suffix tree construction method.[24] Two suffix trees, representing two disjoint sets of sequences, are merged to produce a single suffix tree by performing the pre-order traversal on both trees and combining the paths corresponding to common

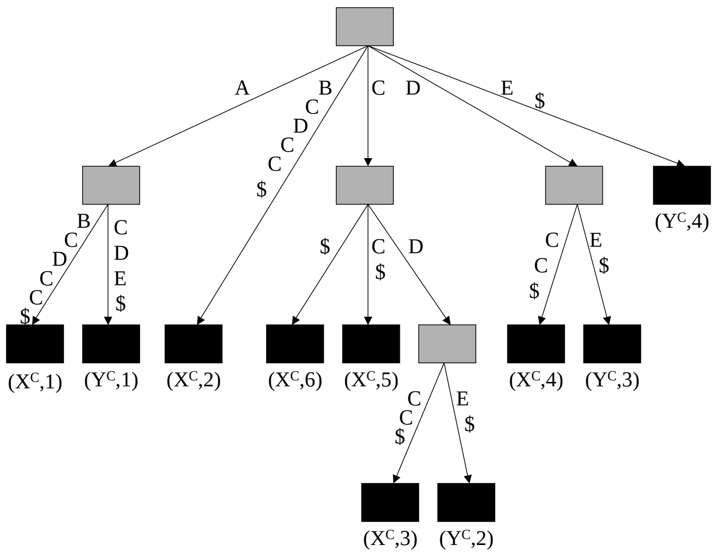


Fig. 2. Suffix tree constructed from $\boldsymbol{X}^{\mathrm{C}} = \langle A, B, C, D, C, C\rangle$ and $\boldsymbol{Y}^{\mathrm{C}} = \langle A, C, D, E\rangle$. Six suffixes $(\langle A, B, C, D, C, C\rangle, \langle B, C, D, C, C\rangle, \langle C, D, C, C\rangle, \langle D, C, C\rangle, \langle C, C\rangle, \langle C\rangle)$ from $\boldsymbol{X}^{\mathrm{C}}$ and four suffixes $(\langle A, C, D, E\rangle, \langle C, D, E\rangle, \langle D, E\rangle, \langle E\rangle)$ from $\boldsymbol{Y}^{\mathrm{C}}$ are extracted and then inserted into the suffix tree. \$ denotes an end marker of a suffix.

subsequences. The construction of the suffix tree from m data sequences, whose average length is $\bar{L}$, has algorithmic complexity of $O(m\bar{L})$.

Figure 2 shows the suffix tree constructed from two symbol sequences, $\boldsymbol{X}^{\mathrm{C}} = \langle A, B, C, D, C, C\rangle$ and $\boldsymbol{Y}^{\mathrm{C}} = \langle A, C, D, E\rangle$ where \$ denotes an end marker of a suffix.

4. Query Processing

When a query image sequence is submitted, it is first converted into the corresponding multi-dimensional sequence $\boldsymbol{q}$ using segmentation and feature extraction. Then the suffix tree is traversed from the root to retrieve a set of candidate subsequences whose lower-bound distances to the query sequence are within the distance tolerance ε.

Since the lower-bound time warping distance is used for filtering, the dissimilar subsequences whose actual time warping distances are larger than ε may be included in the candidate set. These subsequences are called *false alarms*.[1,2] Therefore the proposed algorithm applies the post-processing on the set of candidate subsequences. That is, it retrieves the corresponding subsequences from the database and computes their time warping distances using D_{mtw}. Finally, the subsequences whose actual time warping distances that are not larger than ε are returned as the final answers.

4.1. *Index traversal algorithm*

The proposed index traversal algorithm is shown in Algorithm 1. Starting from the root, the algorithm visits each node in the depth-first order. When the algorithm

Algorithm 1. Index Traversal Algorithm

```
Input  : node $N$, query sequence $\boldsymbol{q}$, distance tolerance $\varepsilon$, cumulative dis-
         tance table $T$
Output: candidateSet

candidateSet ← {};
$CN$ ← GetChildren($N$);

for $i \leftarrow 1$ to $|CN|$ do
    $CT_i$ ← AddRow($T$, $\boldsymbol{q}$, label($N, CN_i$), $D_{mtw-lb}$);
    Let $dist$ be the rightmost column value of the newly added row;
    Let $minDist$ be the minimum column value of the newly added row;
    if $dist \leq \varepsilon$ then
        candidateSet ← candidateSet ∪ {label(root, $CN_i$)};
    if $minDist \leq \varepsilon$ then
        candidateSet ← candidateSet ∪ IndexTraversal ($CN_i$, $\boldsymbol{q}$, $\varepsilon$, $CT_i$);

return candidateSet;
```

visits a node N, it inspects each child node CN_i to find a new candidate and to determine whether the visiting of the subtree of CN_i is needed. For simpler explanation, we assume that every edge connecting the node N and its child node CN_i is labeled with a single symbol.

To find a new candidate, the algorithm computes the lower-bound time warping distance $D_{\mathrm{mtw-lb}}$ between $\mathit{label}(\mathit{root}, CN_i)$ and $\boldsymbol{q}$. Here, $\mathit{label}(\mathit{root}, CN_i)$ denotes a series of labels on the path from the root to CN_i. The definition of $D_{\mathrm{mtw-lb}}$ is given in the following subsection.

For this distance computation, the algorithm uses the dynamic programming technique and thus builds a cumulative distance table with $\boldsymbol{q}$ on the X-axis and $\mathit{label}(\mathit{root}, CN_i)$ on the Y-axis. If N is the root (i.e. CN_i is the direct child of the root), then the distance table is built from the bottom. Otherwise, the distance table is constructed by augmenting a new row on the existing table T which has been accumulated from the root to N. The algorithm calls the function AddRow $(T, \boldsymbol{q}, \mathit{label}(N, CN_i), D_{\mathrm{mtw-lb}})$ to build a new cumulative distance table, using the distance function $D_{\mathrm{mtw-lb}}$, by augmenting a new row for $\mathit{label}(N, CN_i)$ on T. If the rightmost column of the newly added row has the value less than or equal to the distance tolerance ε, then $\mathit{label}(\mathit{root}, CN_i)$ is added into a candidate set.

To determine whether visiting the subtree of CN_i is needed, the algorithm reads each column of the newly added row. If at least one column has a value less than or equal to ε, then the algorithm continues down the tree to find more candidates. Otherwise, the algorithm moves to the next child of N. This branch-pruning method is based on the following theorem.

Theorem 1. *If all columns of the top row of the cumulative distance table have values greater than a distance tolerance ε, adding more rows on this table does not yield the new values less than or equal to ε.*

Proof. The proof is given in Ref. 25. □

4.2. *Lower-bound time warping distance function*

Since every edge of the suffix tree is labeled with symbols, the exact multi-dimensional time warping distance between a query sequence and a subsequence contained in the suffix tree cannot be obtained. Therefore we introduce the new distance function $D_{\mathrm{mtw-lb}}$ that returns a lower-bound distance of D_{mtw}.

Definition 3. Given two subsequences $\boldsymbol{x}$ and $\boldsymbol{y}$ of multi-dimensional elements, the distance function $D_{\mathrm{mtw-lb}}(\boldsymbol{x}^{\mathrm{C}}, \boldsymbol{y})$ that returns a lower-bound distance of $D_{\mathrm{mtw}}(\boldsymbol{x}, \boldsymbol{y})$ is defined as follows:

$$D_{\mathrm{mtw-lb}}(\langle\,\rangle, \langle\,\rangle) = 0$$

$$D_{\mathrm{mtw-lb}}(\boldsymbol{x}^{\mathrm{C}}, \langle\,\rangle) = D_{\mathrm{mtw-lb}}(\langle\,\rangle, \boldsymbol{y}) = \infty$$

$$D_{\mathrm{mtw-lb}}(\boldsymbol{x}^{\mathrm{C}}, \boldsymbol{y}) = D_{\mathrm{mbase-lb}}(\boldsymbol{x}^{\mathrm{C}}[1], \boldsymbol{y}[1]) + \min \begin{cases} D_{\mathrm{mtw}}(\boldsymbol{x}^{\mathrm{C}}, \boldsymbol{y}[2:-]) \\ D_{\mathrm{mtw}}(\boldsymbol{x}^{\mathrm{C}}[2:-], \boldsymbol{y}) \\ D_{\mathrm{mtw}}(\boldsymbol{x}^{\mathrm{C}}[2:-], \boldsymbol{y}[2:-]) \end{cases}$$

$$D_{\mathrm{mbase-lb}}(C, \boldsymbol{y}[1]) = \sum_{h=1}^{k} W_h * D_{\mathrm{base-lb}}(C[h], \boldsymbol{y}[1][h])$$

$$D_{\mathrm{base-lb}}(C[h], \boldsymbol{y}[1][h]) = \begin{cases} 0 & \text{if } C[h]\cdot\min \le \boldsymbol{y}[1][h] \le C[h]\cdot\max \\ C[h]\cdot\min - \boldsymbol{y}[1][h] & \text{if } \boldsymbol{y}[1][h] < C[h]\cdot\min \\ \boldsymbol{y}[1][h] - C[h]\cdot\max & \text{if } \boldsymbol{y}[1][h] > C[h]\cdot\max \end{cases}$$

Here $\boldsymbol{x}^{\mathrm{C}}$ is the symbol sequence obtained from $\boldsymbol{x}$. In this definition, C is the first symbol (i.e. $\boldsymbol{X}^{\mathrm{C}}[1]$) of $\boldsymbol{X}^{\mathrm{C}}$ and $C[h]$ denotes the range (i.e. minimum and maximum values) within which the multi-dimensional elements represented by the symbol C take the values for their hth feature dimension.

It is apparent that $D_{\mathrm{mbase-lb}}$ always produces the distance not larger than the distance returned by D_{mbase} for any two multi-dimensional elements. Therefore it is also obvious that $D_{\mathrm{mtw-lb}}$ always produces the distance not larger than the distance returned by D_{mtw} for any two sequences of multi-dimensional elements.

The proposed query processing algorithm uses $D_{\mathrm{mbase-lb}}$ as a filtering function during the index traversal. When the multi-dimensional time warping distance D_{mtw} between a multi-dimensional data subsequence and a query sequence is not larger than ε, their lower-bound time warping distance $D_{\mathrm{mtw-lb}}$ is certainly smaller than or equal to ε. This implies that all the data subsequences within ε from a query sequence are surely included in the candidate set. On the other hand, the algorithm safely filters out the data subsequences when their lower-bound distances to a query sequence are beyond ε.

4.3. *Algorithm analysis*

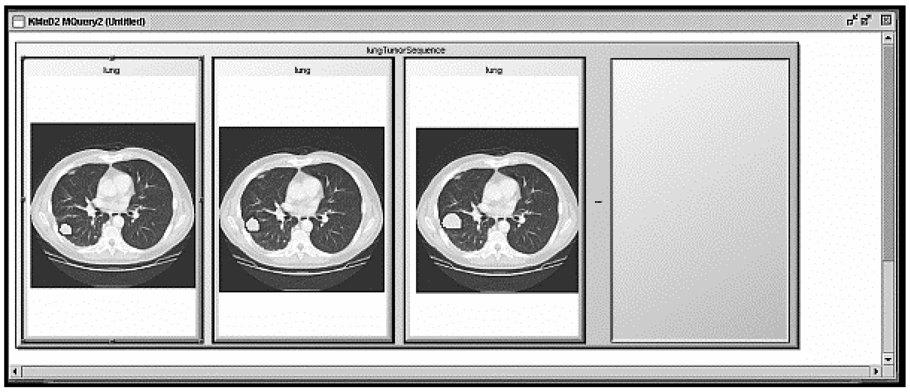
Before analyzing the complexity of the proposed query processing algorithm, let us examine the complexity of the naïve method. The naïve method reads each image sequence and builds as many cumulative distance tables as the number of suffixes contained in the sequence. Let k be the number of features extracted from each image sequence. Then the complexity of building a cumulative distance table for the query image sequence $\boldsymbol{q}$ and the suffix of length L is $O(kL|\boldsymbol{q}|)$. For m data sequences whose average length is $\bar{L}$, there are $m\bar{L}$ suffixes and their average length is $\frac{\bar{L}+1}{2}$. Therefore the complexity of the naïve method is expressed as $O(km\bar{L}^2|\boldsymbol{q}|)$.

The proposed algorithm is computationally less expensive than the naïve method because (1) the proposed branch-pruning method reduces the search space and (2) the suffixes with common prefixes share cumulative distance tables during index traversal. Thus the proposed algorithm has the complexity $O((km\bar{L}^2|\boldsymbol{q}|/R_dR_p) + kn\bar{L}\boldsymbol{q})$, where $R_d(\geq 1)$ is the reduction factor saved by the sharing of the cumulative distance tables, $R_p(\geq 1)$ is the reduction factor gained from the branch-pruning, and n is the number of subsequences requiring the post-processing. Hence, the left expression represents the cost for index traversal and the right expression the cost for post-processing.

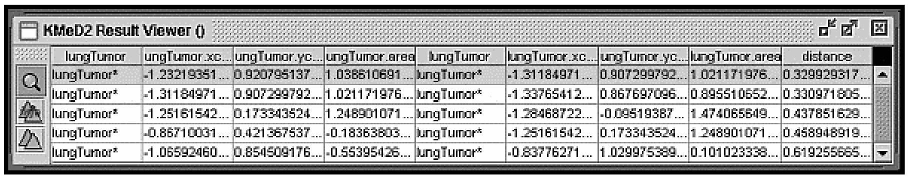
R_d grows as the lengths and the number of common edges of the suffix tree increase. Given h suffixes, $s_1, \ldots, s_h$, whose first t elements are the same, the construction of h cumulative distance tables requires the computation of $|s_1|\,|\boldsymbol{q}| + \cdots + |s_h|\,|\boldsymbol{q}|$ cells of cumulative distance tables. However it is reduced to $t|\boldsymbol{q}| + (|s_1| - t)|\boldsymbol{q}| + \cdots + (|s_h| - t)|\boldsymbol{q}|$ if the cumulative distance table for $\boldsymbol{q}$ and the common prefix of length t is shared by h suffixes. In this case, R_d can be expressed as the following:

$$R_d = \frac{|s_1| + \cdots + |s_h|}{(|s_1| + \cdots + |s_h|) - (h-1)t}\,.$$

While R_d is determined by the distribution of element values contained in sequences, R_p is decided by the number of answers required by a user. That is, R_p increases as the distance tolerance ε and thus the number of answers decrease. If



(a) query image sequence



lungTumor	ungTumor.xc...	ungTumor.yc...	ungTumor.area	lungTumor	lungTumor.xc...	ungTumor.yc...	lungTumor.area	distance
lungTumor*	-1.23219351 ...	0.920795137...	1.038610691 ...	lungTumor*	-1.31184971 ...	0.907299792...	1.021171976...	0.329929317...
lungTumor*	-1.31184971 ...	0.907299792...	1.021171976...	lungTumor*	-1.33765412...	0.867697096...	0.895510652...	0.330971805...
lungTumor*	-1.25161542...	0.173343524...	1.248901071 ...	lungTumor*	-1.28468722...	-0.09519387 ...	1.474065649...	0.437851629...
lungTumor*	-0.86710031 ...	0.421367537...	-0.18363803...	lungTumor*	-1.25161542...	0.173343524...	1.248901071 ...	0.458948919...
lungTumor*	-1.06592460...	0.854509176...	-0.55395426...	lungTumor*	-0.83776271 ...	1.029975389...	0.101023338...	0.619255665...

(b) result viewer

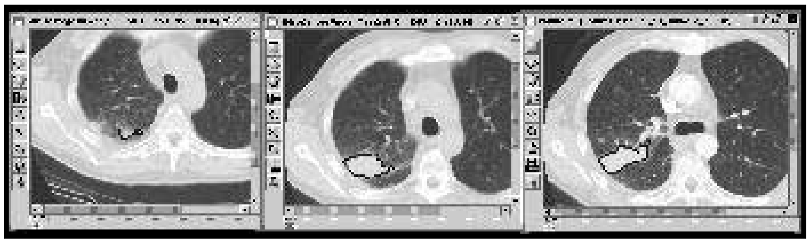
Fig. 3. (a) Lung tumor image sequence submitted as a query, and (b) the viewer of the query results.

ε is so small that just one or two subsequences can be answers, only the topmost part of the index may be visited during the query processing. In another extreme case where ε is large enough for all subsequences to be the answers, all nodes of the index need to be visited, thus making $R_p = 1$.

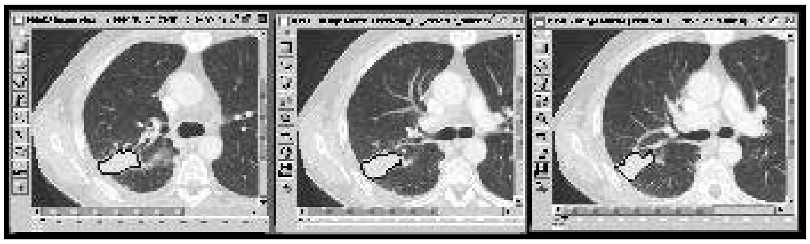
5. Application to KMeD System

This section applies the proposed indexing scheme to KMeD,[26] a knowledge-based multimedia medical distributed database system being developed in the University California at Los Angeles (UCLA). KMeD has the following features: (1) queries medical images by both image content and alphanumeric content, (2) models temporal, spatial, and evolutionary nature of medical objects, and (3) formulates queries using conceptual and imprecise terms and support cooperative processing.

In the KMeD environment, the proposed technique can be applied to the retrieval of medical image subsequences having spatio-temporal characteristics similar to those of the query sequence. Figure 3(a) shows the lung tumor image sequence submitted as a query. Lung tumor objects are identified by segmentation and their representative features such as location, size, circularity, and distance from other organs are computed using various feature extraction functions. Figure 3(b) is the viewer of the query results. The actual image subsequences retrieved by KMeD system are shown in Fig. 4.



(a) first answer



(b) second answer

Fig. 4. The image subsequences returned by the KMeD System. Both subsequences have the feature values (i.e. location and size) similar to those of the query image sequence shown in Fig. 3(a).

6. Evaluation

This section evaluates the proposed indexing scheme, implemented in C++ programming language, in terms of performance and scalability with real and synthetic data sets.

6.1. *Data set*

The real data set contains 20 patients and each patient has three lung images taken at different times. To transform this set of image sequences into a large set, we have divided each lung image into 96 sub-regions, thus making 96 sequences of length 3 for each patient. As a result, 20 patient image sequences were transformed into 1920 sequences with three images for each sequence. We then extracted the following 7 features from each image: (1) percentage of voxels, (2) mean of gray level, (3) standard deviation of gray level, (4) median of gray level, (5) tenth centile, (6) first measure of correlation, and (7) horizontal edge.

We have also generated a large set of synthetic image sequences for scalability testing. The expression for generating the values of each feature dimension was defined as a *random walk*. That is, the values of the jth feature dimension are generated by the expression $\boldsymbol{X}[i][j] = \boldsymbol{X}[i-1][j] + Z_i$, where $Z_i (i = 1, 2, \ldots)$ are independent, identically distributed random variables taken in the range of $[1, 100]$. The number and the average length of synthetic image sequences were determined according to the purpose of each scalability test.

6.2. *System configuration*

The hardware platform for the experiments was the LG-IBM Personal Computer MultiNet X-Pentium IV 2.0 GHz system equipped with 512 KB cache, 512 MB RAM, and 80 GB Seagate hard disk with 7200 RPM and 9 ms average seek time. The software platform was the Windows XP professional version.

6.3. *Performance test*

Using the real data set, we evaluated the performance of the proposed method and the naïve method. The naïve method reads each image subsequence sequentially from the database and computes its distance to the query sequence using the dynamic programming technique.

The first experiment measured the average query processing time of 100 queries with the increasing number of categories used for the discretization. The distance threshold was set to retrieve 5 answers. As shown in Fig. 5, the performance of the naïve method maintains consistency because it is independent of the discretization. However, as a whole, the performance of the proposed method becomes better as the number of categories increases. This is because the proposed lower-bound distance function becomes closer to the original distance metric and thus reduces the number

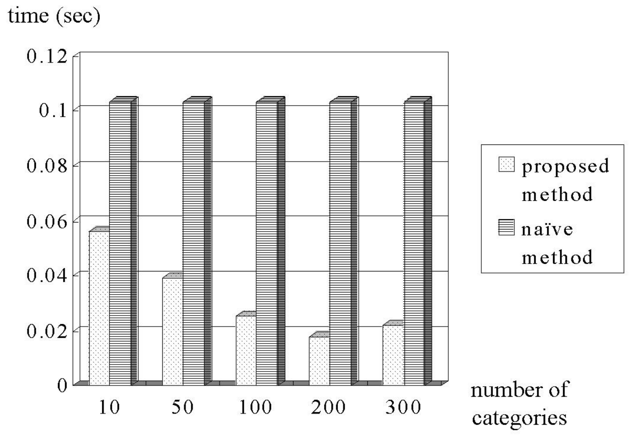


Fig. 5. The average query processing time for a selected number of categories used for the discretization. This experiment used 1920 image data sequences, each of which has three elements with seven features. The average length of query sequences was three. The distance threshold was set to retrieve five answers.

of false alarms. Notice that the performance degrades when the number of categories exceeds a certain threshold. This is because a large number of categories results in a large index tree. This threshold value may be used as the optimal number of categories.

The second experiment compared the two approaches with the increasing number of answers. The number of categories in our approach was set to 200. As shown in Fig. 6, the performance of both approaches degrades slightly as the number of answers increases. However a large number of answers decreases the performance

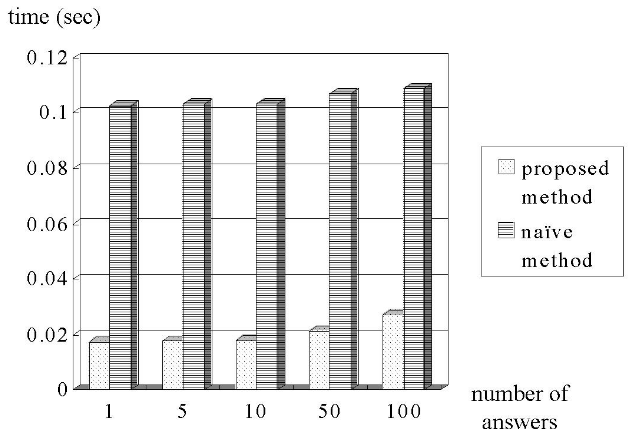


Fig. 6. The average query processing time for a selected number of answers requested by a user. This experiment used 1920 image data sequences, each of which has three elements with seven features. The average length of query sequences and the number of categories were 3 and 200, respectively.

benefit of our approach due to an enlarged search space. However this is not a concern because users are interested in a small number of the high ranking similar answers.

6.4. *Scalability test*

We tested the scalability of the proposed method using the synthetic data set. The parameters used for this scalability test are (1) k (the number of feature dimensions), (2) m (the number of data image sequences), (3) $\bar{L}$ (the average number of images in a data sequence), and (4) $|\boldsymbol{q}|$ (the average number of images in a query sequence). For every scalability test, the number of categories was 100 and the distance tolerance was adjusted to retrieve answers 10^{-3}% of the total number of data subsequences.

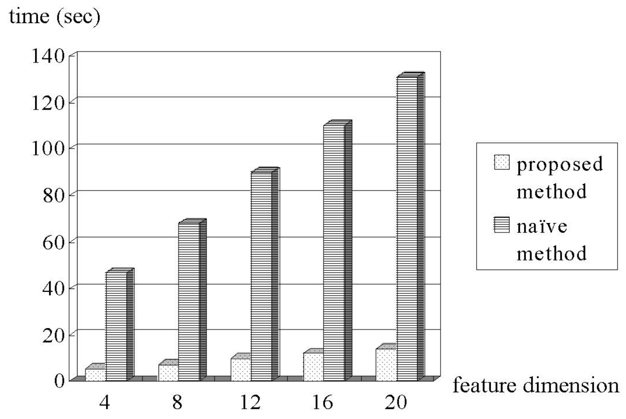


Fig. 7. The average query processing time for a selected number of feature dimensions. The other parameters were set at $m = 500$, $\bar{L} = 200$, and $|\boldsymbol{q}| = 20$.

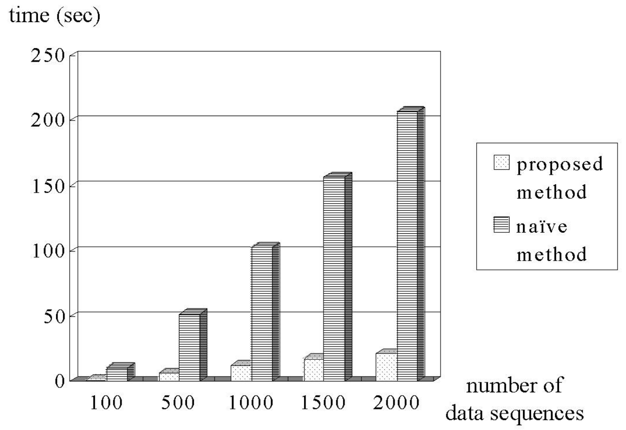


Fig. 8. The average query processing time for a selected number of data sequences. The other parameters were set at $k = 5$, $\bar{L} = 200$, and $|\boldsymbol{q}| = 20$.

January 10, 2003 16:8 WSPC/164-IJIG 00090

The first scalability test compared the average query processing time of both approaches with the number of feature dimensions increasing from 4 to 20. The other parameters were set at $m = 500$, $\bar{L} = 200$, and $|\boldsymbol{q}| = 20$. As shown in Fig. 7, the query processing time for both approaches increases linearly as the number of feature dimensions increases. Notice that our approach has a much lower rate of increase with the number of features.

The second scalability test compared the performance of both approaches with the number of data image sequences increasing from 100 to 2000. The other parameters were set at $k = 5$, $\bar{L} = 200$, and $|\boldsymbol{q}| = 20$. As shown in Fig. 8, the query processing time for both approaches increases linearly with the number of data sequences but the ratio of performance improvement remains approximately constant.

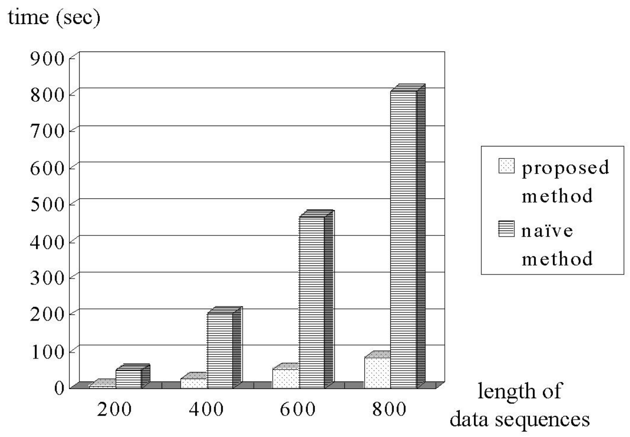


Fig. 9. The average query processing time for the selected average length of data sequences. The other parameters were set at $k = 5$, $\bar{L} = 200$, and $|\boldsymbol{q}| = 20$.

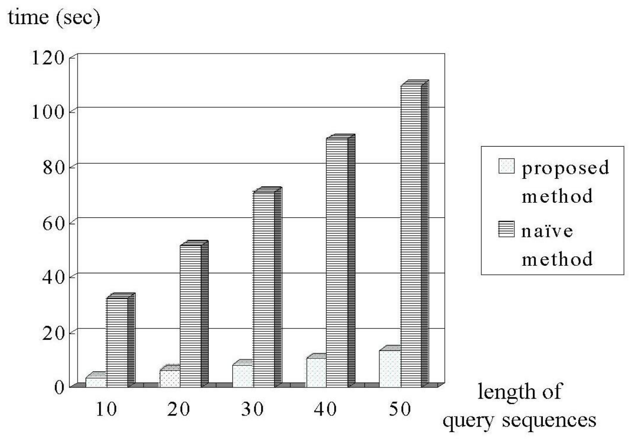


Fig. 10. The average query processing time for selected average length of query sequences. The other parameters were set at $k = 5$, $m = 500$, and $\bar{L} = 200$.

The third scalability test compared the performance for the length of data image sequences increasing from 200 to 800. The other parameters were set at $k = 5$, $m = 500$, and $|\boldsymbol{q}| = 20$. As shown in Fig. 9, the query processing time for both approaches increases quadratically with the length of data sequences but the ratio of performance improvement remains approximately constant.

The final scalability test compared the performance of both approaches for the length of query image sequences increasing from 10 to 50. The other parameters were set at $k = 5$, $m = 500$, and $\bar{L} = 200$. As shown in Fig. 10, the query processing time for both approaches increase linearly with the length of query sequences but the ratio of performance improvement remains approximately constant.

The experimental results validate the algorithm analysis presented in Sec. 4.3, stating that our approach is more scalable than the naïve method.

7. Related Work

This section briefly summarizes the previous work on similarity search in single- and multi-dimensional sequence databases.

7.1. *Work on single dimensional sequences*

There has been much research on similarity search in the database of single dimensional sequences. Agrawal *et al.*[1] proposed the *F-Index*, a similarity searching technique for whole sequence matching. Sequences are converted into the frequency domain by the Discrete Fourier Transform (DFT) and are subsequently mapped onto multi-dimensional points that are managed by an R*-tree; this technique was extended to locate similar subsequences.[3] Since both approaches use the Euclidean distance as a similarity measure, sequences of different lengths or different sampling rates cannot be matched.

A handful of sequence matching techniques that allow transformations were proposed. Goldin *et al.*[8] grouped sequences into equivalent classes using the normal form. Although the normal form is invariant to shape-based transformations such as scaling and shifting, it does not handle the compressions or the stretches of element values along the time axis. Rafiei *et al.*[5] proposed a class of sequence transformations that can be used in a query language to express similarity with an R-tree index. The proposed transformations handle moving average and global time scaling, but not time warping.

More recent approaches permit the matching of sequences of different lengths. Bozcaya *et al.*[27] presented a modified version of an edit distance, judging that two sequences are similar if a majority of their elements match. Yi *et al.*[13] supported the time warping distance by using a two-step filtering process: a FastMap index filter followed by a lower-bound distance filter. The underlying index structures of both approaches[13,27] are based on the triangular inequality. The approach suggested by Keogh *et al.*[28] read a data sequence sequentially from the database, converted it into an ordered list of piece-wise linear segments using the best fitting line, and then applied the modified time warping distance measure.

Park *et al.*[10−12,29] proposed a series of indexing methods supporting the time warping distance. A segment-based subsequence searching scheme[29] was proposed for the database of long sequences. This scheme changed the similarity measure from the time warping distance to the piece-wise time warping distance and limited the number of data subsequences to be compared with a query sequence. A new distance function that consistently underestimates the time warping distance and also satisfies the triangular inequality was proposed in Ref. 10. With L_∞ as a base distance metric for elements, this distance function was employed for whole matching in Ref. 10 and for subsequence matching in Ref. 12. The indexing technique proposed in Ref. 11 used a disk-based suffix tree as an index structure and applied a couple of lower-bound distance functions in index space. To make the index structure compact and thus accelerate the query processing, it converted sequences in continuous numeric domain into sequences in discrete symbol domain and then stored a subset of suffixes whose first values are different from their preceding values.

A couple of shape-based similarity matching schemes were proposed. Agrawal *et al.*[30] demonstrated a shape definition language (SDL) and provided an index structure for speeding up the execution of SDL queries. Shatkay *et al.*[31] introduced the notion of generalized approximate queries that specify the general shapes of data histories. Whereas both approaches may handle the variations of element values on the time axis, they are not suitable for applications that utilizes specific element values.

There are also several approaches for the matching of biological sequences. Bieganski *et al.*[24] proposed to use a disk-based suffix tree for solving the sequence alignment problem, and Wang *et al.*[32] addressed the problem of discovering patterns in protein databases with the similarity measure of a string edit distance.

7.2. *Work on multi-dimensional sequences*

Yazdani *et al.*[33] proposed the access method for the matching of multi-dimensional sequences with a modified version of an edit distance. This method, however, focuses on whole matching and uses an index structure based on the triangular inequality, therefore leading to possible false dismissal when using the time warping distance as a similarity measure.

There has also been extensive work on the similarity search on video databases. A video is a series of frames and is considered as an instance of a multi-dimensional sequence.

A large number of previous video indexing approaches depend on the naïve method to retrieve similar videos in a database. Mohan[34] and Vailaya *et al.*[35] proposed the video sequence matching methods using action similarity and the combination of image content and image motion, respectively. Lienhart *et al.*[36] and Sanchez *et al.*[37] used color coherence vectors and principal components of color histograms, respectively, for detecting similar commercial video clips. Adjeroh *et al.*[38] employed the *vstring* representation for video sequences and introduced *vstring edit distance* as a similarity indicator. Ardizzone *et al.*[39] extracted a single MPEG motion vector from each 16×16 sub-image, and Meng and Chang[40] used low-level motion features such as zoom and pan of camera. Since all the video indexing approaches mentioned above use the naïve method, their performance deteriorates when the database is large.

Squire *et al.*[41] proposed the use of inverted file techniques for feature-based image retrieval, and Hampapur *et al.*[42] applied the inverted file to media tracking. Since both approaches keep the inverted files in the main-memory during the query processing, they are not suitable for a large volume of video databases.

8. Conclusion

An image sequence database is a set of image sequences, each of which is an ordered list of images. Similarity search is an operation that finds sequences or subsequences whose changing patterns are similar to that of a given query sequence. Similarity search is of growing importance in the areas of data mining and digital image/video libraries. Especially, in the medical domain, a search for patients with similar temporal characteristics can augment the process of patient care by providing physicians with insight into the treatment of previous patients with similar medical conditions.

This paper proposed to use a disk-based suffix tree as an indexing method for fast retrieval of similar image subsequences without false dismissals. Since image sequences are apt to be of different lengths and/or different sampling rates, the proposed method employed the multi-dimensional time warping distance as a similarity measure. This similarity metric allows two sequences to be stretched along the time axis in order to minimize their difference.

Experiments on medical and synthetic image sequences verified that the proposed method significantly outperforms the naïve method and scales well in a large volume of image sequence databases. The contributions of our work are (1) proposing a flexible similarity measure suitable for image sequences with different interval lengths between elements, (2) defining a lower-bound distance metric which always underestimates the distance between a query image sequence and a discretized data subsequence and thus assures the absence of false dismissal, (3) presenting an efficient query processing algorithm which integrates the proposed distance functions and the branch-pruning method in a seamless fashion.

References

1. R. Agrawal, C. Faloutsos and A. Swami, "Efficient similarity search in sequence databases," *Proc. Int. Conf. Foundations of Data Organization and Algorithms* (*FODO*) (1993), pp. 69–84.
2. R. Agrawal, K. Lin, H. S. Sawhney and K. Shim, "Fast similarity search in the presence of noise, scaling, and translation in time-series databases," *Proc. Int. Conf. Very Large Data Bases* (*VLDB*) (1995), pp. 490–501.

3. C. Faloutsos, M. Ranganathan and Y. Manolopoulos, "Fast subsequence matching in time-series databases," *Proc. ACM Int. Conf. Management of Data* (*SIGMOD*) (1994), pp. 419–429.
4. M. S. Chen, J. Han and P. S. Yu, "Data mining: An overview from database perspective," *IEEE Trans. Knowledge and Data Eng.* (*TKDE*) **8**(6), 866–883 (1996).
5. D. Rafiei and A. Mendelzon, "Similarity-based queries for time series data," *Proc. ACM Int. Conf. Management of Data* (*SIGMOD*) (1997), pp. 13–24.
6. K. W. Chu and M. H. Wong, "Fast time-series searching with scaling and shifting," *Proc. ACM Symp. Principles of Database Syst.* (*PODS*) (1999), pp. 237–248.
7. G. Das, D. Gunopulos and H. Mannila, "Finding similar time series," *Proc. Principles Practice Knowledge Discovery in Databases* (*PKDD*) (1997), pp. 88–100.
8. D. Q. Goldin and P. C. Kanellakis, "On similarity queries for time-series data: Constraint specification and implementation," *Proc. Constraint Programming* (1995), pp. 137–153.
9. D. J. Berndt and J. Clifford, "Finding patterns in time series: A dynamic programming approach," *Advances Knowledge Discovery Data Mining* (AAAI/MIT, 1996), pp. 229–248.
10. S. W. Kim, S. Park and W. W. Chu, "An index-based approach for similarity search supporting time warping in large sequence databases," *Proc. IEEE Int. Conf. Data Eng.* (*ICDE*) (2001) pp. 607–614.
11. S. Park, W. W. Chu, J. Yoon and C. Hsu, "Efficient searches for similar subsequences of different lengths in sequence databases," *Proc. IEEE Int. Conf. Data Eng.* (*ICDE*) (2000) pp. 23–32.
12. S. Park, S. W. Kim, J. S. Cho and S. Padmanabhan, "Prefix-querying: An approach for effective subsequence matching under time warping in sequence databases," *Proc. ACM Int. Conf. Information and Knowledge Management* (*CIKM*) (2001) pp. 255–262.
13. B.-K. Yi, H. V. Jagadish and C. Faloutsos, "Efficient retrieval of similar time sequences under time warping," *Proc. IEEE Int. Conf. Data Engineering* (*ICDE*) (1998), pp. 201–208.
14. L. Rabinar and B.-H. Juang, *Fundamentals of Speech Recognition* (Prentice Hall, 1993).
15. G. A. Stephen, *String Searching Algorithms* (World Scientific Publishing, 1994).
16. K. Shim, R. Srikant and R. Agrawal, "High-dimensional similarity joins," *Proc. IEEE Int. Conf. Data Eng.* (*ICDE*) (1997), pp. 301–311.
17. K. Fukunaga and P. M. Narendra, "A branch and bound algorithms for computing k-nearest neighbors," *IEEE Trans. Computers* **C-24**(7), 750–753 (1975).
18. T. Brinkhoff, H.-P. Kriegel, R. Schneider and B. Seeger, "Multi-step processing of spatial joins," *Proc. ACM Int. Conf. Management of Data* (*SIGMOD*) (1994), pp. 237–246.
19. M. S. Brown, J. G. Goldin, M. F. McNitt-Gray, L. E. Greaser, A. Sapra, K. T. Li, J. W. Sayre, M. Martin and D. R. Aberle, "Knowledge-based segmentation of thoracic CT images for assessment of split lung function," *Proc. Med. Phys.* (2000).
20. M. S. Brown, M. F. McNitt-Gray, J. G. Goldin, L. E. Greaser, U. M. Hayward, J. W. Sayre, M. K. Arid and D. R. Aberle, "Automated measurement of single and total lung volume from CT," *Computer Assisted Tomography* **23**(4), 632–640 (1999).
21. M. S. Brown, L. S. Wilson, B. D. Doust, R. W. Gill and C. Sun, "Knowledge-based method for segmentation and analysis of lung boundaries in chest X-ray images," *Computerized Med. Imaging Graphics* **22**(6), 463–477 (1999).
22. M. Sonka, V. Hlavac and R. Boyle, *Image Processing, Analysis, and Machine Vision* (Chapman Hall, 1993).

23. W. W. Chu and K. Chiang, "Abstraction of high level concepts from numerical values in databases," *Proc. AAAI Workshop Knowledge Discovery Databases* (1994), pp. 133–144.
24. P. Bieganski, J. Riedl and J. V. Carlis, "Generalized suffix trees for biological sequence data: Applications and implementation," *Proc. Hawaii Int. Conf. Syst. Sci.* (1994).
25. S. Park, W. W. Chu, J. Yoon and C. Hsu, "A suffix tree for fast similarity searches of time-warped subsequences in sequence databases," Tech. Rep. (UCLA-CS-TR-990005, UCLA, 1999).
26. W. W. Chu, A. F. Cardenas and R. K. Taira, "KMeD: a knowledge-based multimedia medical distributed database system," *Information Syst.* **20**(2), 75–96 (1995).
27. T. Bozkaya, N. Yazdani and M. Ozsoyoglu, "Matching and indexing sequences of different lengths," *Proc. ACM Int. Conf. Information Knowledge Management* (*CIKM*) (1997), pp. 128–135.
28. E. J. Keogh and M. J. Pazzani, "Scaling up dynamic time warping to massive datasets," *Proc. Principles Practice Knowledge Discovery Databases* (*PKDD*) (1999).
29. S. Park, D. Lee and W. W. Chu, "Fast retrieval of similar subsequences in long sequence databases," *Proc. IEEE Knowledge and Data Eng. Exchange Workshop* (*KDEX*) (1999) pp. 60–67.
30. R. Agrawal, G. Psaila, E. L. Wimmers and M. Zait, "Querying shapes of histories," *Proc. Int. Conf. Very Large Data Bases* (*VLDB*) (1995), pp. 502–514.
31. H. Shatkay and S. B. Zdonik, "Approximate queries and representations for large data sequences," *Proc. IEEE Int. Conf. Data Eng.* (*ICDE*) (1994), pp. 536–545.
32. J. T. Wang, G. Chirn, T. G. Marr, B. Shapiro, D. Shasha and K. Zhang, "Combinatorial pattern discovery for scientific data: Some preliminary results," *Proc. ACM Int. Conf. Management of Data* (*SIGMOD*) (1994), pp. 115–125.
33. N. Yazdani and M. Ozsoyoglu, "Sequence matching of images," *Proc. Int. Conf. Statistical and Scientific Database Management* (*SSDBM*) (1996), pp. 53–62.
34. R. Mohan, "Video sequence matching," *Proc. Int. Conf. Acoustics Speech and Signal Processing* (*ICASSP*) (1998).
35. A. Vailaya, W. Xiong and A. K. Jain, "Query by video clip," *Proc. Int. Conf. Pattern Recognition* (1998).
36. R. Lienhart, C. Kuhmunch and W. Effelsberg, "On the detection and recognition of television commercials," *Proc. IEEE Int. Conf. Multimedia Computing and Syst.* (1997).
37. J. M. Sanchez, X. Binefa, J. Vitria and P. Radeva, "Local color analysis for scene break detection applied to TV commercials recognition," *Proc. Visual 99* (1999).
38. D. A. Adjeroh, M. C. Lee and I. King, "A distance measure for video sequence similarity matching," *Proc. Int. Workshop Multi-Media Database Management Syst.* (1998).
39. E. Ardizzone, M. L. Cascia, A. Avanzato and A. Bruna, "Video indexing using MPEG motion compensation vectors," *Proc. IEEE Int. Conf. Multimedia Computing System* (1999), pp. 490–501.
40. J. Meng and S.-F. Chang, "CVEPS — A compressed video editing and parsing system," *Proc. ACM Multimedia* (1996).
41. D. M. Squire, H. Muller and W. Muller, "Improving response time by search pruning in content based image retrieval system, using inverted file techniques," *Proc. IEEE Workshop on Content Based Image and Video Libraries* (1990).
42. A. Hampapur and R. Bolle, "Feature based indexing for media tracking," *Proc. IEEE Int. Conf. Multimedia Expo* (*ICME*) (2000).

Sanghyun Park is an assistant professor in the Department of Computer Science and Engineering, Pohang University of Science and Technology (POSTECH) in Korea. He received his BS and MS degrees in Computer Engineering from Seoul National University, Korea, in 1989 and 1991, respectively. He received his PhD degree in Computer Science from the University of California at Los Angeles (UCLA) in 2001. His research interest includes database, data mining and multi-media systems.

He has published several research papers in the areas of time series and web databases. He is currently working on XML indexing, dynamic web data caching and gene sequence searching.

Wesley W. Chu is a professor of Computer Science and was the past chairman (1988–1991) of the Computer Science Department at the University of California, Los Angeles. He received his BSE (EE) and MSE (EE) from the University of Michigan in 1960 and 1961, respectively. He received his PhD (EE) from Stanford University in 1966.

From 1964 to 1966, he worked on the design of large-scale computers at IBM in Menlo Park and San Jose, California. From 1966 to 1969, he researched computer communications and distributed databases at Bell Laboratories, Holmdel, New Jersey. He joined the University of California, Los Angeles in 1969. He directs a research group at UCLA in the areas of distributed processing, knowledge-based multimedia medical information systems, and intelligent information systems.

He was the conference chair of the 16th International Conference on Conceptual Modeling (ER'97). He is also currently a member of the Editorial Board of the Journal on Applied Intelligence and an Associate Editor for the Journal of Data and Knowledge Engineering. Dr. Chu is a Fellow of IEEE.