Approximate Video Search Based on Spatio-Temporal Information of Video Objects

CHIA-HAN LIN[1, *] AND ARBEE L. P. CHEN[2]
[1]*Department of Information Communication, Asia University, Taichung, Taiwan*
[2]*Department of Computer Science, National Chengchi University, Taipei, Taiwan*

ABSTRACT

The spatio-temporal information of a video object is important for content-based video retrieval. In this paper the spatio-temporal information of a video object, such as the velocity, location and orientation of a video object, is represented by an ST-string. Since the user may not be interested in all spatio-temporal information, the user query can be represented by a QST-string which contains some spatio-temporal information only, such as the velocity and orientation of a video object. Therefore, the video retrieval problem can be transformed into the problem of approximately matching a QST-string with ST-strings. A general solution is proposed in this paper, which includes a similarity measure, an index structure and the corresponding matching methodology. The experiment results show that both the approximate and exact results can be found efficiently via the proposed index structure and the matching algorithm.

Key words: spatio-temporal information, approximate match, video retrieval, video model.

1. INTRODUCTION

Due to the great improvement in computer technologies and the mature development of the World-Wide-Web, more and more multimedia data are distributed over the Internet. Consequently, how to retrieve the desired multimedia data efficiently from the Internet has become an important issue. Among various media, video contains the richest content, including images, audio and a sequence of images. Therefore the image information, the audio information and the temporal information of image sequences should be regarded as the video content. In a video database management system, modeling, indexing and query processing should be considered for content-based video retrieval.

Since a video document can be considered as a sequence of images, the image features can be used to model and retrieve the video content. However, some other features such as trajectory, spatial and temporal relationships are only introduced in video documents. Many objects such as people, cars and animals move in a video document from time to time. In other words, they change their locations on the screen frequently when the video is played. This characteristic forms the concept of spatial-temporal relationship of the video.

Since the location of an object may change as time passes, there are relationships between the spatial property and the temporal property. Ren and Singh (2004) proposed a spatial and temporal model for the video. The

[*] Corresponding author. E-mail: edgarlin@asia.edu.tw

relationships between video objects are recorded in a vector and the multi-dimensional indexing techniques can be applied to index these relationships for video retrieval. Liu and Chen (2002) proposed a data structure named 3D-list, which is used to represent the spatial and the temporal information of the video objects and the index is constructed based on the structure for query processing. Jiang and Elmagarmid (1998) proposed a data model for the video objects. The characteristics and the trajectory of the video objects are recorded in the model. Based on some spatio-temporal relationships, such as appear-together and overlap, the query can be specified.

In previous work (Lin & Chen, 2001a; 2001b), we proposed a video data model to represent the content of video data. The values of the attributes of video objects, such as color, object type, location and size, are recorded to represent the static property of video data. Moreover, the high-level motion property of single video objects and multiple video objects is automatically derived.

In general, the spatio-temporal relationships between video objects can be considered as a string of multiple spatio-temporal feature values. The content-based video retrieval problem can therefore be solved by the traditional string matching algorithms. However, since traditional string matching algorithms only deal with strings with one attribute, new index structures and algorithms for matching strings with multiple spatio-temporal attributes should be considered for efficient query processing.

We have developed several methods (Chen et al., 2000; Liu & Chen, 2002) to deal with the string matching problem with a small number of attributes. Some other approaches consider matching all attributes simultaneously (Kahveci, Singh & Gurel, 2002; Lee, Chun, Kim, Lee & Chung, 2000). However, since the user may only be interested in some spatio-temporal features, a query may not involve all attributes. In previous work (Lin & Chen, 2006), the index structures and the matching methodologies, which can be applied to a different number of query attributes, are proposed. Since multiple attributes are considered in the data strings, the multiple index structures are constructed for multiple attributes. To process a query, the query string will first be decomposed into several components. Each component will be individually processed based on the corresponding index structure and the corresponding results combined. The combined results will be further verified to see whether they are the final results. Some pruning techniques are proposed to enhance the efficiency of the query processing.

For the video retrieval problem, approximate query processing can be even more important. However, only the exact matching problem is considered (Lin & Chen, 2006). Moreover, the index structures and the proposed pruning techniques are not suitable for the approximate matching problem. Therefore, in this paper, a similarity measure, a new index structure which considers all attributes simultaneously, and the corresponding matching methodology are designed for the approximate matching problem.

The edit distance (Levenshtein, 1995) is widely used to measure the distance between two strings by the number of edit operations, i.e., insertion, deletion and replacement, to transfer one string to another string. The weighted edit distance

(Gusfield, 1997) measures the weight of insertion and deletion operations depending on the character in the alphabet to be inserted or deleted. In this paper, the similarity measure based on the weighted edit distance is adopted.

The remainder of this paper is organized as follows. The video model with spatio-temporal information is introduced in Section 2. The index structure and matching mechanism for exactly matching the QST-string is described in Section 3. In Section 4, the similarity measurement between an ST-string and a QST-string is introduced. The matching mechanism for approximately matching the QST-string is described in Section 5. To show the efficiency of our approach, a series of experiments are performed. The experiment results are discussed in Section 6. Section 7 concludes this paper and points out future work.

2. VIDEO MODEL WITH SPATIO-TEMPORAL INFORMATION

2.1 Video Model

The video model is used to represent the content of a video. Since the whole video contains rich information, it is first segmented into several scenes. The proposed video model is modified from previous work (Lin & Chen, 2001b).

In the proposed model, a video scene is considered as the basic unit for video representation. The motions and other properties of the video objects appearing in the scene are used to represent the content of the scene. Video object is the base element used to represent the content of the video scene. The information of a video object can be expressed as a quadruple (oid, sid, Type, PA) where

oid:	object ID
sid:	scene ID of the scene containing the object
Type:	type of the object
PA:	perceptual attributes of the object

The perceptual attributes represent the visual information of the video object including the following attributes.

Color:	dominant color of the object
Size:	size of the object
Trajectory:	sequence of object locations in the frames of the scene
Motions:	motion properties of the video object

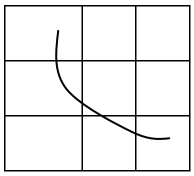
There are three types of motions: velocity, acceleration and orientation. The velocity has four possible values: High, Medium, Low and Zero. Similarly, the acceleration has three possible values: Positive, Zero and Negative, and the orientation has eight possible values: East, Northeast, North, Northwest, West, Southwest, South and Southeast. The characteristic of motions can be represented as a string of motion values. A motion value represents a sequence of frames in which the value of the motion properties remains the same.

The video frame is divided into 9 areas as shown in Figure 1. Therefore, the trajectory of a video object can be represented as a string of areas.

11	12	13
21	22	23
31	32	33

Figure 1. Values of location areas.

Example 1. Assume the alphabet of velocity values is {H, M, L, Z}, the alphabet of orientation values is {E, NE, N, NW, W, SW, S, SE}, the alphabet of the acceleration values is {P, Z, N} and the alphabet of locations is {11, 12, 13, 21, 22, 23, 31, 32, 33}. Suppose that there is a video object whose trajectory is shown as follows.



The motions and trajectory of the video object may be represented as the following strings of feature values.

Velocity:	"H M H M S"
Acceleration:	"P N P Z N Z"
Orientation:	"S SE E"
Trajectory:	"11 21 22 32 33"

2.2 Spatio-temporal String (ST-string)

Based on the video data model proposed in the previous section, the spatio-temporal information of a video object can be represented by the corresponding string of the trajectory and motions. Since the user may be interested in one or more spatio-temporal attributes, the trajectory and motions of a video object will be combined and form an ST-string for indexing and query processing. Each symbol in the ST-string represents a state of the spatio-temporal characteristics of a video object in which all the values of the spatio-temporal features remain the same. That is, if the value of some spatio-temporal feature becomes different, a new symbol will be used and recorded in the ST-string. Since only the changes of the spatio-temporal information are considered, we assume every ST-string recorded in the database is a compact ST-string. That is, no adjacent symbols of the ST-string are the same.

Example 2. The spatio-temporal information of a video object introduced in Example 1 can be represented as an ST-string with the location, velocity, acceleration and orientation feature values as follows.

sts_1	sts_2	sts_3	sts_4	sts_5	sts_6	sts_7	Sts_8
11	11	21	21	22	32	32	33
H	H	M	H	H	M	S	S
P	N	P	Z	N	N	N	Z
S	S	SE	SE	SE	SE	E	E

In Example 2, the ST-string represents a video object, which located in “11” area of the frame, moves with High speed, Positive acceleration toward the South at first, and then the acceleration becomes Negative.

The ST-strings of video objects in a video will be derived and recorded in the index for query processing. However, since the user may only be interested in some spatio-temporal features, a query may not involve all attributes. Therefore, a query will be represented as a QST-string which is formed by q spatio-temporal attributes, where $q \leq 4$. Assume the user is interested in velocity and orientation of a video object, the query will be represented as a QST-string with the corresponding 2 spatio-temporal attributes as shown in the following.

H	M	H	M	H	M
S	SE	SE	SE	S	SE

Besides the ST-string recorded in the index, the QST-string is also a compact string.

The video retrieval problem is therefore transformed into a problem of matching a QST-string with ST-strings recorded in the index structure. Since the symbols in the QST-string and ST-string consider a different number of spatio-temporal attributes, we define the concept of symbol containment for matching a QST symbol with an ST symbol. A QST symbol *qs* is contained in an ST symbol *sts* if the values of the corresponding *q* attributes are the same, that is, the spatio-temporal characteristics of the video object is the same while considering the *q* spatio-temporal attributes. For example, the QST symbol (H, E) is contained in an ST symbol (11, H, N, E) since the velocity and orientation represented by theses two symbols are the same, i.e., High Speed and East.

Based on the symbol containment, we define that an ST symbol *sts* matches a QST symbol *qs* if *qs* is contained in *sts*.

For the ST-string, since only q attributes are considered in the matching process, the corresponding q feature values of the contiguous ST symbols may be the same. For example, considering sts_1 and sts_2 in Example 2, the velocity and orientation of the symbols are the same. Therefore, due to the compact property, these symbols with the same q feature values will be compressed first while matching a QST symbol since the corresponding q feature values are not changed. That is, these symbols will match the same QST symbol. Therefore, we say that an ST-string STS = $sts_1sts_2\text{L}sts_d$ exactly matches a QST-string QST = $qs_1qs_2\text{L}qs_a$ if sts_1 matches qs_1, sts_d matches qs_q and the following criteria are satisfied.

$$\forall\, sts_i\,,\, \mathit{if}\ sts_i\ \mathit{matches}\ qs_j\,,$$
$$sts_{i+1}\ \mathit{matches} \begin{cases} qs_j & \mathit{if}\ sts_{i+1}\ \mathit{does\ not\ match}\ qs_{j+1} \\ qs_{j+1} & \end{cases}$$

However, in most cases, only part of the spatio-temporal information of the complete video scene will be specified in the user query. Therefore, we say that an ST-string STS matches a QST-string QST if there exists a substring of STS, say STS′, such that STS′ exactly matches QST.

Example 3. Assume a query is represented by a QST-string, including the velocity and orientation of a moving object, shown as follows.

qs_1	qs_2	qs_3
M	H	M
SE	SE	SE

For the ST-string introduced in Example 2 denoted as STS, the substring STS′ = $sts_3\text{L}sts_6$ of STS exactly matches QST since sts_3 matches qs_1, sts_4 matches qs_2, sts_5 matches qs_2, and sts_6 matches qs_3. Therefore, STS matches QST since there exists a substring STS′ which exactly matches QST.

In the following section, an index structure as well as the corresponding matching mechanism is proposed to solve the QST-string matching problem.

3. EXACT QST-STRING MATCHING

3.1 Index Structure

The ST-string can be indexed by a traditional index structure for string matching. A suffix tree is a representative index structure, which is proposed for string matching. However, since the ST-string is used to represent the spatio-temporal information of a video object, the length of a string may be very long. Moreover, since the matching of symbols is based on symbol containment, a

QST symbol may be contained in more than one ST symbol. Therefore, the number of paths to be traversed will be increased substantially for longer paths. Thus, due to performance considerations, the suffix tree structure for the ST-string is modified as the K-Prefix suffix tree (Lin & Chen, 2006) whose tree height is, at most, k.

3.2 Query Processing

The suffix tree index structure is proposed for the string matching problem. In this paper, the symbol containment is used as the matching criteria. Therefore, the suffix tree traversal procedure to find the matched results has to be modified such that it can be applied on the QST-string matching problem.

Figure 2 shows the steps to match a QST-string with ST-strings. Given a QST-string, it is used in the KP suffix tree traversal step to find all the paths in the KP suffix tree, which may contain the matched substrings. Since only the KP suffixes of the ST-strings are indexed, the matched resulted should be further verified to see whether the matched ST-strings exist.

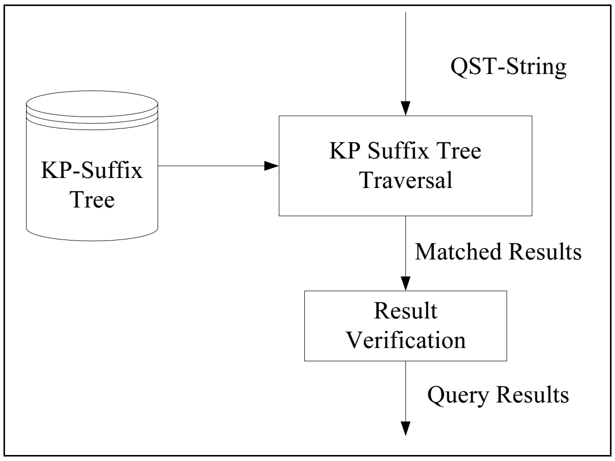


Figure 2. The steps for matching a query with ST-strings.

Figure 3 shows the steps in the KP suffix tree traversal. The traversal starts from the root node of the KP suffix tree and the QST-string, say S. The edge e_i starting from the root, which is exactly matched with some prefix of S, denoted PS_i, will be found and the traversal moves to the node pointed by the e_i. Assume the last symbol of PS_i is *ls*. Since each QST symbol may match more than one ST symbol, the symbol *ls* should be considered in the next step. Assume S_i' denotes a suffix of S where $PS_i + S_i' = S$ and S_i'' denotes another suffix of S with $S_i'' = ls + S_i'$. Both S_i' and S_i'' will also be processed to see whether the matched edge exists. The algorithm will be recursively executed and the matched results will be recorded in RS.

```
Algorithm Tree_Traversal (S, N)
Input:   The QST-string to be matched $S = s_1 s_2 \ldots s_l$
         and the starting node N.
Output:  The matched results set RS.
1.    if N is a leaf node
2.        append N.data to RS
3.    for all edges pointed from node N, denoted $e_i$
4.        let N' is the child node of N pointed by $e_i$.
5.        if $S=\phi$
6.            call Tree_Traversal($\phi$, N')
7.        else if $e_i$ is exactly matched with some prefix
      of S, say $PS_i = s_1 \cdots s_{l_i'}, where\ l_i' \leq l$
8.            let $S_i' = s_{l_i'+1} \cdots s_l$ and $S_i'' = s_{l_i'} \cdots s_l$.
9.            call Tree_Traversal(S', N')
10.           call Tree_Traversal e(S", N')
11.       else if S exactly matches some prefix of $e_i$
12.           call Tree_Traversal($\phi$, N')
13.   return
```

Figure 3. Algorithm for the KP suffix tree traversal.

4. SIMILARITY MEASURE BETWEEN AN ST-STRING AND A QST-STRING

Moreover, the approximate results are also important for video retrieval. In this paper, the dissimilarity between a query, i.e., the user specified spatio-temporal information, and a video is measured by the distance between a QST-string and an ST-string, which is defined based on the weighted edit distance. The distance between an ST symbol and a QST symbol is used to define the weight of edit operations. Since an ST symbol matches a QST symbol if the corresponding q feature values are the same, the distance between an ST symbol $sts = (s_1, s_2, \ldots, s_4)$ and a QST symbol $qs = (q_1, q_2, \ldots, q_q)$ is defined as the difference of the corresponding q feature values, which is shown as follows. The QS denotes the set of q spatio-temporal attributes considered in a QST-string.

$$dist(sts, qs) = \omega_i \sum_{i=1}^{q} d(q_i, s_{p_i}),\ where\ \omega_i\ is\ the\ weight\ of$$
$$feature\ i\ and\ d(q_i, s_{p_i})\ is\ the\ predefined\ distance\ between$$
$$q_i\ and\ s_{p_i}\ p_i \in QS$$

Note that $0 \le dist(sts, qs) \le 1$.

The distance can be considered as the cost to change feature values of *qs* such that they can be matched with *sts*. If all the feature values were changed, the cost will be the maximum value, i.e, 1. If there were no feature value to be changed, that is, *qs* is matched with *sts*, the cost will be the minimum value, i.e., 0.

Example 4. Assume the distance metric for feature 2 and 4 are shown in table 1 and table 2. Moreover, assume the weight for feature 2 and 4 are 0.6 and 0.4, respectively.

Table 1. *The distance metric for feature 2*

	H	M	L
H	0	0.5	1
M	0.5	0	0.5
L	1	0.5	0

Table 2. *The distance metric for feature 4*

	N	NE	E	SE	S	SW	W	NW
N	0	0.25	0.5	0.75	1	0.75	0.5	0.25
NE	0.25	0	0.25	0.5	0.75	1	0.75	0.5
E	0.5	0.25	0	0.25	0.5	0.75	1	0.75
SE	0.75	0.5	0.25	0	0.25	0.5	0.75	1
S	1	0.75	0.5	0.25	0	0.25	0.5	0.75
SW	0.75	1	0.75	0.5	0.25	0	0.25	0.5
W	0.5	0.75	1	0.75	0.5	0.25	0	0.25

Assume an ST-symbol *sts* = (11, M, P, NE) and a QST symbol *qs* = (H, NE), the distance between *qs* and *sts* is $dist(sts,qs) = 0.6 \times 0.5 + 0.4 \times 0 = 0.3$.

The weight of the edit operations is defined based on the distance between the edited symbol and the symbol to be matched in the ST-string. The cost of the edit operations applied on a string can therefore be measured as the total weights of the edit operations.

Therefore, the q-edit distance between an ST-string STS and a QST-string QST can be measured as the minimum cost of edit operations applied on QST such that the edited string of QST is matched with STS.

A dynamic programming method is then proposed to calculate the q-edit distance between an ST-string and a QST-string.

Given an ST-string STS = $sts_1 sts_2 \ldots sts_d$ and a QST-string QST = $qs_1 qs_2 \ldots qs_l$, let D(i, j) denote the q-edit distance between $QS_{1,i} = qs_1 qs_2 \ldots qs_i$ and $STS_{1,j} = sts_1 sts_2 \ldots sts_j$, which can be recursively defined as follows:

$$D(i, j) = \min\{D(i-1, j-1), D(i-1, j), D(i, j-1)\} + \text{dist}(sts_j, qs_i).$$

The base conditions are:

$$D(0, 0) = 0 \text{ and } D(i, 0) = i,\ D(0, j) = j,\ \forall i, j.$$

The following example shows how to calculate the q-edit distance, i.e., D(d, l), between STS and QST.

Example 5. Given an ST-string STS = $sts_1sts_2\ldots sts_6$ and a QST-string QST = $qs_1qs_2qs_3$, which are shown as follows.

sts_1	sts_2	sts_3	sts_4	sts_5	sts_6
11	21	22	22	32	33
H	H	M	M	M	M
Z	N	Z	Z	P	Z
E	S	S	E	E	S

qs_1	qs_2	Qs_3
H	M	M
E	E	S

The distance D(i, j) for all i = 0, 1, …, 3, j = 0, 1, …, 6 can be calculated as shown in cells of Table 3 and Table 4. In this example, the weights of the feature 2 and 4 are 0.6 and 0.4, respectively.

The base conditions, i.e., the values of the cells in column 0 and row 0, are shown in Table 3 and Table 4. When sts_1 has been processed, the distances between $QST_{1,1}$, $QST_{1,2}$, $QST_{1,3}$ and $STS_{1,1}$, i.e., D(1,1), D(2,1) and D(3,1), are calculated as shown in column 1 of Table 3.

After the symbols sts_1, sts_2 …, sts_6 have been processed, the values of the cells in column 1, 2, …, 6 will be obtained as shown in Table 4.

Table 3. *The distance between $QST_{1,1}$, $QST_{1,2}$, $QST_{1,3}$ and $STS_{1,1}$*

		sts_1	sts_2	sts_3	sts_4	sts_5	sts_6
	0	1	2	3	4	5	6
qs_1	1	0					
qs_2	2	0.3					
qs_3	3	0.8					

Table 4. *The distance between STS and QST*

		sts_1	sts_2	sts_3	sts_4	sts_5	sts_6
	0	1	2	3	4	5	6
qs_1	1	**0**	**0.2**	0.7	1	1.3	1.8
qs_2	2	0.3	0.5	**0.4**	**0.4**	**0.4**	0.6
qs_3	3	0.8	0.6	0.4	0.6	0.6	**0.4**

The fourth row represents the q-edit distance between QST and the prefix of STS, i.e., $STS_{1,j}$, j = 1, 2, …,6. The bold-faced values can be used to indicate the edit operations to transform QST to the one which is matched with STS. The ST-string and the edited QST-string can be shown as follows. Note that the bold-faced symbols show the insertion of QST, and the underlined symbol shows the replacement of QST.

sts_1	sts_2	sts_3	sts_4	sts_5	sts_6
qs_1	$\mathbf{qs_1}$	$\underline{qs_2}$	$\mathbf{qs_2}$	$\mathbf{qs_2}$	qs_3

From the bold-faced values indicated in the previous table, we can find that sts_1 matches qs_1, qs_1 is inserted and one of the feature values is changed to be matched with sts_2, qs_2 is replaced by changing one feature value to be matched with sts_3, and the inserted qs_2 matches with sts_4 and sts_5 without changing any feature value. Therefore, the cost of the edit operations is the total weights of the three insertions and one replacement, i.e., (0.2 + 0 + 0) + 0.2 = 0.4, which can be found in the previous table.

Based on the q-edit distance, the approximate QST-string matching problem is therefore defined as follows.

Definition. *Approximate QST-string Matching Problem*

Given an ST-string STS, a QST-string QST, and a user-defined threshold ε. We say that STS approximately matches QST if there is a substring STS′ such that the q-edit distance between STS′ and QST is smaller than or equal to ε.

In order to solve the approximate QST-string matching problem, a matching algorithm is proposed in the following section.

5. APPROXIMATE QST-STRING MATCHING

As described in Section 3, all the substrings of ST-strings can be found from the index structure. In the definition of the approximate QST-string matching problem, if an ST-string STS approximately matches the QST-string QST, there must be a substring of STS, say STS′, such that the q-edit distance between STS′ and QST is less than or equal to the predefined threshold. Therefore, in order to verify whether the ST-strings recorded in the database matches the QST-string, all the substrings recorded in the index have to be processed and the corresponding distances with the QST-string have to be calculated.

Each substring of the ST-strings can be considered as a prefix of some suffix recorded in the index structure. Since the suffix tree is modified as the KP-Suffix tree in this paper, the length K prefix of all the substrings are indexed by the KP-Suffix tree structure. In order to verify all substrings of the ST-strings recorded in the database, each path of the KP-Suffix tree has to be processed and the corresponding distance with the QST-string will be calculated to verify whether the substring indexed by the path may approximately match the QST-string.

If the q-edit distance between QST and some prefix of a path is less than or equal to the threshold, all the suffixes indexed by the path approximately match QST. Otherwise, a further comparison has to be applied to find the matched results.

Observing the equation of D(i, j), we can find that while computing the values of cells in column i, only the values of cells in column i-1 are referenced. That is, only two columns are needed while calculating the value of cells in each column. Based on the observation, the ST symbols in a path can be processed one by one and the values of the cells in the corresponding column can be sequentially calculated.

Since the values of the cells in each column are calculated based on the values of the cells in the previous column, the minimum value in each column can be considered as the lower bound of the distance, that is, the minimal value in column i cannot be less than the one in column i-1. Therefore, the lower bound can be used to prune some paths which are impossible to be the final result.

Lemma 1. *Lower Bounding Property*

Assume the minimum value of column i is V, and the minimum value of column j is V_j, where $j > i$, then $V_j \geq V$.

Based on the Lower Bounding Property, assume the minimum value of column i is V and the threshold is ε, if V > ε, , then D(j, l) > ε, for all $j > i$. That is, the substring indexed by the path can not be approximately matched against the QST-string. Therefore, while sequentially processing the ST symbols in a path, the minimum value of column i, denoted V_i, can be obtained after the ith symbol has been processed. If V_i is larger than the threshold, the substring indexed by this path can not be the matched answer. That is, the matching procedure of this path can be terminated.

On the other hand, if D(i, l) ≤ ε, the length i prefix of the suffixes recorded in the leaf node of the path match the QST-string. Therefore, all the suffixes indexed by the path will be returned.

Example 6. Following the example in the previous section, suppose the distance threshold is 0.6. The matching process of the path will be terminated after sts_3 has been processed since the minimum value of column 3 is 1 which is larger than the threshold, i.e., 0.6. That is, the values of the cells in column 4~6 do not need to be calculated. On the other hand, if the distance threshold is 1, after sts_2 has been processed, all the suffixes recorded in the leaf node of the path match QST since $STS_{1,2}$ matches QST.

The algorithm of the matching procedure is shown in Figure 4.

6. EXPERIMENTAL RESULTS

```
Algorithm Approximate_Matching (S, N, $\varepsilon$, C )
Input:   The QST-string to be matched $s=s_1s_2...s_l$
         the starting node N, the threshold $\varepsilon$ and the
         initial column C
Output:  The matched results set RS
1.   if N is a leaf node
2.        append N.data to RS
3.   for all edges pointed from node N, denoted $e_i$
4.      let N' is the child node of N pointed by $e_i$.
5.      if $S=\phi$
6.         call Approximate_Matching($\phi$, N', $\varepsilon$, C)
7.      else
8.         for each symbol $s_j$ in $e_i$
9.      Calculate column j $C_j$ based on C and S
10.            $C = C_j$
11.       if $Minimum(C) > \varepsilon$
12.              break
13.       else if the lth element of $C \leq \varepsilon$
14.              $S=\phi$
15.        call Approximate_Matching(S, N', $\varepsilon$, C)
16.  return
```

Figure 4. Algorithm for approximate matching.

In a previous work (Lin & Chen, 2001a), an automatic motion event derivation technique was proposed. Moreover, there are many research results which focus on video object detection (Xu, Younis & Kabuka, 2004). Based on these techniques, we have implemented a semi-automatically annotation interface to derive and record the spatio-temporal information of the video objects as ST-strings.

To show the efficiency of the proposed index structure and matching techniques, we perform a series of experiments on 10,000 ST-strings, with the lengths of the strings being from 20 to 40. The execution time of each experiment is measured by the average elapsed time of matching 100 queries with the ST-strings. We use the 1D-List approach (Lin & Chen, 2003) for the comparison.

Figure 5 and Figure 6 show the performance of the exact QST-string matching mechanism. Figure 5 illustrates the execution time versus the length of the query. Since a QST symbol will be contained in more ST symbols if the value of *q* is small, more paths have to be traversed while matching a QST-string. Therefore, the overall execution time will be larger for smaller q. In our approach, about 70 ms are needed to process a query when the value of *q* is 1 and only 2 ms processing time is needed when the value of *q* is 4.

Figure 6 illustrates the comparison between our approach and the 1D-List approach for different query lengths. Our approach performs much better than the 1D-List approach.

Figure 7 illustrates the execution time versus different thresholds. The execution time increases for larger threshold. For small thresholds, more paths can be pruned based on the lower bounding property. Therefore, the overall execution time for smaller threshold is smaller. Since a QST symbol will be matched with more ST symbols if the value of *q* is small, more paths have to be traversed while matching a QST-string. Therefore, the overall execution time will be larger for smaller q.

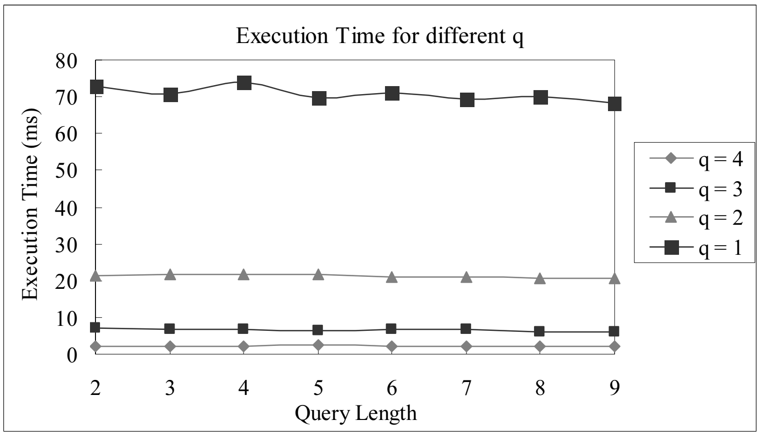


Figure 5. Execution time for matching strings with different lengths (K=4).

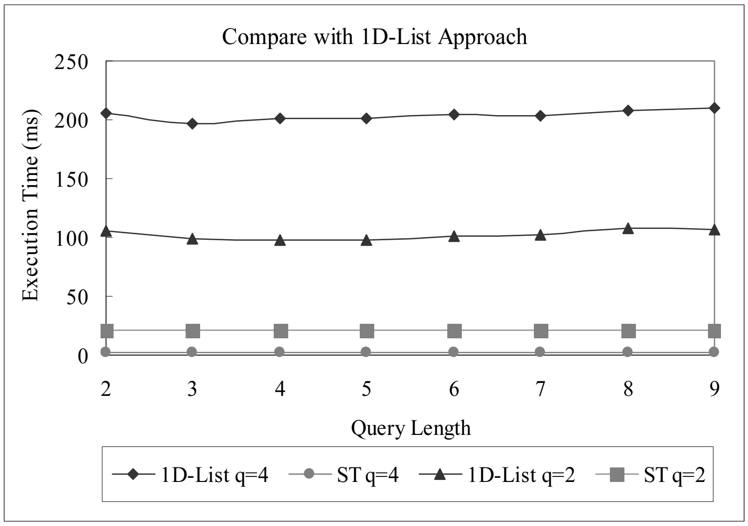


Figure 6. Compare with the 1D-List approach (K=4).

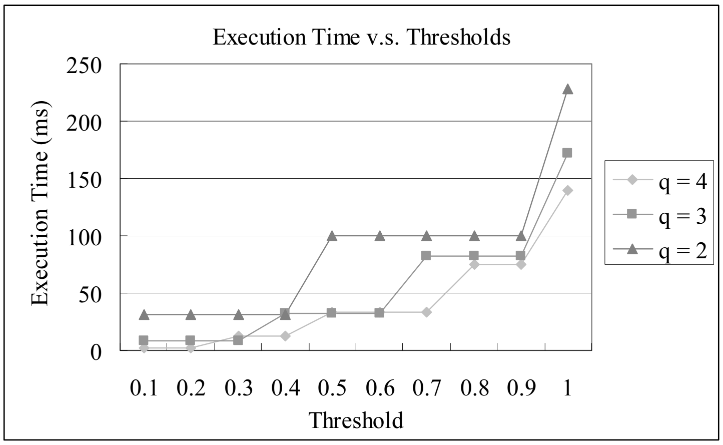


Figure 7. Execution time v.s. thresholds.

7. CONCLUSIONS

Since the spatio-temporal information of video objects can be represented as ST-strings, the retrieval problem is therefore transformed into the approximate QST-string matching problem. This paper proposes an index structure and the matching methodologies for the QST-string matching problem. Moreover, the similarity measure and the corresponding matching methodology are also proposed for the approximate QST-string matching problem. To show the efficiency of the proposed algorithms, we performed a series of experiments. Compared with the 1D-List approach, our approach needs only about 1% to 20% execution time. We are currently working on extending the proposed methodology to the data stream environment. The index structure and the corresponding matching algorithm are currently under development.

REFERENCES

Chen, A. L. P., Chang, M., Chen, J., Hsu, J. L., Hsu, C. H., & Hua, S. Y. S. (2000). Query by music segments: an efficient approach for song retrieval. *Proceedings of IEEE International Conference on Multimedia and Expo*., 873-876, New York, NY, USA.

Gusfield, D. (1997). *Algorithms on strings. trees, and Sequences*, New York, USA: Cambridge University Press.

Jiang, H., & Elmagarmid, A. K. (1998). Spatial and temporal content-based access to hypervideo database. *The VLDB Journal*, *7*(4), 226-238.

Kahveci, T., Singh, A., & Gurel, A. (2002). Similarity searching for multi-attribute sequences. *Proceedings of International Conference on Scientific and Statistical Database Management*, 175-184, Edinburgh, Scotland, UK.

Lee, S. L., Chun, S. J., Kim, D. H., Lee, J. H., & Chung, C. W. (2000). Similarity Search for Multidimensional Data Sequences. *Proceedings of International Conference on Data Engineering*, 599-608, San Diego, California, USA.

Levenshtein, V. I. (1995). Binary codes capable of correcting spurious insertions and deletions of ones. *Problems of Information Transmission*, *1*, 8-17.

Lin, C. H., & Chen, A. L. P. (2001a). Motion Event Derivation and Query Language for Video Databases. *Storage and Retrieval for Media Databases 2001, Proceedings of SPIE, 4315*, San Hose, CA, USA.

Lin, C. H., & Chen, A. L. P. (2001b). Indexing and Query Processing for Video Databases. *Proceedings of IEEE International Conference on Image Processing*, Thessaloniki, Greece.

Lin, C. H., & Chen, A. L. P. (2003). Indexing Strings Representing Multimedia Data for Efficient Query Processing. *Technical Report MAKETR-03-00*.

Lin, C. H., & Chen, A. L. P. (2006). Indexing and Matching Multiple-Attribute Strings for Efficient Multimedia Query Processing. *IEEE Trans. on Multimedia*, *8*(2), 408- 411.

Liu, C. C., & Chen, A. L. P. (2002). 3D-List: A Data Structure for Efficient Video Query Processing. *IEEE Trans. on Knowledge and Data Engineering*, *14*(1), 106-122.

Ren, W., & Singh, S. (2004). Video Sequence Matching with Spatio-Temporal Constrains. *Proceedings of IEEE International Conference on Pattern Recognition*, Cambridge, UK.

Xu, H., Younis, A. A., & Kabuka, M. R. (2004). Automatic Moving Object Extraction for Content-Based Applications. *IEEE Trans. on Circuits and Systems for Video Technology*, *14*(6), 796-812.

Chia-Han Lin received a B. S. degree in information science from TungHai University, Taiwan, in 1993, an M. S. and Ph. D. degree in computer science from National Tsing Hua University, Taiwan, in 1995 and 2005, respectively. He is currently an Assistant Professor in the Department of Information Communication, Asia University, Taiwan. His current research interests are in the area of multimedia data management, data mining and database system.

Arbee L.P. Chen received his Ph. D. degree in computer engineering from the University of Southern California, Los Angeles, in 1984. Dr. Chen is currently Dean of the College of Science and Chengda Chair Professor at National Chengchi University, Taiwan. He also holds a joint professorship at National Tsing Hua University where he has been a professor since 1990. Before returning to Taiwan in 1990, Dr. Chen was a member of technical staff at Bell Communications Research, New Jersey, from 1987 to 1990, an adjunct associate professor in the Department of Electrical Engineering and Computer Science, Polytechnic University, New York, and a research scientist at Unisys, California, from 1985 to 1986. His current research interests include multimedia databases, data mining, and data stream management systems. Dr. Chen organized the 1995 IEEE Data Engineering Conference, the 1999 International Conference on Database Systems for Advanced Applications, and the 2002 International Computer Symposium in Taiwan. He was the program committee co-chair of the 2008 IEEE Data Engineering Conference to be held in Mexico. He was invited to deliver a speech at the US National Science Foundation-sponsored Inaugural International Symposium on Music Information Retrieval at Plymouth, USA, 2000, and in the IEEE Shannon Lecture Series at Stanford University, California 2005. Dr. Chen was a visiting scholar at Tsinghua University, China, 1999, Kyoto University, Japan, 1999, Stanford University, 2003-2005, and King's College London, UK, 2005. He has published more than 170 papers in international journals and conference proceedings, and has been a recipient of the National Science Council's Distinguished Research Award since 1996. He is a senior member of the IEEE Computer Society.