An Approach to Content-based Video Retrieval

Anthony J.T. Lee, Ruey-Wen Hong and Meng-Fang Chang
Department of Information Management
National Taiwan University
No. 1, Sec. 4, Roosevelt Road, Taipei 106, Taiwan R.O.C.
Email:{jtlee, d90725004, r86009}@ntu.edu.tw

Abstract

In this paper, we propose an approach to retrieve the videos similar to the given query video from the database. Our proposed approach consists of three phases. First, every database video is segmented into several shots. Second, for each shot, one or more key frames are selected, and then a feature vector for each key frame is computed. So, every database video is transformed into a sequence of feature vectors. Third, to retrieve the videos similar to the given query video, the query video is also transformed into a sequence of feature vectors. Then, we use a dynamic programming approach to compute the similarity between the query video and each database video. The database videos with the similarity higher than a predefined threshold are output and returned to the user. The experimental results demonstrate that the more key frames are used in the query video, the better precision and recall can be obtained.

1. Introduction

With the advances in information technologies, the amount and volume of multimedia data, such as television broadcasts and surveillance videos, rapidly increases. The content-based video retrieval systems that can effectively retrieve the videos similar to a given query video have been attracted much attention in the recent years.

Pickering et al. [1] proposed the k-nearest neighbors and boosting learning methods to perform video retrieval tasks. Shearer et al. [2] presented the largest common subgraph detection algorithm based on decision trees to retrieve the similar videos. Doulamis et al. [3] used a multidimensional fuzzy histogram for each video to implement content-based indexing and retrieval. However, they are all adapted from content-based image retrieval systems [4][5] and so ignore the frame ordering.

Therefore, in this paper, we propose an approach to retrieve the videos similar to the given query video from the database. Our proposed approach can deal with frame ordering and provide similarity retrieval. Query-by-example is also supported by our approach. Our proposed approach consists of three phases as follows.

(i) Segmentation: Every video in a database is first segmented into several shots.
(ii) Indexing: For each shot, one or more key frames are selected, and then a feature vector for each key frame is computed. Since every database video contains a sequence of key frames, there exists a sequence of feature vectors for the database video. The sequences of feature vectors of database videos are stored in the feature database.
(iii) Searching: The given query video is also segmented into several shots. For each shot, one or more key frames are selected. A feature vector is computed for each selected key frame. So, there also exists a sequence of feature vectors for the query video. Then, we use a dynamic programming approach to compute the similarity between the sequence of feature vectors for the query video and each sequence of feature vectors in the feature database. The database videos with the similarity higher than a predefined threshold are output and returned to the user.

The rest of this paper is organized as follows. Section 2 describes our system model. Section 3 describes the experiment and results. The conclusion is made in Section 4.

2. System model

A video consists of a sequence of frames. We first segment a video into several shots, select a key frame from each shot, and extract a feature vector from each key frame. Features such as color, texture, and shape can be extracted from a key frame. A video can then be represented by a sequence of feature vectors. Computing the similarity between two videos can be transformed into the problem of computing the similarity between two sequences of feature vectors.

To retrieve the similar videos from the database, a query can be formed by a video or a sequence of images. If a query is a video, it is first segmented into several shots, each of which can be represented by a key frame. A feature vector is extracted for each key frame or image.

Thus, a query can be represented by a sequence of feature vectors. The sequence of feature vectors is called the *query sequence*. Then, we use a dynamic programming approach to compute the similarity between the query sequence and each sequence of feature vectors in the feature database. The database videos with the similarity higher than a predefined threshold are output and returned to the user. An example of a query and retrieved database video is shown in Figure 1.

Figure 2 shows an example of a query result. The numbers in the bracket parentheses below the key frames represent the starting and ending frames of the shot. The second line represents the similarity (defined in Section 2.2) between the query video and the retrieved database video, rank order and video id.

(a) Key frames of a query video

(b) Key frames of a retrieved database video

Fig. 1. An example of a query and retrieved database video.

2.1 Video segmentation and key frame selection

Most previously proposed video segmentation algorithms [6][7] compare the distance between feature vectors of two adjacent frames. When the distance is greater than a predefined threshold, the boundary of a shot is discovered and video is segmented. These steps are repeatedly executed up to the last frame of the video. In order to reduce storage space and increase search speed, we segment a video into shots and select a frame to represent a shot. The selected frame is called the key frame. However, if frames in a shot change gradually, a key frame may be quite different from the other frames. That is, if we select the first frame as the key frame, it may be significantly different than the last frame.

To solve this problem, we further divide a shot into several sub-shots. This ensures that the key frame is similar to the other frames in a sub-shot. The feature vector of the key frame is the average of all feature vectors of frames in the sub-shots. When a new frame is added, we recalculate the average of all feature vectors of frames in the sub-shot. All frames in the sub-shot are then compared to the average. If the difference is larger than a predefined threshold, the boundary of a sub-shot is found and segmentation is performed. For convenience, in this paper a shot represents both a shot and a sub-shot.

(a) Query video

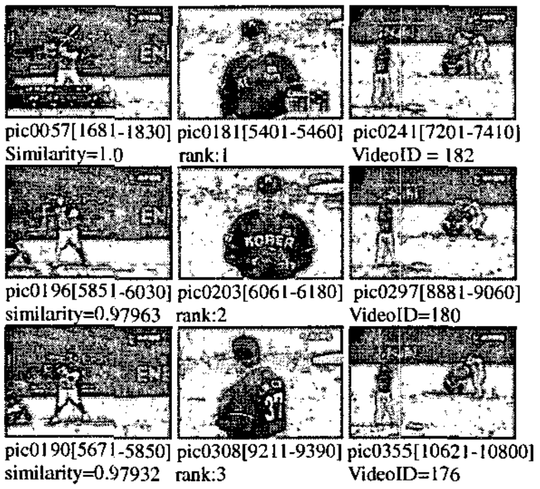
pic0057[1681-1830] pic0181[5401-5460] pic0241[7201-7410]
Similarity=1.0 rank:1 VideoID = 182
pic0196[5851-6030] pic0203[6061-6180] pic0297[8881-9060]
similarity=0.97963 rank:2 VideoID=180
pic0190[5671-5850] pic0308[9211-9390] pic0355[10621-10800]
similarity=0.97932 rank:3 VideoID=176
(b) Query result

Fig. 2. An example of the query video and result.

2.2 Definition of similarity

Since a video is segmented into shots and each shot is represented by a key frame associated with a feature vector whose length is equal to one, we define the similarity between two shots as $S(q,v) = 1 - \frac{D(q,v)}{\sqrt{k}}$, where q and v are the feature vectors of both shots, k is the dimension of a feature vector and D(q, v) is the Euclidean distance between q and v.

Say that a user inputs a query video Q, and video V is very similar to query Q, meaning that video V contains most of shots similar to Q's shots. These shots will appear in the same order as they do in Q. That is, if V's shots are not similar to Q's, or matching shots aren't in the same order, then the similarity between Q and V is low.

From this concept, we can define the similarity between videos. Assume that Q is segmented into m shots, $Q_1, Q_2, \ldots, Q_m$. Let q_i be the feature vector of key frame of Q_i, $1 \le i \le m$. V is segmented into n shots, V_1,

$V_2, \ldots, V_n$, where $n \geq m$. Let v_j be the feature vector of key frame of V_j, $1 \leq j \leq n$. If $S(q_i,v_j)$ is higher than a predefined threshold, we say that Q_i and V_j is similar, where $1 \leq i \leq m$ and $1 \leq j \leq n$. Let V_{subset} be a subset of $\{V_1, V_2, \ldots, V_n\}$ and shots in V_{subset} have the same order as those in V. Assume that there exists $V_{subset} \subseteq \{V_1, V_2, \ldots, V_n\}$ such that the similarity between Q and V_{subset} achieves the optimum. We denote it by $VS(Q, V)$ and use it to represent the similarity between Q and V. $VS(Q, V)$ is defined as the average value of the similarity of matched shots, which is computed by $VS(Q,V) = \frac{\sum_{i=1}^{m} S(q_i, v_{x_i})}{m}$, where Q_i matches V_{xi}, $1 \leq i \leq m$ and $x_j < x_k$ if $j<k$. The detailed steps to computed $VS(Q,V)$ is described in Section 2.3.

If V contains every key frame of Q and those key frames are in the same order as those in Q, then the similarity between Q and V is 1. If the key frames of V are not similar to those of Q or are in a different order, the similarity is low.

2.3 Dynamic programming algorithm

The property of computing the similarity between Q and V is suitable for the utilization of a dynamic programming approach. The problem can be formulated as shown in Figure 3, where M_{ij} denotes the optimal similarity when shots $Q_1, Q_2, \ldots, Q_i$ match shots $V_1, V_2, \ldots, V_j$. So, $VS(Q,V)=M_{mn}$.

$$M_{ij} = \begin{cases} S(q_i,v_j) + Max(M_{i-1,i-1},\ldots,M_{i-1,j-1}) & j = i..n-m+i \\ S(q_i,v_j) & i = 1 \\ -\infty & (j<i) \vee (j>n-m+i) \end{cases}$$

Fig. 3. Computing video similarity

If we solve the equation shown in Figure 3 recursively, the time complexity is bounded by $O(mn^2)$. This is too high, thus we reduce time complexity by the algorithm as shown in Figure 4. Let $MS[i][j]$ be the optimal similarity when shots $Q_1, Q_2, \ldots, Q_i$ match shots $V_1, V_2, \ldots, V_j$. We calculate the similarity between Q_i and V_j and store it in array $S[i][j]$ so that we don't need to calculate it each time.

The time complexity of **VideoMatching** algorithm is bounded by $O(m(n-m))$, where n is the number of key frames in V and m is the number of key frames in Q, $n \geq m$.

To demonstrate how the algorithm works, for simplicity, assume that all feature vectors are one-dimensional and their values are between 0 and 1. Also assume that the feature vectors extracted from the query video Q are {(0.8), (0.1), (0.5)} and the feature vectors extracted from the database video V are {(0.2), (0.3), (0.6), (0.3), (0), (0.2), (0.3), (0.1)}. First of all, the similarity between each shot of Q and each shot of V can be computed as shown in Figure 5.

We then calculate S[i][j] and MS[i][j] row by row. The result is shown in Figure 6. The first number in each cell is S[i][j] and the second number is MS[i][j].

```
VideoMatching(Q, V)
  (1) for $i \leftarrow 1$ to $m$
  (2)   for $j \leftarrow i$ to $n-m+i$
  (3)     $S[i][j] \leftarrow S(q_i, f_j)$
  (4) $MS[1][1] \leftarrow S[1][1]$
  (5) for $j \leftarrow 2$ to $n-m+1$
  (6)   $MS[1][j] \leftarrow$ MAX($S[1][j]$, $MS[1][j-1]$)
  (7) for $i \leftarrow 2$ to $m$
  (8)   for $j \leftarrow i$ to $n-m+i$
  (9)     $MS[i][j] \leftarrow$ MAX($S[i][j] + MS[i-1][j-1]$, $MS[i][j-1]$)
  (10) return $MS[n][m] / m$
```

Fig. 4. Video matching algorithm

	0.2	**0.3**	**0.6**	**0.3**	**0**	**0.2**	**0.3**	**0.1**
0.8	0.4	0.5	0.8	0.5	0.2	0.4		
0.1		0.8	0.5	0.8	0.9	0.9	0.8	
0.5			0.9	0.8	0.5	0.7	0.8	0.6

Fig. 5. Shot similarity between Q and V

	0.2	**0.3**	**0.6**	**0.3**	**0**	**0.2**	**0.3**	**0.1**
0.8	*S*=0.4 *MS*=0.	0.5 0.5	0.8 0.8	0.5 0.8	0.2 0.8	0.4 0.8		
0.1		0.8 1.2	0.5 1.2	0.8 1.6	0.9 1.7	0.9 1.7	0.8	
0.5			0.9	0.8	0.5	0.7	0.8	0.6

Fig. 6. Calculation of S[i][j] and MS[i][j]

	0.2	**0.3**	**0.6**	**0.3**	**0**	**0.2**	**0.3**	**0.1**
0.8	0.4 0.4	0.5 0.5	0.8 0.8	0.5 0.8	0.2 0.8	0.4 0.8		
0.1		0.8 1.2	0.5 1.2	0.8 1.6	0.9 1.7	0.9 1.7	0.8 1.7	
0.5			0.9 2.1	0.8 2.1	0.5 2.1	0.7 2.4	0.8 2.5	0.6 2.5

Fig. 7. The resultant matrix where the optimal matched cells are circled.

The resultant matrix is shown in Figure 7. The similarity between Q and V is 2.5/3=0.83. The optimal match between Q and V is that Q's first key frame matches V's third key frame, Q's second key frame matches V's fifth key frame, and Q's third key frame matches V's seventh key frame.

3. Experiments and Performance Evaluation

All the experiments are performed on a Pentium III-800 personal computer with 256MB main memory, running Windows 2000. The program is written in C++ and compiled with Microsoft Visual C++ 6.0. Figure 8 shows the experiment database, which consists of 200 videos from six different categories.

Category	*Quantity*	*Category*	*Quantity*
News	50	Cartoons	20
Sports	50	TV Shows	40
Advertisements	20	Movies	20

Fig. 8. Experiment database

After segmenting all videos in the experiment database, the database contains 11,980 shots and 758,815 frames. Queries are formed by a sequence of images which are picked up from ten randomly selected database videos. First, we select a single key frame from a video to form a length-one query. We then select another key frame from the video and add it into the query, according to their original appearance order. We repeat this procedure until there are ten key frames in a query and thus we have ten queries of different lengths. Repeating the same procedure on ten selected videos produces 100 queries. In this expreriment, the feature vectors of spatial color histogram and color correlogram are used.

In Figure 9, we demonstrate the effect of the number of query frames on query quality. The result shows that, in most cases, the more query frames are used, the better precision and recall can be obtained. However, the improvement appears quite limited while increasing the number of query frames from 5 to 7. Using 5 query frames seems to be good enough to achieve a satisfied result in most of cases.

4. Conclusion

In this paper, we propose an approach to content-based video retrieval. Our proposed approach consists of three phases. First, every video in the database is segmented into several shots. Second, for each shot, one or more key frames are selected, and then a feature vector for each key frame is computed. Since every database video contains a sequence of key frames, there exists a sequence of feature vectors for the database video. The sequences of feature vectors of database videos are stored in the feature database. Third, the given query video is also segmented into several shots. For each shot, one or more key frames are selected. A feature vector is computed for each selected key frame. So, there also exists a sequence of feature vectors for the query video. Then, we use a dynamic programming approach to compute the similarity between the sequence of feature vectors of the query video and each sequence of feature vectors in the feature database. The database videos with the similarity higher than a predefined threshold are output and returned to the user.

Our proposed approach can deal with frame ordering and provide similarity retrieval. Query-by-example is also supported by our approach.

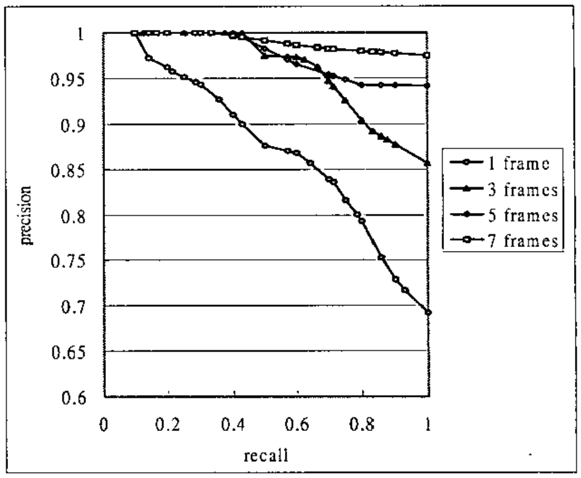

Fig. 9. Effect of the number of query frames

References

[1] Pickering, M. J., Ruger, S., "Evaluation of key frame-based retrieval techniques for video", *Computer Vision and Image Understanding,* Vol. 92, No. 2-3, 2003, pp. 217-235.

[2] Shearer, K., Bunke, H., Venkatesh, S., "Video indexing and similarity retrieval by largest common sub-graph detection using decision trees", *Pattern Recognition,* Vol. 34, No. 5, 2001, pp. 1075-1091.

[3] Doulamis, A. D., Doulamis, N. D., Kollias, S. D., "A fuzzy video content representation for video summarization and content-based retrieval", *Signal Processing,* Vol. 80, No. 6, 2000, pp. 1049-1067.

[4] Woo, H., Jang, D.S., Jung, S.H., Park, J.H., Song, K.S., "Visual information retrieval system via content-based approach", *Pattern Recognition,* Vol. 35, No. 3, 2002, pp 749-769.

[5] Eakins, J.P., "Towards intelligent image retrieval", *Pattern Recognition,* Vol. 35. No. 1, 2002, pp 3-14.

[6] Bertini, M., Bimbo, A.D., Pala, P., "Indexing for reuse of TV news shots", *Pattern Recognition,* Vol. 35, 2002, pp. 581-591.

[7] Ralph, M.F., Robson, C., Temple, D., Gerlach, M., "Metrics for shot boundary detection in digital video sequences", *Multimedia Systems,* Vol. 8, 2000, pp. 37–46.