Mining Repetitive Clips through Finding Continuous Paths

Junsong Yuan[1], Wei Wang[2], Jingjing Meng[2], Ying Wu[1], Dongge Li[2]
[1]Northwestern University, EECS Dept.
2145 Sheridan Road
Evanston, IL, USA 60208
[2]Motorola Application Research Center
1295 East Algonquin Road
Schaumburg, IL, USA 60196

ABSTRACT

Automatically discovering repetitive clips from large video database is a challenging problem due to the enormous computational cost involved in exploring the huge solution space. Without any *a priori* knowledge of the contents, lengths and total number of the repetitive clips, we need to discover all of them in the video database. To address the large computational cost, we propose a novel method which translates *repetitive clip mining* to the *continuous path finding* problem in a matching trellis, where sequence matching can be accelerated by taking advantage of the temporal redundancies in the videos. By applying the locality sensitive hashing (LSH) for efficient similarity query and the proposed continuous path finding algorithm, our method is of only *quadratic* complexity of the database size. Experiments conducted on a 10.5-hour TRECVID news dataset have shown the effectiveness, which can discover repetitive clips of various lengths and contents in only 25 minutes, with features extracted off-line.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining, Image databases*

General Terms

Algorithms, Performance

Keywords

repetitive pattern discovery, video data mining

1. INTRODUCTION

Recent research of repetitive clip discovery present many potential applications in multimedia database, including commercial recognition and detection [5], object skipping for compression [2], news topic detection and tracking [11] [4], news broadcast structure analysis [9] [7].

Compared with video clip search, where an input query is usually provided, repetitive clip mining is a more challenging problem because of its unsupervised characteristics [8]. For example, it is generally *a priori* unknown (i) where the repetitive clips are and (ii) how long they are; or even (iii) whether such repetitive clips exist. Exhaustive search through all possible clip lengths and locations is computationally demanding for large database, *e.g.*, can be of complexity $O(N^4)$ with database size N (see the Appendix). One possible solution to address the large computational cost is through translating video sequences into symbolic sequences, where efficient matching methods can be applied in finding video repetitions [7]. However, the quantization errors introduced in such a translation may significantly affect the performance, especially when the inappropriate cluster number is selected. In [9], a self-similarity matrix is proposed for video repetition mining, with computational cost of complexity $O(N^3)$ due to the exhaustive search of the possible clip lengths. Some other repetition discovery methods in image or video shot level, such as near-duplicate image detection [11] and identical shot detection [4] [6], cannot be directly extended to repetition mining in video clip level, where repetitive clips are likely of various lengths, and are composed of different number of shots.

We propose an efficient repetitive clip mining method in this paper, with complexity less than $O(N^2)$. Firstly we chop the long video sequence into fixed length *video segments* (VS) with overlaps. The similarity between two VS is measured based on their visual signatures. Such signatures combine the color and spatial-temporal information together, thus are discriminative features and are also robust to video coding variations. For each VS, we search for its best matches in the database, through the ϵ-nearest neighbor (ϵ-NN) query. Once obtained the ϵ-NN query results of each VS, a matching trellis can be built as illustrated in Fig. 1. The repetition mining problem can thus be formulated as a continuous path finding problem in the trellis. Through searching for the continuous paths in the trellis, our method can (1) automatically discover all the repetitive clips without knowing the total number of them in advance; and (2) dynamically determine the length of each repetitive clip instead of exhaustive search. By applying Locality Sensitive Hashing for efficient ϵ-NN query, building the trellis is of complexity less than $O(N^2)$. The proposed continuous path search is of complexity $O(NK^2)$. Thus the total complexity of our method is less than $O(N^2)$. We introduce our algorithm in Sec. 2, followed by the experiments (Sec. 3) and conclusion (Sec. 4).

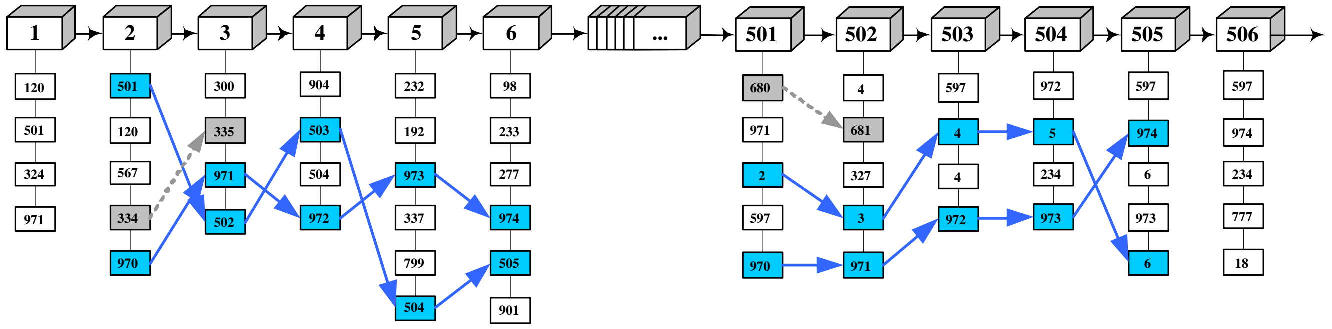

Figure 1: Mining repetitive clips by searching "continuous" paths. Each node in the trellis denotes a fixed-length video segment (VS) labeled by its temporal index. The VS sequence in the top-row denotes the video database of length N, and each column represents the matches of the VS in the top-row. Take the first column for example, the 1_{st} VS has four matches in the database: $\{120_{th}, 501_{st}, 324_{th}, 971_{st}\}$. There are four continuous paths (highlighted) in the trellis, where each path is composed of a sequence of VS with consecutive temporal indices. These continuous paths indicate repetitions of the corresponding sequence in the top-row. For example, the VS sequence $[2-6]$ in the top-row has two repetitions at positions $[501-505]$ and $[970-974]$ (*blue path*). The *gray paths* ($[334-335]$ and $[680-681]$) are considered invalid because they are not long enough.

2. ALGORITHM

2.1 Visual signature extraction

For video matching, it is desirable that the visual features are robust under video coding variations, such as compression rate, frame size and color format changes. Moreover, the visual features should also be unique enough to identify different videos. Considering the above two requirements, we choose two types of compact visual signatures to characterize each VS: (1) color signature and (2) spatial signature.

Color Signature

To characterize the color information of a VS, we use three average color histograms of all its frames, corresponding to three color channels. Let h_i^C denote the color histogram of the i_{th} frame under color channel $C \in \{Y, C_b, C_r\}$, the average color histograms (normalized) to characterize a VS are denoted as:

$$\mathbf{H}^C = \frac{1}{L}\sum_{i=1}^{L} h_i^C, \quad and \quad \sum_{j=1}^{B} \mathbf{H}^C(j) = 1, \tag{1}$$

where L is the length of the clip, $B = 24$ is the number of bins of the histogram. The summation of two histograms is the summation of each corresponding bin.

Spatial Signature

To compensate for the spatial information missing from the color signatures, we apply the ordinal measure method [10] to generate spatial signatures. Firstly, an image is partitioned into $m \times n$ equal size grids. We then calculate the average intensity value of each of the $m \times n$ sub-images. By performing the ordinal measure, these sub-images are ranked based on their intensity values. The brightest sub-image, which is of the highest intensity, is ranked as the 1_{th}, and the darkest sub-image is ranked as the $(m \times n)_{th}$. In our implementation, we partition an image into 2×2 sub-images, hence there are in total $(2 \times 2)! = 24$ possible ordinal measure results for an image. We associate each possible ordinal measure result to an integer pattern code p. Therefore an image $\mathbf{I}$ can be mapped into one of these 24 pattern codes after ordinal measure: $\mathcal{M}(\mathbf{I}) = p \in \{1, 2, ..., 24\}$, where $\mathcal{M}(\cdot)$ is the ordinal measure function. If two images are identical, then their ordinal measure values should also be the same. Similar to the color signatures, such spatial signatures are invariant to coding variations. Moreover, the spatial signatures are insensitive to global color shifting, *i.e.*, the rank of the sub-images are stable.

In order to obtain more distinguishable spatial signatures to identify different images, we apply *three* different spatial layouts when partitioning an image, which is illustrated in Fig. 2. An image is mapped three times, corresponding to ordinal measure functions: $\mathcal{M}_1(\cdot)$, $\mathcal{M}_2(\cdot)$ and $\mathcal{M}_3(\cdot)$ respectively. Given two *different* images $\mathbf{I}_a$ and $\mathbf{I}_b$, the error probability that all of their three hash values are identical is as small as:

$$P[\mathcal{M}_1(\mathbf{I}_a) = \mathcal{M}_1(\mathbf{I}_b), \mathcal{M}_2(\mathbf{I}_a) = \mathcal{M}_2(\mathbf{I}_b), \mathcal{M}_3(\mathbf{I}_a) = \mathcal{M}_3(\mathbf{I}_b)]$$
$$= (\frac{1}{(m \times n)!})^3 = (\frac{1}{24})^3 = 0.000072,$$

where we assume $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$ are independent.

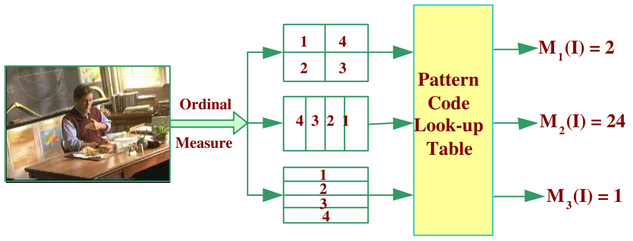

Figure 2: Ordinal measure for a single image under three different spatial layouts: 2×2, 1×4 and 4×1.

To characterize a VS of length L, we apply the ordinal measure to every frame and accumulate all these pattern codes $\mathcal{M}(\mathbf{I}_k)(k = 1, 2, ..., L)$ to form a histogram. As we have three spatial layouts, there are three corresponding *pattern histograms* (normalized):

$$\sum_{j=1}^{B} \mathbf{J}^M(j) = 1, \quad M \in \{\mathcal{M}_1, \mathcal{M}_2, \mathcal{M}_3\}, \tag{2}$$

where M denotes the spatial layout, $B = 24$, and each bin $\mathbf{J}^M(j)$ presents the percentage of frames that share the same pattern code j. These pattern histograms describe the spatial-temporal pattern of the VS.

Finally, to combine the color and spatial signatures, we concatenate the 3 color histograms and 3 spatial pattern histograms into a single normalized histogram $\mathbf{F}$. Since each individual histogram is of 24-dimension, finally a VS is characterized by a feature vector: $\mathbf{F} \in \mathcal{R}^d$ $(d = 24 \times 6 = 144)$.

2.2 Repetitive clip discovery

To build the trellis in Fig. 1, the first step is to find the matches of each VS to establish columns in the trellis. We briefly explain the ϵ-NN query problem as follows. Suppose the video database is represented as a set of VS: $\mathcal{V} = \{v_i\}_{i=1}^n$, where i denotes the temporal index. Given a query $v_i \in \mathcal{V}$, namely a feature vector $\mathbf{F}_i \in \mathcal{R}^d$, our task of ϵ-NN query is to find $v_j \in \mathcal{V}$, such that $\|\mathbf{F}_i - \mathbf{F}_j\| \leq \epsilon$, where $\epsilon \geq 0$ is a matching threshold and $\|\cdot\|$ denotes the Euclidean distance. To avoid redundant matches caused by segment overlaps, *e.g.*, v_i may match v_{i+1} due to heavy overlaps, we filter those temporal neighbors from the ϵ-NN query results for each VS. Formally, we define the retrieved ϵ-NN of a VS v_i as its *match-set* $\mathcal{N}_i$, which corresponds to the i_{th} column in the trellis:

$$\mathcal{N}_i = \{v_j : \|\mathbf{F}_j - \mathbf{F}_i\| \leq \epsilon, |i - j| \geq \tau\}, \quad (3)$$

where we set $\tau = 15$ seconds as the threshold. It is worth noting that exhaustive linear search for ϵ-NN is computationally expensive for large database (*e.g.* $O(N)$), and we need to query N times. To speed up the ϵ-NN search, we apply the Locality Sensitive Hash (LSH) method [1], which provides a randomized solution for the high-dimensional ϵ-NN query problem. Instead of searching for the exact ϵ-NN, LSH searches for the approximate ϵ-NN and thus can trade the performance for the search speed. Sub-linear complexity can be obtained by using LSH for ϵ-NN query.

After building the matching trellis, the next step is to find continuous paths. However, discovering all the continuous paths through an exhaustive check is computationally expensive. Suppose K is the average length of the columns (*i.e.* the average size of the ϵ-NN set) and N is the length of the trellis, there are in total K^N full paths (of length N) in the trellis. For each full path, we need to check if it contains short continuous sub-path. Thus the total cost for exhaustive check is $O(NK^N)$.

Motivated by the idea of dynamic programming, we propose a novel method that can efficiently discover continuous paths in $O(NK^2)$. For each pair of neighboring v_i and v_{i+1} in the original VS sequence, we compare their match-sets $\mathcal{N}_i$ and $\mathcal{N}_{i+1}$ to check if their match-sets contain continuous pairs:

$$\exists v_k \in \mathcal{N}_i, v_l \in \mathcal{N}_{i+1}, \quad such \;\; that \quad L \leq l - k \leq U, \quad (4)$$

where k and l are the temporal indices of the continuous pair; L and U are the continuation requirements. Fig. 1 gives an example when applying strict requirement $L = U = 1$: v_k and v_l should be consecutive VS. For example, there are three continuous pairs between match-sets $\mathcal{N}_2$ and $\mathcal{N}_3$: $(501, 502)$, $(971, 972)$, $(334, 335)$, where $\mathcal{N}_2 = \{501, 120, 567, 334, 970\}$ and $\mathcal{N}_3 = \{300, 335, 971, 502\}$. Each found continuous pair (v_k, v_l) either initializes a new continuous path or is a continuation of an existing path. At each time step, we check the existing paths to see if they can grow in the next step, *i.e.*, can find a continuous pair in the next step. If an existing continuous path cannot grow in the next step, it will be terminated. To finish the whole trellis, we need to check $N - 1$ steps, where for each step we need to compare K^2 pairs (on average) to find continuous pairs. Therefore the total cost to discover all the continuous paths is of complexity around $O(NK^2)$.

When a continuous path is discovered in the trellis, we represent it by its start and end frame: $P = [s, e]$, $s, e \in \mathcal{V}$. A continuous path is valid only if it lasts long enough. Finally we collect all the valid continuous paths to form the candidate set $\mathcal{P} = \{P_i : |P_i| \geq \lambda\}_{i=1}^T$, where $\lambda = 5$ seconds is the minimum valid length. To recover all repetitive clips in the original video sequence $\mathcal{O} = \{\mathbf{I}_i\}_{i=1}^N$, we map each continuous path $P_i \in \mathcal{P}$ back to $\mathcal{O}$. For example, given a continuous path $P = [s, e] \in \mathcal{P}$, we vote for each frame $s \leq \mathbf{I}_i \leq e$ in the original sequence $\{\mathbf{I}_i\}_{i=1}^N$. The more votes a frame $\mathbf{I}_i$ receive, the more repetitive instances it has. For a VS that does not repeat in the database, none of its frames will be voted.

3. EXPERIMENTAL RESULTS

The video database contains 22 streams of half-hour ABC news video collected from the TRECVID dataset. All these half-hour segments are combined into one video sequence with total length of 10.5 hours. The frame rate is 30 fps and frame size is 352×240 or 352×264. We also collect a set of 47 commercial clips from the video database for evaluation. The length of these commercials varies from 15 to 60 seconds. Each of these commercials has 2 to 8 re-occurrences in the database and we obtain the ground truth through exhaustive search. These 47 repetitive clips have in total 153 instances in the database, which may suffer from slight color variations. Our task is to discover all these repetitive commercials. As commercials contain dynamic contents, it is difficult to obtain accurate shot boundaries. Consequently, shot-based approach is not suitable in discovering the whole commercial clip. An ϵ-NN trellis is built for this 10.5-hour sequence, where each node denotes a VS of 8 seconds with an interval of 0.4 seconds. Thus each VS has an overlap of 7.6 seconds with its neighbors. We have in total $N = 94,512$ VS in the database. All the experiments were performed on a standard Pentium-4 3.19GHz PC with 1 GB RAM. The algorithm is implemented in C++.

3.1 Computational cost

The computational costs of our mining method are from two aspects: (1) building of the ϵ-NN Trellis, and (2) continuous path search through the trellis. Table 1 compares the computational CPU cost of (1) using our proposed algorithm, (2) the self-similarity matrix method proposed in [9] and (3) the naive method (discussed in the Appendix) but using a fast linear search method in [3], which can search for a clip in a 24-hour video database in less than 1 second. We estimate the CPU cost of method [9] by assuming the possible clip lengths it needs to exhaustively check are from 5 to 125 seconds.

The ϵ-NN query speed is very fast by applying LSH with $\epsilon = 0.1$. In our database of size $N = 94,512$, the average time for each ϵ-NN query is only 15 milliseconds. The average number of ϵ-NN found for each VS is around $K = 24$. Thus the trellis is of size approximately $K \times N = 24 \times 94,512$. Compared with [9], which uses an $N \times N$ self-similarity matrix to describe the relations between VS, our trellis is a more compact representation, of size around $K \times N$ ($K << N$). In discovering continuous paths from the matching trellis, the continuous requirements in Eq. 4 are set as: $L = 0$ and $U = 2$.

Table 1: Computational Cost Comparison

Method	Complexity	CPU Cost*
build of the ϵ-NN Trellis	$< O(N^2)$	1357.7 sec
continuous path finding	$O(NK^2)$	101.4 sec
proposed method (total cost)	$< O(N^2)$	1459.1 sec
self-similarity matrix [9]	$O(N^3)$	689 min
naive method [3]	$O(N^4)$	41335 h

* file I/O cost is not included

3.2 Repetitive clip mining results

We evaluate the performance of our method in both simulated data and real TRECVID news video data. The simulation is on a 5-hour video. We manually insert 10 different clips into this 5-hour video database. Each clip is 30-second long and has 5 re-occurrences in the database. A continuous path is valid only if it can last for 10 seconds. We evaluate the performance in the simulated data by the precision and recall scores as follows:

$$Recall = \#detects/(\#detects + \#miss\ \ detects);$$
$$Precision = \#detects/(\#detects + \#false\ \ detects).$$

For the TRECVID news video data, we evaluate the performance by comparing the discovered repetitive video sequences with the ground truth obtained from exhaustive search for the 47 commercials. The recall score reflects how many re-occurrences of these 47 commercials are discovered; however, the precision score is not provided because some discovered repetitions, such as anchor person shots, are *not* included in our 47-commercial set. Fig. 3 presents some repetitions discovered in the TRECVID dataset. It is worth noting that the anchor person shots are actually not identical from each other, *e.g.* they are of various lengths and reporting different news. Because it is hard to decide whether anchor person shots should be treated as correct detections, which is application dependent, we exclude the precision score since it cannot fairly evaluate the performance.

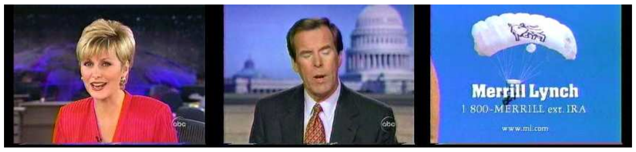

Figure 3: Repetitive clip discovery examples from the TRECVID dataset: anchor person shot 1 (*left*), anchor person shot 2 (*middle*) and commercial (*right*).

The following table 2 shows the recall scores for the simulated and real data; and the precision score for the simulated data only. The miss detection is mainly caused by the discontinuation in finding continuous paths. Many possible affects can introduce such discontinuations, such as imperfectness of the visual signatures due to color and frame size variations, and miss retrieval of LSH.

Table 2: Performance Evaluation

Data	Precision	Recall
Simulated Data (5 *h*)	93.1%	95.0%
TRECVID video (10.5 *h*)	*N.A.*	86.3%

4. CONCLUSION

This paper presents a novel repetitive clip mining algorithm for video database with quadratic complexity. The algorithm is fast, feature independent and fully automatic. Although our experiments only dealt with video data, the proposed method can be easily extended to other types of sequence database for mining repetitive temporal patterns, where the only requirement is to define the similarity between segments.

APPENDIX

Let N denote the database size, we need to check for candidate clips of various lengths and at any possible locations for repetition mining. The total number of candidates for repetitive clips can be counted as follows, where $N-K+1$ denotes the possible number of locations for the clip of length K:

$$\sum_{K=1}^{N}(N-K+1)\times K = \sum_{K=1}^{N}((N+1)K - K^2)$$
$$= \frac{(N+1)(N+1)N}{2} - \frac{N(N+1)(2N+1)}{6}$$
$$= \frac{N^3+3N^2+2N}{6}.$$

Thus our candidate pool is of size $O(N^3)$. For each candidate, we need to search for the whole database to find re-occurrences. Finally the total cost for the naive method is of complexity $O(N^4)$.

5. REFERENCES

[1] M. Datar, N. Immorlica, P. Indyk, and V. Mirrokni. Locality-sensitive hashing scheme based on p-stable distribution. In *Proc. DIMACS workshop on Streaming Data Analysis and Mining*, 2003.

[2] C. Herley. Argos: Automatically extracting repeating objects from multimedia streams. *IEEE Trans. on Multimedia*, 8(1):115–129, 2006.

[3] K.Kashino, T.Kurozumi, and H.Murase. A quick search method for audio and video signals based on histogram pruning. *IEEE Trans. on Multimedia*, 2003.

[4] K.M.Pua, J.M.Gauch, S.E.Gauch, and J.Z.Miadowicz. Real time repeated video sequence identification. *Computer Vision and Image Understanding*, 2004.

[5] R.Lienhart, C.Kuhmuench, and W. Effelsberg. On the detection and recognition of television commercials. In *Proc. IEEE Conf. on Multimedia Computing and Systems*, 1997.

[6] S.Satoh. News video analysis based on identical shot detection. In *Proc. IEEE Conf. on Multimedia Expo*, 2002.

[7] P. Wang, Z.-Q. Liu, and S.-Q. Yang. A probabilistic template-based approach to discovering repetitive patterns in broadcast videos. In *Proc. ACM Multimedia*, 2005.

[8] L. Xie, L.Kennedy, S.-F. Chang, A. Divakaran, H. Sun, and C.-Y. Lin. Discovering meaningful multimedia patterns with audio-visual concepts and associated text. In *Proc. IEEE Conf. on Image Processing*, 2004.

[9] X. Yang, P. Xue, and Q. Tian. A repeated video clip identification system. In *Proc. ACM Multimedia*, 2005.

[10] J. Yuan, L.-Y. Duan, Q. Tian, and C. Xu. Fast and robust short video clip search using an index structure. In *Proc. ACM Multimedia Workshop on Multimedia Information Retrieval*, 2004.

[11] D.-Q. Zhang and S.-F. Chang. Detecting image near-duplicate by stochastic attributed relational graph matching with learning. In *Proc. ACM Multimedia*, 2004.