GENRE-ADAPTIVE NEAR-DUPLICATE VIDEO SEGMENT DETECTION

Ichiro IDE[*], *Kazuhiro Noda*[†], *Tomokazu Takahashi*[‡], *Hiroshi Murase*

Nagoya University, Graduate School of Information Science
Furo-cho, Chikusa-ku, Nagoya, 464–8603, Japan
ide@is.nagoya-u.ac.jp

ABSTRACT

This paper proposes a fast and accurate method to detect all near-duplicate segments in a video stream. To reduce the computation time while ensuring the detection accuracy equivalent to that by brute-force frame-by-frame comparison, a two-step detection method is proposed; a fast but rough detection applied in a compressed feature vector space spanned by the result of a PCA, followed by confirmation of candidates in the original high dimension space. The results show that the proposed method accelerates the detection by more than 1,000 times while maintaining the detection accuracy. We also propose an entropy-based pixel selection scheme to generate feature vectors optimized for comparison of video segments within programs with mostly common pictures. The results show that the proposed scheme eliminates the false positives drastically, which should lead to even faster detection.

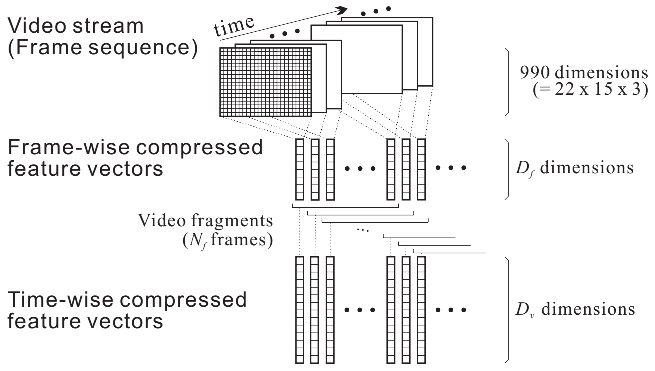

Fig. 1. Spatiotemporal feature vector compression.

1. INTRODUCTION

Recent advance in storage technologies has provided us with the ability to archive many hours of video streams accessible as online data. We have been working on detecting every single pair of *near-duplicate* segments from a long broadcast video stream in realistic computation time.

Near-duplicate video segments are mostly identical video segments from the image perspective, except for minor local differences such as overlay of captions or logos, or minor overall color difference. Detecting near-duplicates is different with the traditional similar video segment detection which detects not only mostly identical but also somewhat similar, but originally different video segments. Detection of near-duplicates is very important to understand semantic structures in video streams by detecting the repetitions.

Detecting all pairs of near-duplicates in a video stream is, however, extremely time-consuming compared to detecting those of a given segment, since it essentially requires computation of a square order of the video length. Due to this nature, attempts to detect near-duplicates have been either approached by brute-force frame-by-frame comparison for a relatively short video stream, or by selecting representative frames from video segments [1, 2, 3]. In order to detect near-duplicates from a long video stream, few efficient detection methods have been proposed. Alike the method proposed in this paper, their major approach is to take a two step detection strategy; fast but rough detection in the first step, and slow but accurate confirmation in the second step. Sekimoto et al. proposes a method by clustering video fragments represented as a sequence of VQ histograms [4]. Naturel and Gros proposes a method that makes use of hash indices of features represented by DCT coefficients [5]. Yamagishi et al. proposes a method that compares frames by normalized cross correlation (NCC) of their intensity histograms, accelerated by the SR-Tree indexing method [6]. Yang et al. proposes a method that compares frames by video correlation analysis combined with locality sensitive hashing of color fingerprints [7]. These methods do accelerate the detection, but they do not necessarily guarantee that the detection in the first step has no false negatives. Considering this problem, the method proposed in this paper guarantees that it has no false negatives in the first step; the results are theoretically equivalent to the brute-force frame-by-frame comparison.

In the following sections, first, a method that detects near-duplicate video segments efficiently by feature compression is introduced in Sect. 2. Next, to make the method more efficient, a genre-adaptive feature extraction method that selects pixels with high dissimilarity across frames is introduced in Sect. 3. Section 4 concludes the paper.

[*]Also affiliated with National Institute of Informatics
[†]Currently at DENSO Corporation
[‡]Affiliated with Japan Society for the Promotion of Science

2. EFFICIENT DETECTION OF NEAR DUPLICATE VIDEO SEGMENTS

2.1. Overview of the method

This section introduces a general framework that detects all pairs of near-duplicates efficiently from a long video stream (Length: n frames). The basic idea is to make the $O(n^2)$ times comparison as fast and accurate as possible by 1) comparing the features of short video fragments instead of a frame, and also by 2) compressing the dimension of the feature vectors, as shown in Fig. 1. The first approach makes the comparison efficient and at the same time robust to noise. Meanwhile, the second approach reduces computation time together with i/o time and storage space which is a significant problem when processing a long video stream.

2.2. Selecting video features for compression

In order to compress video features efficiently, it is necessary to select features more informative than others. In the proposed method, PCA (principal component analysis) is applied to feature vectors composed of raw pixel values obtained from each frame of a training video stream.

For preparation, each frame i extracted from an MPEG-1 video stream (Original size: 352×240 pixels) is first degraded to 22×15 pixels. Next, a feature vector $\boldsymbol{f}_i = \left[f_{i,1}, f_{i,2}, ..., f_{i,m_f}\right]^T$ is composed by arranging the RGB values $f_{i,j}$ of all the pixels as an array, which forms an $m_f = 990\ (= 22 \times 15 \times 3)$ dimension vector. In order to absorb color difference across different video sources, $\boldsymbol{f}_i$ is normalized to $\hat{\boldsymbol{f}}_i$ (hereafter, *frame vector*) as follows:

$$\mu_i = \frac{1}{m_f}\sum_{j=1}^{m_f} f_{i,j} \quad (1)$$

$$\bar{\boldsymbol{f}}_i = \left[f_{i,1}-\mu_i, f_{i,2}-\mu_i, ..., f_{i,m_f}-\mu_i\right]^T \quad (2)$$

$$\hat{\boldsymbol{f}}_i = \bar{\boldsymbol{f}}_i / \|\bar{\boldsymbol{f}}_i\| \quad (3)$$

As shown in Fig. 1, the compression is realized by selecting features by the following two steps:

1. Frame-wise feature selection
   Create a matrix $\boldsymbol{F} = \left[\hat{\boldsymbol{f}}_{i_1}\hat{\boldsymbol{f}}_{i_2}...\hat{\boldsymbol{f}}_{i_{K_f}}\right]$ by arranging $K_f(\geq m_f)$ frame vectors. Obtain the unit eigenvectors $\{\boldsymbol{e}_{f,1}, \boldsymbol{e}_{f,2}, ..., \boldsymbol{e}_{f,D_f}\}$ corresponding to the top $D_f(\leq m_f)$ large eigenvalues of the auto-correlation matrix $\boldsymbol{Q}_{\boldsymbol{F}} = \boldsymbol{F}\boldsymbol{F}^T$. The vector space spanned by the basis $\langle \boldsymbol{e}_{f,1}, \boldsymbol{e}_{f,2}, ..., \boldsymbol{e}_{f,D_f}\rangle$ is used as the frame-wise compressed feature space.
2. Time-wise feature selection
   For each frame, transform the frame vector $\hat{\boldsymbol{f}}_i$ on to the vector space $\langle \boldsymbol{e}_{f,1}, \boldsymbol{e}_{f,2}, ..., \boldsymbol{e}_{f,D_f}\rangle$ to obtain a frame-wise compressed vector $\boldsymbol{f}'_i$ as follows:

   $$\boldsymbol{f}'_i = \left[\boldsymbol{e}_{f,1}\boldsymbol{e}_{f,2}...\boldsymbol{e}_{f,D_f}\right]^T \hat{\boldsymbol{f}}_i \quad (4)$$

   Next, N_f adjoining compressed frame vectors starting from frame i are concatenated as a spatiotemporal feature vector $\hat{\boldsymbol{v}}_i$ with a dimension of $m_v = D_f N_f$:

   $$\hat{\boldsymbol{v}}_i = \left[\boldsymbol{f}'^T_i, \boldsymbol{f}'^T_{i+1}, ..., \boldsymbol{f}'^T_{i+N_f-1}\right]^T \quad (5)$$

   where N_f is the size of the video fragments. A matrix $\boldsymbol{V} = \left[\hat{\boldsymbol{v}}_{i_1}\hat{\boldsymbol{v}}_{i_2}...\hat{\boldsymbol{v}}_{i_{K_v}}\right]$ is created by arranging the $K_v(\geq m_v)$ spatiotemporal vectors. Again, obtain the unit eigenvectors $\{\boldsymbol{e}_{v,1}, \boldsymbol{e}_{v,2}, ..., \boldsymbol{e}_{v,D_v}\}$ corresponding to the top $D_v(\leq m_v)$ large eigenvalues of the auto-correlation matrix $\boldsymbol{Q}_{\boldsymbol{V}} = \boldsymbol{V}\boldsymbol{V}^T$. The vector space spanned by the basis $\langle \boldsymbol{e}_{v,1}, \boldsymbol{e}_{v,2}, ..., \boldsymbol{e}_{v,D_v}\rangle$ is used as the time-wise compressed feature space.

Thus is obtained the spatiotemporal compressed feature space that efficiently represents the video segments.

2.3. Detecting near-duplicate video segments

2.3.1. Extraction of spatiotemporal feature vectors

The same preparation as in the training phase described in Sect. 2.2 is applied to an input video stream. Likewise, a normalized input frame vector $\hat{\boldsymbol{f}}_i$ is frame-wise compressed by Eq. 4. It is then concatenated with the following $N_f - 1$ compressed frame vectors as in Eq. 5 $(= \hat{\boldsymbol{v}}_i)$ and then time-wise compressed by the following transformation:

$$\boldsymbol{v}'_i = \left[\boldsymbol{e}_{v,1}\boldsymbol{e}_{v,2}...\boldsymbol{e}_{v,D_v}\right]^T \hat{\boldsymbol{v}}_i \quad (6)$$

In this manner, compressed spatiotemporal feature vectors are created for all video fragments by shifting a window with a size of N_f frames, frame by frame.

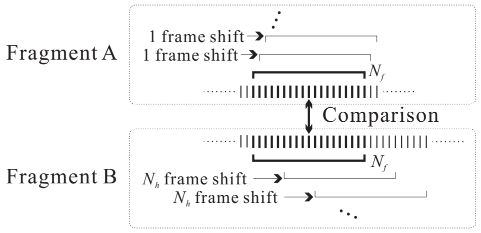

Fig. 2. Comparison between fragments. The reference fragment (A) is shifted frame by frame, while the fragment to be compared (B) is shifted by N_h frames at a time. In this case, segments longer than $N_{min} = N_f + N_h$ frames could be detected.

2.3.2. Step 1: Comparison of video fragments in the compressed feature space

Video fragments i_1, i_2 are compared by the L_2 distance d_1 between their spatiotemporal feature vectors as follows:

$$d_1(\boldsymbol{v}'_{i_1}, \boldsymbol{v}'_{i_2}) = \sqrt{\sum_{l=1}^{D_v}\left(v'_{i_1,l} - v'_{i_2,l}\right)^2} \quad (7)$$

where $v'_{i,l}$ represents the l-th component of a vector $\boldsymbol{v}'_i$. When d_1 is shorter then a threshold θ_1, the fragments are considered as near-duplicate candidates.

Theoretically, it is necessary to compare all fragments versus all fragments, which results in $\binom{n-N_f+1}{2}$ times of comparison for a video stream with a length of n frames. This may however, be reduced if we can restrict the minimum length of a near-duplicate video segment that should be detected. As shown in Fig. 2, if the minimum length is set to N_{min} frames long, it is possible to skip $N_h = N_f - N_{min}$ frames on one side of the comparison, which results in reducing the total times of comparison. As a matter of fact, when we consider general broadcast video streams as a target, the combination of N_f and N_h could be set to relatively high numbers depending on the application. For example, when detecting all commercials longer than $N_{min} = 900$ frames (30 seconds) and possibly some of those longer than 450 frames (15 seconds), $N_f = 450, N_h = 450$ may be a good combination. N_f and N_h may be any combinations as long as the constraints are met, but a long N_f will make the detection stable, and a long N_h will reduce the computation time.

2.3.3. Step 2: Confirmation of near-duplicate fragments in the original feature space

The comparison of video fragments in the low-dimension feature space derives numerous candidates of near-duplicate video fragment pairs. In order to filter out false positives among the candidates, the pairs are compared in the original (high-dimension) feature space by the following function:

$$d_2(\boldsymbol{v}_{i_1}, \boldsymbol{v}_{i_2}) = \sqrt{\sum_{l=1}^{m_v}\left(v_{i_1,l} - v_{i_2,l}\right)^2} \quad (8)$$

Table 1. Parameters used in the experiment.

K_f, K_v	15,000	$\theta_1 = \theta_2$	0.8
D_f	10 [dimensions]	N_f	150 [frames]
D_v	10 [dimensions]	N_h	150 [frames]

where $\boldsymbol{v}_i = \left[\boldsymbol{f}_i^T, \boldsymbol{f}_{i+1}^T, ..., \boldsymbol{f}_{i+N_f-1}^T\right]^T = [v_{i,1}, v_{i,2}, ..., v_{i,m_v}]^T$ and $m_v = m_f N_f$. When $d_2 \leq \theta_2$, the fragments are confirmed as near-duplicates.

Note that the distance d_1 in the compressed feature space is always shorter or equals to the distance d_2 in the original feature space, due to the characteristics of L_2 distance. This fact theoretically guarantees that if $\theta_1 = \theta_2$, the detection in the compressed feature space is equivalent to the detection on the original feature space for video fragments longer than N_f frames.

Since the cost for this process is comparable to the brute-force frame-by-frame comparison, the key of improving the overall efficiency is to suppress as much false positives as possible during the candidate detection process.

2.3.4. Post-processing

After detecting near-duplicate fragments, precise boundaries are obtained as a final result by adjusting them by frame-by-frame comparison at both ends of the fragments.

2.4. Experiment

The proposed method was applied to broadcast video data to measure the computation time on a Pentium IV 3.0GHz PC with 1.0GB of main memory. Parameters were set to the values shown in Tab. 1. The parameters ensure the detection of near-duplicate video segments at least 300 frames (10 seconds) long.

In order to obtain bases general enough to compress most broadcast television video efficiently, 150 hours of continuous video stream obtained from a Japanese channel during June 1–7, 2004 were used as training data. The sample frames/fragments were selected randomly.

The computation time measured by 1, 3, 6, 12, and 24 hours of general video streams with many near-duplicates (mostly commercials) is shown in Fig. 3. The computation time by brute-force frame-by-frame comparison was also measured with the same similarity threshold for comparison.

The result shows that the proposed method significantly reduces the computation time required for the detection. Although the process was more than 1,000 times faster, the result was equivalent to that of the brute-force frame-by-frame comparison for segments longer than N_{min} frames. From the result, we estimate that it should be possible to reduce the computation time required for a 1 week long video stream from 50 years to 5 days.

3. GENRE-ADAPTIVE PIXEL SELECTION

The method proposed in Sect. 2 was applicable to general video streams regardless of their contents. We have however, noticed that depending on the genre of the shows, there are regions in a frame that are mostly stable across frames, such as the gallery seats in a tennis match, or the super-imposed caption area in a news show. In other words, these regions are not appropriate to refer to when comparing the difference of frames within the same genre or show. Considering this, in this section, we introduce a scheme that selects highly dissimilar pixels across frames based on time-wise entropy, for better comparison of the feature vectors.

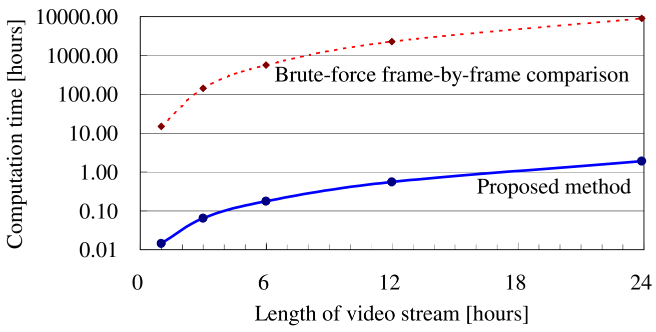

Fig. 3. Computation time for the detection.

3.1. Selecting significant pixels for comparison

The following schemes were applied to five shows in four genres of broadcast video programs.

- **Scheme 0:** No selection ($22 \times 15 \times 3 = 990$ dimensions)
  Obtain features from all the pixels in a frame. Identical to the method described in Sect. 2.
- **Scheme 1:** Fixed selection
  Obtain features from pixels in fixed locations.
  - **1h:** Horizontal line ($22 \times 3 = 66$ dimensions)
    Select pixels on a horizontal line in the center.
  - **1v:** Vertical line ($15 \times 3 = 45$ dimensions)
    Select pixels on a vertical line in the center.
- **Scheme 2:** Selection by time-wise entropy (D_s dimensions)
  In order to select pixels highly dissimilar across frames, we propose to evaluate the time-wise entropy E_j of the RGB values at each pixel.

$$E_j = -\sum_c \frac{H_j(c)}{N_s} \log_2 \frac{H_j(c)}{N_s} \qquad (9)$$

A certain number ($\lceil \frac{D_s}{3} \rceil$) of pixels are selected according to the entropy of the color histogram $H_j(c)$ at pixel j created across N_s frames.

The general method proposed in Sect. 2 also selects significant pixels during the process of calculating auto-correlation matrices. It however, considers only the overall amount of dissimilarity in a given period of time, but not the temporal changes of dissimilarity necessary to discriminate a frame to another one.

3.2. Experiment

The pixel selection schemes were applied separately to four genres and five shows as shown in Tab. 3.

The selected pixels were considered as the original feature vector $\boldsymbol{f}_i$, and the same procedure was applied as in the experiment in Sect. 2.4. Parameters were set to the values as shown in Tabs. 1 and 2, while the window size N_s for calculating the entropy was set to all frames in the entire training video stream for each genre. Note that the number of pixels D_s selected in Scheme 2 was set to 45, which

Table 2. Parameters used in the experiment.

N_f	30 [frames]	Histogram bins	64
N_h	30 [frames]	D_s	45

Table 3. Genres and corresponding data used in the experiment.

Genre	Broadcaster		Training data	Test data
Tennis	NHK	(Japan)	120 min.	60 min.[1]
Soccer	Fuji	(Japan)	90 min.	60 min.[1]
Baseball	Asahi	(Japan)	100 min.	100 min.[1]
News_JP	NHK	(Japan)	90 min.	50 min.
News_US	CNN	(U.S.A.)	120 min.	30 min.

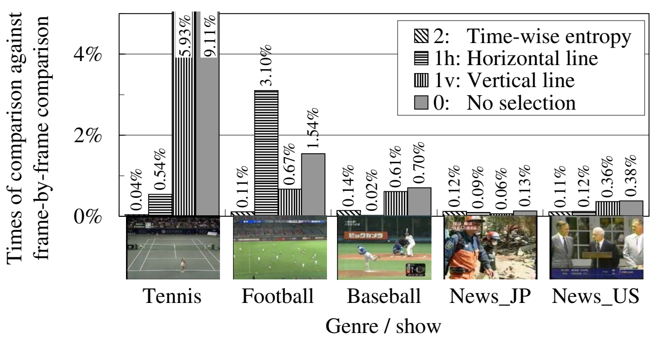

Fig. 4. Reduction of the times of comparison.

is the same as in Scheme 1v for fair comparison, and that the dimensions D_f, D_v of the compressed feature space were the same for all schemes.

Figure 4 shows the ratio of the numbers of fragment pairs to be confirmed in Step 2 against all combinations of frame-by-frame comparison by the brute-force comparison. We can observe how much times of comparison may be spared compared to the method proposed in Sect. 2 (corresponds to Scheme 0: No selection) by applying the pixel selection schemes. Note that the precision is always 100% when the brute-force frame-by-frame comparison is considered as the ground-truth, since the method applies process equivalent to that in Step 2. As for the recall, we confirmed that applying the pixel selection scheme does not affect the performance; 100% recall.

Though there are some cases that fixed selection (Scheme 1) outperforms others, selection by time-wise entropy (Scheme 2) always outperforms the general method (Scheme 0). Among the genres, pixel selection was especially effective when applied to those with mostly fixed angles, but not always for those with relatively random pictures. Thus, we conclude that Scheme 2 should be the best approach to automatically select significant pixels adaptive to shows or genres.

The efficiency of applying Scheme 2, however, differs among genres; it does not show significant improvement in genres such as news, since most of the frame pictures do not have a common structure. On the other hand, it is generally effective (ranging from $\frac{1}{227}$ to $\frac{1}{5}$) to genres related to sports with mostly fixed shots. This result suggests that the dimension D_s may also be set according to each genre, which we will investigate in the future.

4. CONCLUSION

We have proposed and evaluated an efficient and accurate method for detecting all near-duplicate video segments in a long video stream. Experiments showed that the method is more than 1,000 times faster than brute-force frame-by-frame comparison with equivalent accuracy, and also that additional improvements may be possible in some genres when significant pixels are selected for comparison.

In the future, we will implement efficient indexing methods, and evaluate the method with a longer video stream. Using the detected near-duplicates for higher-level applications, such as summarization and story tracking are also included in future works.

We will also work on developing applications that make use of the detected near-duplicate segments such as detecting same news events across different channels broadcast in different languages [8].

[1]Note that some short segments were artificially copied into the test data for Tennis, Soccer, and Baseball, due to the lack of genuine near-duplicates in the original video streams.

5. ACKNOWLEDGMENTS

Parts of this work were supported by the Grants-in-Aid for Scientific Research and the 21st century COE program from the Ministry of Education, Culture, Sports, Science and Technology, and the Japan Society for the Promotion of Science, and also by the Research Grant from Kayamori Foundation of Information Science Advancement.

6. REFERENCES

[1] Pinar Duygulu, Jia-Yu Pan, and David A. Forsyth, "Towards auto-documentary: Tracking the evolution of news stories," in *Proc. 12th ACM Intl. Conf. on Multimedia*, Oct. 2004, pp. 820–827.

[2] Yun Zhai and Mubarak Shah, "Tracking news stories across different sources," in *Proc. 13th ACM Intl. Conf. on Multimedia*, Nov. 2005, pp. 2–10.

[3] Dong-Qing Zhang and Shih-Fu Chang, "Detecting image near-duplicate by stochastic attributed relational graph matching with learning," in *Proc. 12th ACM Intl. Conf. on Multimedia*, Oct. 2004, pp. 877–884.

[4] Nobuhiro Sekimoto, Takuichi Nishimura, Hironobu Takahashi, and Ryuichi Oka, "Continuous retrieval of video using segmentation-free query," in *Proc. 15th Intl. Conf. on Pattern Recognition*, Sept. 2000, pp. 375–378.

[5] Xavier Naturel and Patrick Gros, "A fast shot matching strategy for detecting duplicate sequences in a television stream," in *Proc. 2nd Intl. Workshop on Computer Vision meets Databases*, June 2005, pp. 21–27.

[6] Fuminori Yamagishi, Shin'ichi Satoh, Takashi Hamada, and Masao Sakauchi, "Identical video segment detection for large-scale broadcast video archives," in *Proc. 3rd Intl. Workshop on Content-Based Multimedia Indexing*, Sept. 2003, pp. 135–141.

[7] Xianfeng Yang, Ping Xue, and Qi Tian, "A repeated video clip identification system," in *Proc. 13th ACM Intl. Conf. on Multimedia*, Nov. 2005, pp. 227–228.

[8] Ichiro Ide, Kazuhiro Noda, Akira Ogawa, Shin'ichi Satoh, and Hiroshi Murase, "Semantic analysis of a large-scale news video archive," in *Proc. Asia-Pacific Workshop on Visual Information Processing (VIP) 2006*, Nov. 2006, pp. 166–171.