Real-time Monitoring System for TV Commercials Using Video Features

Sung Hwan Lee, Won Young Yoo, and Young-Suk Yoon

Electronics and Telecommunications Research Institute (ETRI),
161 Gajeong-dong, Yuseong-gu, Deajeon, 305-350, Korea
{lovin, zero2, ys.yoon}@etri.re.kr

Abstract. For companies, TV commercial is a very important way to introduce and advertise their products. It is expensive to put an advertisement on TV. So these companies generally charge other companies to monitor that their TV commercials are broadcasted properly as contracted. Currently, these monitorings have been done manually. The monitoring company records all the TV programs and their air-times while they are being broadcasted. Then the agent checks the starting-times and the ending-times of TV commercials. Video retrieval and matching techniques can be used to monitor TV commercials automatically. By extracting visual features that can identify commercials, we can measure similarities and identify a target video from thousands of videos. To process all the TV programs of 24 hours a day, feature extraction and matching process must be done in real-time. In this paper, we designed the visual feature DB for real-time processing and implemented real-time TV commercial monitoring system. To construct the DB, we extracted scene change information, block-based dominant colors and edge pattern histograms of TV commercial clips.

1. Introduction

Many companies introduce their products on TV. It is very expensive to put a commercial on TV. So these companies generally charge other companies to verify that their TV commercials are actually broadcasted as contracted. Currently, monitoring-jobs are done by manually. All the TV programs are recorded with their air-time. Then the agent checks the start time and the length of the commercial. There are several tens of TV channels to monitor. It's a hard work to be done manually. However, using video identification techniques, we can automate the monitoring job. Most video identifications are done by feature extracting and matching it. There are many visual features that can be extracted from a video. The more features make it more accurate but requires more time. To make it useful, all the processes must be done in real-time.

In content based video retrieval and identification, the video sequence is often segmented into shots and the video content is described based on those shot content descriptors [1, 2, 3]. This shot based content representation is suitable for indexing and retrieval. After shot boundaries are located, the content of frames within the shot

can be modeled and efficiently matched against those in the database. Key frame color histogram is a widely used color content descriptor for shot. One or more frames are selected to represent the shot, and the selection criterion can be based on color or motion [1, 4]. However, when noise is added or frame rate changes, different frames may be chosen as key frames, which will lead to variation of this color descriptor. To reduce this variation, robust color histogram descriptors are proposed [5]. In this descriptor, the color histogram is computed from all the frames within the shot. The simple way is averaging color histograms of all the frames, and it works well with most cases. Although this color histogram descriptor is robust to rotation, scaling, translation and outlier frames, it is not robust to color distortion that lasts in a group of frames. For example, brightness and contrast of the whole video are significantly modified in a way that the visual quality does not change much. Since this type of content modification can be conveniently implemented in many video editing tools, content attackers can take advantage of this means to produce pirated copies which can avoid detection of video identification methods based on color. Therefore robustness to such type of color distortions should be really considered in content based video copy detection. To avoid the color distortion difficulty, local edge descriptor is proposed as key frame feature [6], but this feature is sensitive key frame shift. In key frame representation one shot maps to a point in a feature space, while other approaches represent the shot content by feature trajectory. In [7, 8, 9] the color feature of each frame is quantized into symbols, and the shot is represented by a compact symbol sequence. Because the sequence length can be variable under different frame rates, shot matching often employs symbol edit operations or dynamic programming which can be very complex when the number of frames is large. The quantization process can tolerate some range of color variation, but if the feature vectors of group of frames shift across the quantization boundaries, this string representation will totally change to another one whose similarity with the original one can change dramatically based on symbol sequence matching. We used the scene change information and the block-based dominant color [10] and the edge pattern histogram of a scene as visual feature.

2. System Components

Our system consists of three major components – feature extractor, feature DB and matching module in Fig. 1. The ‘feature extractor’ extracts video features such as scene change information, block-based dominant color and edge pattern histogram. All the extracted features are stored in the ‘feature DB’. The ‘matching module’ retrieves the feature DB and performs similarity measure.

We build the DB using about 250 TV commercials. Each commercial has a length of about 10 seconds. The DB is constructed with known commercials initially. While our system monitors the captured TV programs, a new commercial can be found. If an unidentified video clip that is located among two commercials and the length is shorter than 30 seconds, we consider it as a commercial. In most case, the commercials are located in series and the length is shorter than 30 seconds. The features of a new commercial are extracted and added to the feature DB.

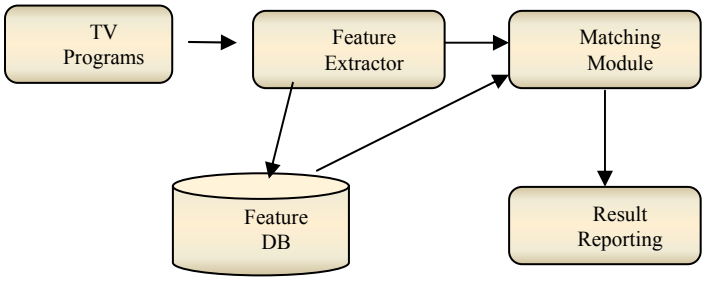


Fig. 1. System components

Our system works as follows. We build feature DB from already known TV commercials. At that time, feature extractor is used. Then, TV capture module captures all the TV programs and store the captured video clip. Feature extractor gets features from the captured clip and the matching module retrieves features from the feature DB. In matching module, each similarity functions of the features are used to find most similar commercial.

3 Feature Extraction

Comparing frames between two videos is a very simple and ignorant manner to find out whether they are absolutely the same or not. Although a video is semantically equal to another, the video can be slightly different from another because of all kinds of noise. Therefore, we should consider visual features to detect video data owing to both identifying the semantically same video and rising sharply a processing time in those schemes. There are many visual features that can be used to identify a video data. However, we cannot utilize a lot of features to detect exactly a video data owing to a waste of processing time. So, we chose three visual features such as scene change, edge, and dominant color.

Our approach progresses as follows. First, the scene change detection method reduces video clip into several key frames and produces scene length information. Next, the edge information for key frames obtains the shape of video data. Concurrently, we employ block-based dominant color in order to identify commercials with different caption because TV commercials have captions of broadcast company in many cases. Most captions are located in the corner of the video screen. So we applied block-based dominant color to identify video data with different caption despite the same video sequences. The proposed system works very well through simple and fast visual features.

3.1 Scene Length

Scene change is defined as a switch from a frame to a consecutive frame. Thus, scene change detection schemes are used to divide a video data into its elementary scenes. We can classify scene change detection methods in two groups such as compress or uncompress domain.

First, scene change detection scheme in uncompress domain can be divided by several methods as follows. While a pixel based method analyses the difference between pixel values over two adjacent frames, a block based method works by partitioning a frame into blocks and carrying out analysis on each of those blocks. Histogram representation uses that histogram has either colors or intensities grouped into bins and represented the frequencies of those values. Moreover, the edge change ratio involves calculating the number of new edge pixels entering the scene change and the number of old pixels leaving the scene change.

Next, scene change detection algorithms in compress domain can use the types of frames, DCT coefficients, and motion vectors. And an unsupervised clustering algorithm defines shot detection as a clustering problem where are clustered into 2 classes, a shot boundary class and non-shot boundary class.

Scene Change Detection Scheme In a general pixel based method, the difference value at every pixel position within a frame is added and the result normalized, giving a difference which can be compared against a threshold to determine where this is a boundary frame. Eq. (1) defines $\Delta I_{x,y}(t)$ as the absolute value of pixel difference with Δt frame distance

$$\Delta I_{x,y}(t) = \left| I_{x,y}(t+\Delta t) - I_{x,y}(t) \right| \tag{1}$$

where $I_{x,y}(t)$ denotes intensity value of pixel located at (x,y) when time t. If we set $\Delta t = 1$, we could get the absolute value of consecutive frame difference.

We obtain the absolute sum of the inter-frame difference $Sum_{|FD|}(t)$ using Eq. (1). This is a simplest way of evaluating an image difference.

$$Sum_{|FD|}(t) = \sum_{x} \sum_{y} \left| \Delta I_{x,y}(t) \right| \tag{2}$$

Unfortunately, it is difficult to distinguish between a large change in a small area and a small change in a large area when we use the sum of frame difference value. So, we tend to be unable to detect cuts when a small part of a frame undergoes a large, rapid change.

To improve detection performance, we proposed $Rate_{|FD|}(t)$ meaning the number of pixels into the number of absolute values of inter-frame difference larger than a given threshold. Eq (3) denotes $Rate_{|FD|}(t)$,

$$Rate_{|FD|}(t) = \frac{n(\{\Delta I_{x,y}(t) \mid 2^{b-4} < \Delta I_{x,y}(t)\})}{x_{\max} \times y_{\max}} \tag{3}$$

where, the function *n* represents the number of set, *b* is defined as the used number of bits to represent the intensity of pixels. If we use 8 bits on the intensity channel, 2^{b-4} should be 16 in which 2^{b-4} is the empirical threshold. We can easily detect scene changes due to non-linear relationship between the proposed $Rate_{|FD|}(t)$ and $Sum_{|FD|}(t)$.

Furthermore, we designed a scene change detection filter so as to detect scene change in spite of frame rate conversion, slow motion, abrupt flash. The proposed method feeds $Rate_{|FD|}(t)$ the scene change detection filter composed of local maximum filter and minimum filter. Eq. (4) and Eq. (5) represent filters computing maximum and minimum value within the local range, respectively.

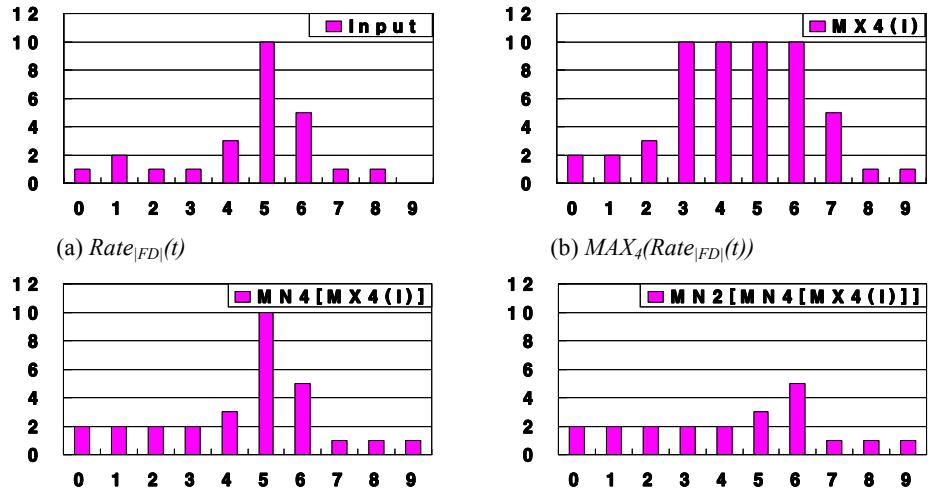
$$MAX_{range}(t) = \max(Rate_{|FD|}(t+i)) \quad (-\frac{range}{2} \le i \le \frac{range}{2} - 1) \tag{4}$$

$$MIN_{range}(t) = \min(Rate_{|FD|}(t+j)) \quad (-\frac{range}{2} + 1 \le i \le \frac{range}{2}) \tag{5}$$

And then, we define the scene change detection filter as Eq. (6) using Eq. (4) and Eq. (5)

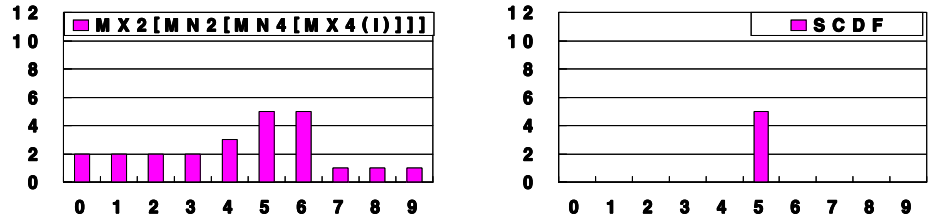
$$SCDF(t) = MIN_4(MAX_4(t)) - MAX_2(MIN_2(MIN_4(MAX_4(t)))) \tag{6}$$

Figure 2 shows a simple process of proposed scene change detection filter. Fig. 2 (a) is a simple example in order to simulate those processes. The proposed scheme obtains the output of scene change detecting filter. It is getting on to zero but has value lager than 0.2 when a frame is scene change in general.



(a) $Rate_{|FD|}(t)$ (b) $MAX_4(Rate_{|FD|}(t))$

(c) $MIN_4(MAX_4(Rate_{|FD|}(t)))$ (d) $MIN_2(MIN_4(MAX_4(Rate_{|FD|}(t))))$



(e) $MAX_2(MIN_2(MIN_4(MAX_4(Rate_{|FD|}(t)))))$ (f) $SCDF(t)$

Fig. 2. Scene change detecting filter

Property of Scene Change As we mentioned before, we make a good use of a detected scene change as a visual feature. The found scene changes offer largely two facts. First, the scene change represents key frame at a scene. Thus, we do not need to consider other frames to process in real-time at the scene. Next, the scene length computed from detected scene changes obtains a specific character of video data. Furthermore, we can overcome the synchronization problem detecting where the start position of stored video set is in quarry video.

3.2 Histogram for Edge Pattern

Edges can be used as a shape descriptor of an object in video data. There are many methods using edge to identify an object. However, we need only simple feature with edge characteristics to process a video identification system in real-time. Consequently, we used a histogram of main directions for the edges to describe the object. We classified the directions of edges with six patterns such as vertical edge, horizontal edge, diagonal_45 edge, diagonal_135 edge, no directional edge, and no edge. A frame occurred by the scene change is divided into small blocks and then we can compute the histogram of edge pattern for blocks with Eq. (7). Finally, we obtain edge directions using edge filters from Fig. 3.

1	-1
1	-1

a)vertical

1	1
-1	-1

b)horizontal

$\sqrt{2}$	0
0	$-\sqrt{2}$

c)diagonal_45

0	$\sqrt{2}$
$-\sqrt{2}$	0

d)diagonal_135

2	-2
-2	2

e)no directional

Fig. 3. Edge filters

$$
\begin{aligned}
ver\_edge\_count(i,j) &= \left| \sum_{k=0}^{3} A_k(i,j) * ver\_edge(k) \right| \\
hor\_edge\_count(i,j) &= \left| \sum_{k=0}^{3} A_k(i,j) * hor\_edge(k) \right| \\
dia45\_edge\_count(i,j) &= \left| \sum_{k=0}^{3} A_k(i,j) * dia45\_edge(k) \right| \\
dia135\_edge\_count(i,j) &= \left| \sum_{k=0}^{3} A_k(i,j) * dia135\_edge(k) \right| \\
nond\_edge\_count(i,j) &= \left| \sum_{k=0}^{3} A_k(i,j) * nond\_edge(k) \right|
\end{aligned} \tag{7}
$$

We determine edge pattern by the maximum number of same direction. After calculating edge patterns, histogram of edge patterns are constructed at the key frame.

3.3 Block-based Dominant Color

Dominant color is defined in MPEG-7 standard [10]. This color descriptor is most suitable for representing local (object or image region) features where a small number of colors are enough to characterize the color information in the region of interest. Whole images are also applicable, for example, flag images or color trademark images.

Color quantization is used to extract a small number of representing colors in each region/image. The percentage of each quantized color in the region is calculated correspondingly. A spatial coherency on the entire descriptor is also defined, and is used in similarity retrieval.

```
1) Assign each color to its cluster using Euclidean
   distance
2) Calculate cluster centroid.
3) Calculate the total distortion and split colorBins
   to increase CurrentBinNum until CurrentBinNum = Fi-
   nalBinNum.
4) Merge colorBins using a clustering method.
5) Repeat to 1) until process all pixels of a block.
```

TV commercials are captured with the captions of the broadcast company. It is required to identify a commercial with different logos. Figure 4 shows the logo area of TV commercials. Most logos are located at the corners of video screen. The logo-blocks are removed while we extract dominant color.

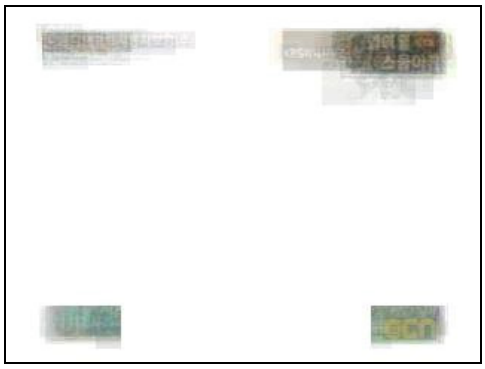


Fig. 4. Caption locations of captured TV commercials

4. Matching Method

The captured TV programs are processed by the feature extractor to produce features. The extracted features are passed to the matching module. Matching module retrieves candidates using scene length information using Eq. (8).

$$\begin{aligned} &C_i = ID(V_j)\ if\ \sum\left|(SL_{q,k} - SL_{d,k})\right| < T \\ &\quad where\ ID(V)\ is\ ID\ of\ a\ video\ V, \\ &\quad SL_{q,k}\ is\ k\text{-}th\ shot\ length\ of\ query, \\ &\quad SL_{d,k}\ is\ k\text{-}th\ shot\ length\ of\ database \end{aligned} \tag{8}$$

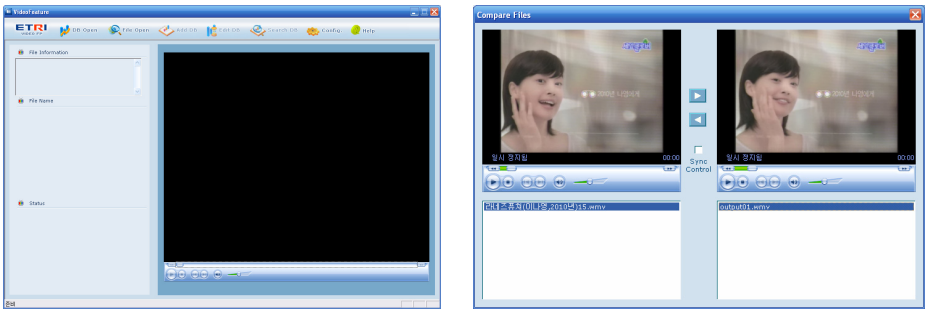
Using threshold T, we generate the candidates which have similar shot length. The final similarity function is Eq. (9).

$$\begin{aligned} &S(C_i) = w1 * \sum\left|DC(C_i) - DC(q)\right| + w2 * \sum\left|EH(C_i) - EH(q)\right| \\ &\quad where\ w1, w2\ is\ weight\ constant \end{aligned} \tag{9}$$

We combined the summation of the differences of the shot length with weight value *w1* and the summation of the differences of the dominant colors with *w2*.

5. Experimental Results

Figure 5 shows the implementation of our system. Fig. 5 (a) is main application that can manage the feature DB, identify captured video and add a new clip to the DB. Fig. 5 (b) is a result window that can compare the query video and the result.



a) Main application b) Result window

Fig. 5. Implementation result

Figure 6 shows feature DB management tool. With this tool, we can search, add, update and remove feature DB records.

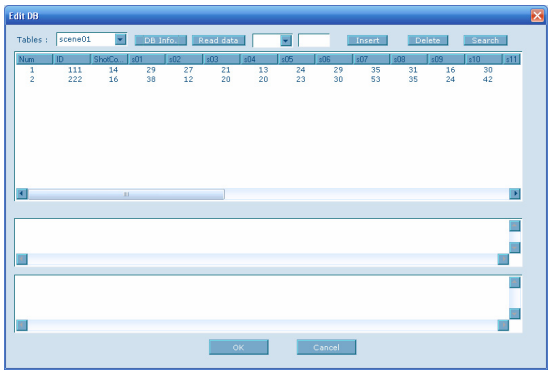


Fig. 6. Feature DB management tool

We used a computer with performance of P-4 3.0GHz CPU and 1GB memory. We constructed feature DB with recent 260 commercials in common database management system (DBMS). In order to check the performance of proposed system, we fed our TV commercial monitoring system into 364 minutes query video composed of 91 query video clips of 4 minutes length. Table 1 shows the experimental result. Our system requires about a half of playing time in order to identify video data. It's fast enough to process in real time. The matching rate is approximately 90%.

Table 1. Query result

Length of Query video	Success	False Positive	False Negative	Processing Time
364 minutes	87.51%	9.27%	3.22%	175 minutes

6. Conclusion

TV commercials are effective and expensive way to introduce a product. Currently, TV commercials are monitored to verify the contract manually. We designed and implemented the TV commercial monitoring system. We employed shot detection method, histogram of edge patterns, and block-based dominant color to identify a video. By combining three features, our system can identify 90% commercials in real-time. Our system can be improved with other features. So we are still in progress to develop features that can be processed in real-time.

References

1. Y. Rui, T. S. Huang, and S. Mehrotra, Exploring video structure beyond the shots, Proc. of IEEE Conf. on Multimedia Computing and Systems, June 1998.
2. H. J. Zhang, J.Wu, D. Zhong, and S.W. Somaliar, An integrated system for content-based video retrieval and browsing, Pattern Recognition 30 (1997), no. 4, 643–658.
3. J.M. Snchez, X. Binefa, and J. Vitri, Shot partitioning based recognition of tv commercials, Multimedia Tools and Applications 18 (2002), 233–247.
4. W. Wolf, Key frame selection by motion analysis, Proc. ICASSP, vol. II, 1996, pp. 1228–1231.
5. A.Mfit Ferman, A.MuratTekalp, and Rajiv Mehrotra, Robust color histogram descriptors for video segment retrieval and identification, IEEE Trans. on Image Processing 11 (2002), no. 5, 497–508.
6. Arun Hampapur and Ruud Bolle, Comparison of distance measures for video copy detection, International Conference on Multimedia and Expo, 2001.
7. D. A. Adjeroh and I. King M.C. Lee, A distance measure for video sequence, Computer Vision and Image Understanding 75 (1999), no. 1/2, 25–45.
8. Liping Chen and Tat-Seng Chua, A match and tiling approach to content-based image retrieval, IEEE Intl Conference on Multimedia and Expo (Tokyo, Japan), August 2001, pp. 417–420.
9. Xianfeng Yang, Qi Tian, and Sheng Gao, Video clip representation and recognition using composite shot models, ICICS-PCM, 2003.
10. MPEG-7, ISO/IEC TR 15938-8:2002, Information technology -- Multimedia content description interface -- Part 8: Extraction and use of MPEG-7 descriptions