VIDEO SCENE SEGMENTATION USING A NOVEL BOUNDARY EVALUATION CRITERION AND DYNAMIC PROGRAMMING

Bo Han, Weiguo Wu
Sony China Research Laboratory, Beijing 100190, China
{bo.han, weiguo.wu}@sony.com.cn

ABSTRACT

Video scene segmentation is a fundamental step for video summarization and browsing, which is a very promising application of multimedia analysis. There are two key elements, namely, boundary evaluation and boundary searching, in a scene segmentation algorithm. In this paper, we propose a novel boundary evaluation criterion, including the multiple normalized min-max cut scores, which consider not only neighboring but non-neighboring scene similarities with a memory-fading model, and the maximal cross-boundary strict shot similarity, which considers both color and structure similarities. Dynamic programming with a heuristic search scheme is adopted to quickly find the global optimal scene boundary sequence. Moreover, a Monte Carlo method is adopted to improve the stability of the searching process. Experimental results on a dataset of 40 diversified videos have proven the algorithm efficient, robust, and superior to the existent methods.

Index Terms—Scene segmentation, normalized min-max cut, LDA, dynamic programming, heuristic search, Monte Carlo, video summarization, video browsing

1. INTRODUCTION

With the fast growing of amount and accessibility of video contents, video summarization and browsing techniques are becoming more and more important in real applications. A video scene consists of a group of semantically closely-related consecutive shots. Scene is the essential video content unit and the most effective clue for fast video content understanding. For both of the usual video abstraction forms, i.e., key frame sequence and video skim, the main scenes of the video should be segmented and well-represented in the results. Hence, video scene segmentation, or temporal shot grouping, is a fundamental step for video summarization and browsing.

There have been many research efforts aiming for automatic video scene segmentation. The MCMC framework is introduced in [1]. However, because of some strong assumptions, parameters in the algorithm need to be tuned for different video types. An SVM-based sliding window boundary detection scheme is proposed in [2]. But the paper only exhibited experimental results on simulated data samples. Audio classification and speaker identification information is adopted in [3] for segmenting documentary films. Visual, audio, and textual clues are integrated in the framework of [4] to detect program changes in TV broadcast streams. Such approaches [3] [4] are only suitable for some specific tasks. In [5], the complex multi-cut problem of scene segmentation is simplified using a recursive shot similarity graph bi-partitioning method. Ref. [6] proposed a shot clustering and sequence alignment scheme, and reported better performance than [5]. For performance comparison, the latter two scene segmentation algorithms [5] [6] are implemented and tested in this paper.

A scene segmentation algorithm has two key elements, namely, boundary evaluation and boundary searching. The evaluation part tells how likely an assumed scene boundary should be a real one from the pattern classification viewpoint. The searching part investigates all the possible boundary combinations in a certain order and selects the best scene boundary sequence as the result. In this paper, a novel scene evaluation criterion is proposed. Each possible scene boundary is evaluated using both the multiple normalized min-max cut scores, which consider not only neighboring but non-neighboring scene similarities with a memory-fading model, and the maximal cross-boundary strict shot similarity, which considers both color and structure similarities. Based on the scores and the similarity, an LDA classifier, which is obtained in a supervised learning manner, judges whether the scene boundary is a real one. Dynamic programming with a heuristic search scheme is adopted for fast global optimal boundary searching. Moreover, a Monte Carlo method is introduced to improve the stability of the searching process, i.e., dynamic programming is executed many times with random boundary evaluation thresholds, and the final boundaries are determined via voting.

Comprehensive experiments were carried out on a dataset of 40 diversified videos. The 17-hour dataset contains about 660 scenes. As far as we know, this is the most complex scene segmentation dataset in the literature. In the experiments, the proposed algorithm proved its effectiveness and outperformed the existent methods [5] [6].

978-1-61284-350-6/11/$26.00 ©2011 IEEE

Fig. 1. Part of a scene from the movie Spider Man II. Each shot is represented by a frame in it. The consecutive shots are numbered according to their temporal order.

The rest of this paper is organized as follows. The scene boundary evaluation criterion is proposed in Section 2. The algorithm for finding all the boundaries is detailed in Section 3. Section 4 demonstrates experimental results of the different algorithms. Section 5 concludes the paper.

2. SCENE BOUNDARY EVALUATION

Generally, video scene segmentation is based on the shot change detection results. The schemes in [7] and [8] are integrated in this work as the shot change detector. Every detected shot boundary in the video is considered as a possible scene boundary, and need to be evaluated as a candidate. The novel evaluation criterion will be introduced in the sub-sections below.

2.1. Multiple Normalized Min-max Cut Scores

The basic principle of scene boundary evaluation is that the content should be quite similar inside a scene, and should have some obvious changes across a boundary.

The graph partition model [9], which regards the shots in a video as connected vertices in a graph, transforms video segmentation problem into partitioning the graph into isolated parts. Although graph partition model has been adopted in [5], intra-scene similarity was not well exploited in the recursive bi-partitioning scheme. The min-max cut [10] is a graph partition criterion that characterizes intra-scene similarity and inter-scene dissimilarity at the same time. However, it tends to partition the video into equal-length scenes by its design. To solve the aforementioned problems, we propose multiple normalized min-max cut scores for scene boundary evaluation.

Figure 1 shows some consecutive shots from one same scene in the movie Spider Man II. We can observe that, although the shots 12 – 14 (spider man thrown out through the roof and then fly back soon) are just a very short part of the whole scene, they share similar visual features and are quite dissimilar from other shots.

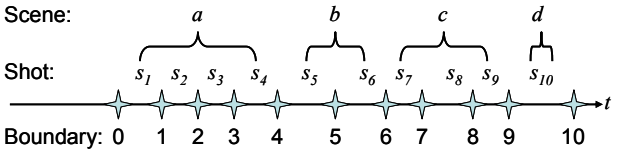

Fig. 2. A simple video segmentation example.

Considering only the closest neighboring scenes on both sides of the assumed boundary will result in over-segmentation (before 12 and after 14 in Figure 1). Since the scene boundaries separate all the shots that locate on different sides of them, the relationships between shots in non-neighboring scenes (1-11 and 15-20) are also taken into account to avoid such over-segmentations.

As is shown in Figure 2, suppose that a video has 4 consecutive scenes *a*, *b*, *c*, and *d*, of which each consists of at least one shot. Then evaluation of the boundary 6 (between *b* and *c*) is based on the multiple graph cut scores,

$$\vec{E}(b,c)=[C(b,c),C(a,c),C(b,d),C(a,d)] \tag{1}$$

where the normalized min-max cut $C()$ is defined as,

$$C(a,d)=\left(\frac{A(a,d)}{A(a,a)}+\frac{A(a,d)}{A(d,d)}\right)\cdot\exp[-v\cdot D(a,d)] \tag{2}$$

where constant v is the memory fading speed parameter. The function $D()$ is defined as the distance between the last frame of the temporally former segment and the first frame of the latter one. The length weighted association function $A()$ is defined as,

$$A(a,d)=\frac{\sum_{s_i\in a}\sum_{s_j\in d}S_C(s_i,s_j)\cdot L(s_i)\cdot L(s_j)}{\sum_{s_i\in a}\sum_{s_j\in d}L(s_i)\cdot L(s_j)} \tag{3}$$

where s_i and s_j represent the ith and jth shot in the video, respectively. $S_C()$ is the shot color similarity function. The function $L()$ means the length of the segment. Since longer shots occupy larger portion in the scene than shorter ones, they should play a more important role in the computation of scene association. So, the association definition in (3) is a shot-length-weighted mean of the shot similarities. Note that it is normalized compared to that in min-max cut [10], which is positively correlated to the shot number in each scene and makes equal-length scenes favored. $S_C()$ is defined as,

$$S_C(s_i,s_j)=I\left\{\underset{f_m\in s_i}{\text{mean}}[H_C(f_m)],\underset{f_n\in s_j}{\text{mean}}[H_C(f_n)]\right\} \tag{4}$$

where f_m and f_n are representative frames of s_i and s_j, respectively. $H_C()$ means the 64-bin HSV (2 bits for each channel) histogram of the frame. The function $I()$ means histogram intersection.

For two shots that are far apart in the video, if the shots in-between are quite different from them, then it is natural to separate them into different scenes, no matter how similar

they are. Hence, the memory fading model is introduced in (2) so that the relationship (edge weights in the graph) between two assumed scenes becomes weaker as their temporal distance becomes larger. In our system, the memory fading parameter v is set to be 0.05 (1/second) through all experiments. In practice, since the total length of 3 consecutive scenes must be relatively large, the graph cut in (2) is almost zero if the two scenes are more than 3 boundaries apart. Therefore, at most 6 scenes around the boundary need to be considered in (1), and considering 4 neighboring scenes is also acceptable in general applications.

2.2. Maximal Cross-boundary Strict Shot Similarity

In Figure 1, the color similarities among the shots are depicted in (4) using the global color histogram. It can be observed that some of them, such as 1, 16 and 20, share almost identical visual features. Besides (4), their similarity should contribute more in the boundary evaluation criterion, so that they are more likely to be grouped into one same scene. Thus, we propose the strict shot similarity, which should be large if and only if the two shots are very similar both in color and in structure, as,

$$S_S(s_i, s_j) = \min_{f_m \in s_i, f_n \in s_j} S_S^F(f_m, f_n) \quad (5)$$

where the strict shot similarity $S_S()$ is the minimum of strict similarities between their representative frames, which is defined as,

$$S_S^F(f_m, f_n) = \underset{r \in regions}{\text{mean}} \left\{ I\left[H_S^r(f_m), H_S^r(f_n)\right] \cdot I\left[H_C^r(f_m), H_C^r(f_n)\right] \right\} \quad (6)$$

where each of the two frames is divided in to 4 * 4 regions. Every pair of corresponding regions in the two frames is compared. $H^r_C()$ means the color histogram, as that in (4), of region r, $H^r_S()$ means the R-HOG feature [11] of region r. Each frame is down-sampled to a size between 80 * 60 and 160 * 120 before R-HOG feature extraction. It is obvious that a region pair will not contribute to the strict frame similarity if they are dissimilar either in color or in texture.

To make the similarity measure more strict, and to emphasize the structure similarity, (5) is further tuned as,

$$S_S'(s_i, s_j) = \left[S_S(s_i, s_j)\right]^2 / S_C(s_i, s_j) \quad (7)$$

Finally, the maximal cross-boundary strict shot similarity for boundary 6 in Figure 2 is defined as,

$$E'(b,c) = \max_{s_i \le b, s_j \ge c} S_S'(s_i, s_j) \cdot \exp\left[-v \cdot D(s_i, s_j)\right] \quad (8)$$

where the constant v and function $D()$ are defined as in (2).

2.3. Forming Evaluation Criterion

In order to convert the multiple scores and the maximal similarity to an evaluation value for classification, they are concatenated into a feature vector, and a supervised learning is performed to find the projection $\boldsymbol{p}$ that best distinguishes the real scene boundaries from the false ones. Then, the simplest way of judging whether to accept an assumed boundary or not is to compare (9) with a fixed threshold.

$$E(b,c) = \left[\vec{E}(b,c), E'(b,c)\right] \cdot \vec{p} \quad (9)$$

The projection $\boldsymbol{p}$ is obtained via Linear Discriminant Analysis (LDA), which gives the best linear classification under Fisher criterion, as,

$$\vec{p} = (\Sigma_+ + \Sigma_-)^{-1} * (\vec{\mu}_- - \vec{\mu}_+) \quad (10)$$

where μ_+ and Σ_+ represent the mean and covariance of the feature vectors of positive samples, respectively; the counterpart two symbols represent those of the negative samples. Positive samples are generated using the scene boundary ground truth. Negative samples are simulated using randomly selected shot boundaries which are not real scene boundaries.

Note that some non-linear classifiers, such as Support Vector Machine (SVM), may take the place of LDA to achieve a better projection $\boldsymbol{p}$, if enough (for example, thousands of) positive samples are available for training.

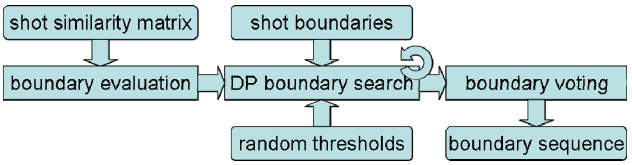

Fig. 3. The flowchart of boundary sequence searching.

3. BOUNDARY SEQUENCE SEARCHING

It can be seen from the definitions in Section 2 that the evaluation of one scene boundary is dependent on the positions of its neighbors. Hence, the proposed boundary sequence searching algorithm aims at finding the global optimum. The shot number in a scene varies greatly (from only 1 to over 100) in different cases. Compared with the fixed-length sliding window searching schemes [2] [6], which find every boundary with neighboring boundary positions unknown, the proposed algorithm is more robust to scene length variances.

The flowchart of our algorithm is shown in Figure 3. Dynamic programming (DP) with a heuristic search scheme is proposed to quickly find the optimum in the huge searching space of possible boundary sequences, of which each consists of some shot boundaries in the video. During DP, each boundary case is evaluated using the LDA projection in (9) based on the shot similarity matrix. The DP is carried out for multiple times with random boundary evaluation thresholds. After each round, every boundary in the result sequence obtains a vote. The final result is determined by the vote numbers of the boundary candidates.

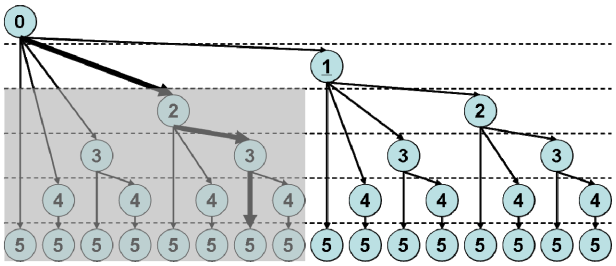

Fig. 4. A simple DP searching tree example.

3.1. Dynamic Programming

Suppose the total shot number in a video is n (A common Hollywood movie has about 2000 shots.) then the total number of possible scene boundary combinations is 2^{n-1}. To find the global optimal boundary sequence in the huge searching space is an NP-hard problem. This is why DP with a heuristic search scheme is proposed here.

DP breaks the problem of finding the optimal sequence into recursive simpler searching steps. Since the boundary evaluation is dependent on its neighbors, at each step, one possible segment of the boundary sequence, instead of one single boundary candidate, is investigated.

The simple searching tree of a 5-shot video is shown in figure 4. Each possible boundary sequence must start with boundary 0 and end with 5. There are totally 16 possible boundary sequences in the searching space. For example, a segment investigated in one DP step may be (2, 3, 5), and one of its possible preceding segments is (0, 2, 3). These two segments form a solution (0, 2, 3, 5) which divides the video into 3 segments (thick arrows in Figure 4) at boundary 2 and 3.

The basic philosophy of the searching process is that, we hope to find as many as possible real boundaries; and in the case of same boundary number, we hope to find the most probable boundaries.

The aim of each DP step is to find one best preceding segment. Denote a boundary sequence segment as (b_p, b_c, b_n). A step of the proposed DP algorithm is described as follows. The whole process is performed via ascending looping for b_n, b_c, and b_p.

1) Find one/another valid preceding segment candidate (b_f, b_p, b_c). If successful, go to 2); otherwise, go to 5).
2) Compute the evaluation value as in (9) for b_p in the boundary sequence. If the value is smaller than the threshold, go to 3); otherwise, go to 1).
3) If (b_p, b_c, b_n) currently has no preceding segment, record (b_f, b_p, b_c) as its preceding segment, then go to 1); otherwise, go to 4).
4) Compare (b_f, b_p, b_c) with the current preceding segment of (b_p, b_c, b_n). If (b_f, b_p, b_c) is "better", replace the current preceding segment. Go to 1).
5) If (b_p, b_c, b_n) has no preceding segment, set it as invalid; otherwise, set it as valid. Step finished.

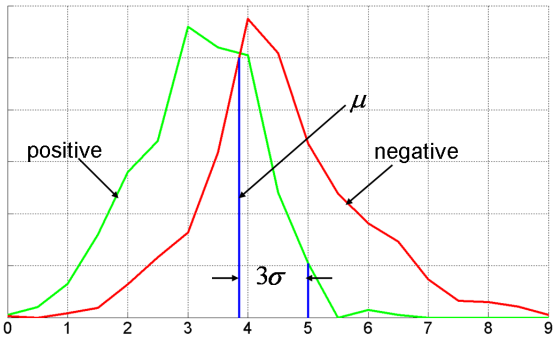

Fig. 5. Parameter selection for random thresholds.

According to the aforementioned philosophy, the term "better" in 4) means larger preceding boundary number and smaller average of preceding evaluation values.

After the whole DP process, the best (b_p, b_c, b_n) with b_n being the end boundary of the video can be found. Then, the optimal boundary sequence can be obtained via recursively retrieving the preceding segments.

To further decrease the computational burden, a heuristic search scheme is proposed in our algorithm. For every b_n, a temporary optimal boundary sequence is found via comparing all valid segment candidates. For each possible boundary, the number of times that it is included in the temporary optimal sequences is accumulated during the DP process. When the number becomes larger than a fixed threshold, such as 20 or 50, the corresponding boundary is heuristically considered as a real boundary. All the sequences that skip this boundary will be considered invalid in the following DP steps. For example, if boundary 1 in Figure 4 is heuristically approved, then all the 8 paths in the left part of the searching tree will be ignored in the subsequent process.

3.2. Monte Carlo Method

As is stated in Section 2, the aim of LDA is to find the best projection to separate real boundaries from false ones. The normalized histograms of the evaluation values of positive (green) and negative (red) samples are shown in Figure 5. We can observe that even after the projection, the positive samples can not be perfectly distinguished with any fixed threshold.

Moreover, since the boundary evaluation is dependent on neighboring boundaries, one classification error, which is caused by the threshold problem, may introduce a series of errors. That is to say, a small change in the threshold may introduce a distinct change in the result. Therefore, a Monte Carlo method is employed here to make the searching process more stable.

For each DP step, a random threshold is used for boundary evaluation in 2). The random thresholds are generated from a Normal distribution $N(\mu, \sigma)$. The expectation is decided as in (11), where the symbols have

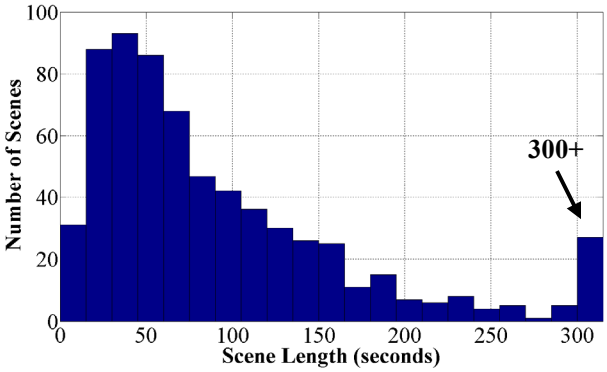

Fig. 6. Histogram of scene length in the dataset.

the same meaning as in (10). And the standard deviation is decided as is shown in Figure 5, so that every real boundary has the chance of being detected.

$$\mu = (\bar{\mu}_+ + \bar{\mu}_-) \cdot \bar{p}/2 \tag{11}$$

As is shown in Figure 3, the DP process is performed for many rounds. After each round, one optimal boundary sequence is obtained. Each boundary in the sequence gains one vote. Finally, the boundaries with enough votes will be considered as real scene boundaries in the result. In experiments, we found that 10 to 20 rounds of DP have a good balance between stability and speed.

Besides stability, the Monte Carlo method brings another benefit. Some specific applications may prefer over-segmentation or under-segmentation. The boundary voting data may be re-processed to generate more or less scene boundaries as needed, without re-running the program.

4. EXPERIMENTAL RESULTS

Experiments were carried out on a dataset of 40 videos, including 4 movies (*Spider Man II, The Legend of Zorro, Hancock,* and *The Pursuit of Happiness*), 7 TV dramas (in Chinese, Japanese, and Korean), 3 sitcoms, 6 DV works, 4 interviews, and various kinds of TV programs and non-professional videos. Most of the data were downloaded from different video websites. The 17-hour dataset contains about 660 scenes. This dataset is very complex and we believe that such a tough test is quite real-application-like.

4.1. Performance Evaluation

To properly evaluate video scene segmentation algorithms, both the F_1 measure, which is defined in (12), and coverage/overflow [12] are adopted.

$$F_1 = 2 \cdot recall \cdot precision / (recall + precision) \tag{12}$$

In [5] and [6], a temporal boundary error tolerance of 30 seconds is used when calculating recall and precision. That is to say, a detected boundary which is less than 30 seconds apart from a real boundary can be regarded as a true positive. While in our dataset, we found that more than 40% of the scenes are within 60 seconds (See Figure 6.) Thus, an error of 30 seconds is not so tolerable. A stricter tolerance of 20 seconds is adopted in our experiments. Note that the F_1 measure will decrease by more than 10% as the tolerance decreases from 30s to 20s, as is also shown in [5].

Table 1. Overall experimental results.

Method	F_1	Coverage	Overflow
[5]	*53.6%*	*46.6%*	*21.1%*
[6]	*63.7%*	*74.1%*	*27.5%*
Ours	*66.8%*	*75.1%*	*27.2%*

Coverage and overflow of a real scene are defined in (13) and (14), respectively. The overflow definition is modified from that in [12], so that it is normalized to 1. Note that smaller overflow indicates better performance.

$$CR(x_i) = \max_j \left[L(x_i \cap y_j) \right] / L(x_i) \tag{13}$$

$$OF(x_i) = 1 - L(x_i) / \sum_{j, x_i \cap y_j \neq \phi} L(y_j) \tag{14}$$

Here x_i is a scene in the ground truth, and y_j is a scene in the segmentation result. $L()$ is the length function as defined in (3). Coverage/overflow of a video is defined as the length-weighted average.

$$CR(video) = \sum_{x_i \in video} CR(x_i) \cdot L(x_i) / L(video) \tag{15}$$

$$OF(video) = \sum_{x_i \in video} OF(x_i) \cdot L(x_i) / L(video) \tag{16}$$

The overall performance on the video dataset is calculated as the weighted average of those on all the videos. Note that F_1 is weighted by the scene boundary number in the video; while coverage and overflow are weighted by the length of the video.

The advantage of F_1 is its consideration of both recall and precision on each video. The advantage of coverage/overflow is its independence of any tolerance threshold setting.

4.2. Algorithm Performance

Our algorithm was evaluated in the manner of 5-fold cross-validation. The dataset is divided into 5 sub-sets. Each sub-set is tested with the other four being used for training. Besides the proposed algorithm, we also implemented those in [5] and [6] for comparison. Since these two methods and their parameters are quite detailed, the implementations can be considered same as those in the papers.

The overall results on the dataset are exhibited in Table 1. Comparing the top two rows, we can see that the shot clustering and sequence alignment algorithm [6] obviously outperforms the recursive bi-partition one [5]. This result is very similar to that in [6]. We can also observe that the coverage and overflow of [5] is quite un-balanced, which

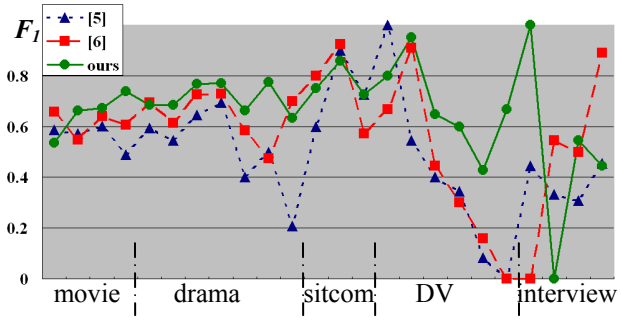

Fig. 7. Performance comparison on different videos.

indicates serious over-segmentation. In the bottom two rows, the F_1 measure of our algorithm is more than 3% higher; and our algorithm achieved a higher coverage and a lower overflow compared to that of [6]. On the whole, it is clear that our algorithm achieved the best performance.

To show the performance on different videos, the F_1 values obtained by the three algorithms are exhibited in Figure 7. Here only results on the main video categories are shown to save space. From Figure 7, we can see that the performance of our algorithm is more stable on different videos, compared to those of [5] and [6]. We can also observe the dramatic variation of performances on the DV works and interviews. This is mainly because the scene number is much less in these videos.

During the process of video scene ground truth labeling, we found that an un-trained person can hardly achieve 90% accuracy after viewing the video only once. Based on this knowledge, our result is acceptable for automatic general video scene segmentation. It is obvious that only visual similarity information is not enough to characterize video scene boundaries. Many other features, such as shot length [1], MFCC [2], audio class [3], can be adopted in the proposed framework as shot similarity measures to better satisfy the requirements of real applications.

Time complexity of the proposed algorithm is shown in Figure 8. The typical processing time for a movie is within 10 minutes; that of a drama is within 2.5 minutes. The result is obtained on an Intel Core2 2.33 GHz CPU with single thread. Note that the processing time is proportional to the number of rounds (10 in the experiments) that the DP runs for each video.

5. CONCLUSION

This paper proposed a novel and superior video scene segmentation algorithm. This algorithm better exploits the correlations among different scenes using the new boundary evaluation criterion, and it finds the global optimal boundary sequence via dynamic programming and a Monte Carlo method. The effectiveness and robustness of the algorithm have been proven through extensive experiments on diversified videos. More features may be integrated in the framework to further enhance the performance.

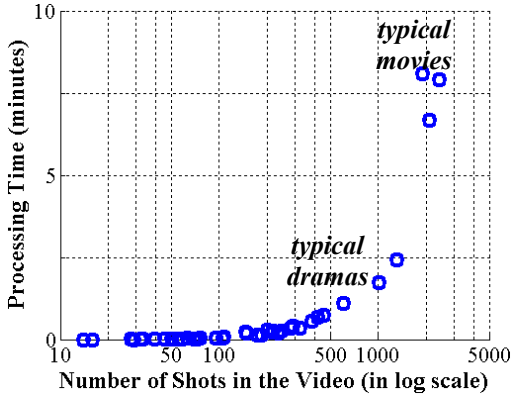

Fig. 8. Time complexity of the proposed algorithm.

6. REFERENCES

[1] Y. Zhai, and M. Shah, "Video scene segmentation using Markov chain Monte Carlo," *IEEE Trans. Multimedia*, vol. 8, no. 4, pp. 686-697, 2006.

[2] N. Goela, K. Wilson, et al, "An SVM framework for genre-independent scene change detection," *IEEE ICME*, 2007, pp. 532-535.

[3] P. Sidiropoulos, V. Mezaris, et al, "Multi-modal scene segmentation using scene transition graphs," *ACM MM*, 2009, pp. 665-668.

[4] J. Wang, L. Duan, et al, "A multimodal scheme for program segmentation and representation in broadcast video streams," *IEEE Trans. Multimedia*, vol. 10, no. 3, pp. 393-408, 2008.

[5] Z. Rasheed, and M. Shah, "Detection and representation of scenes in videos," *IEEE Trans. Multimedia*, vol. 7, no. 6, pp. 1097-1105, 2005.

[6] V. T. Chasanis, A. C. Likas, et al, "Scene detection in videos using shot clustering and sequence alignment," *IEEE Trans. Multimedia*, vol. 11, no. 1, pp. 89-100, 2009.

[7] B. Han, Y. Hu, et al, Enhanced Sports Video Shot Boundary Detection Based on Middle Level Features and a Unified Model, *IEEE Trans. Consumer Electronics*, vol. 53, no. 3, pp. 1168-1176, 2007.

[8] Y. Hu, B. Han, et al, Enhanced Shot Change Detection using Motion Features for Soccer Video Analysis, *IEEE ICME* 2007, pp. 1555-1558.

[9] M. Yeung, B.-L. Yeo, et al, "Segmentation of video by clustering and graph analysis," *Computer Vision and Image Understanding*, vol. 71, no. 1, pp. 94-109, 1998.

[10] C. H. Q. Ding, X. He, et al, "A min-max cut algorithm for graph partitioning and data clustering," *IEEE ICDM*, 2001, pp. 107-104.

[11] N. Dalal and B. Triggs. "Histograms of Oriented Gradients for Human Detection," *IEEE CVPR*, 2005, pp. 886-893.

[12] J. Vendrig, and M. Worring, "Systematic evaluation of logical story unit segmentation," *IEEE Trans. Multimedia*, vol. 4, no. 4, pp. 492-499, 2002.