MSR-Asia at TREC-10 Video Track: Shot Boundary Detection Task

Yu-Fei Ma[*], *Jia Sheng*[+], *Yuan Chen*[#], *Hong-Jiang Zhang*[*]

[*] Microsoft Research, Asia {i-yfma, hjzhang}@microsoft.com
[+]Department of Computer Science and Technology, Tsinghua University
[#] Computer and Information Science and Engineering Department, University of Florida

Abstract

The video track is added into TREC-10, composed of two tasks, automatic shot boundary detection and video retrieval. In this year, we (MSR-Asia) participated in the video track, focusing on shot boundary detection task. Our work is to find out all of boundaries the shot changes by a fast algorithm based on uncompressed domain. In our algorithm, all of non-cut transitions are considered as gradual transition, including dissolve, fade-in, fade-out, and all kinds of wipes. Experimental results indicate that the accuracy and processing speed of our algorithm are all very satisfactory.

1. Introduction

Shot is the basic unit of video sequence, hence, is important for digital video processing. Shot boundary detection is the first step for video content analysis. Since there are usually many shots in a video sequence, the automatic algorithm for shot boundary detection is indispensable.

The shot transition can be classified into two types: abrupt transition (cut) and gradual transition. Gradual transition usually includes dissolve, fade-in, fade-out and all kinds of wipes. Cuts are generated by camera operations, such as starting or stopping recording, or editing operations, while gradual transitions are generated only by editing operations

There are so many literatures addressing the algorithms of shot boundary detection [1-7]. Two fundamental approaches are used: 1) Compressed domain based methods [3,4,5], and 2) Uncompressed domain based methods [2,6,7]. The former usually is much faster than the latter, but its accuracy is difficult to be improved. Additionally, the former must be adaptive to different compression formats or decoders. Comparing with the former, much more methods could be used for the uncompressed domain based methods. Moreover, with the enhancement of hardware and compression standards, the decoding speed is never a drawback.

In this paper, we propose an uncompressed domain based approach for fast shot boundary detection. We employ a block-wise comparison based algorithm for cut detection and a run-length based algorithm for gradual transition detection. They are integrated seamlessly under the framework of Finite State Automata. In addition, the self-adaptive thresholds are used for robustness purpose. The experiments were carried out on large amount video sequences. The test set is provided by NIST (National Institute of Standard and Technology). The results are also evaluated by NIST.

The rest of paper is organized as follows. In section 2, we first introduce the system framework. The details of our algorithms will be described in section 3. Then the experimental results are presented in section 4. Section 5 draws some conclusions.

2. Framework

Our approach consists of four functional modules which are decoding, feature extraction, inter-frame comparison, and decision, as shown in Fig.1. Since our algorithm is based on uncompressed data, the video sequences would be decoded in the decoding module first, if it is in a compressed format, such as mpeg. Then, the visual features are extracted from each decoded frames and compared in the feature extraction module and the Inter-frame comparison module respectively. In our

---

[+] [#] This work was done while these authors were visiting Microsoft Research, Asia.

algorithm, block-based average color and color histograms are used as visual features. With a Finite State Automata, the shot boundaries are detected by self-adaptive threshold in the decision module.

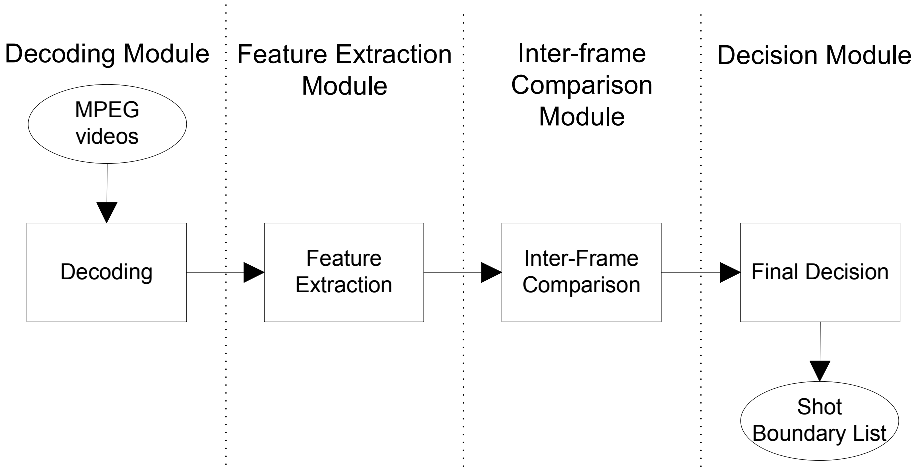


Fig.1: Framework

3 Algorithms Description

In our algorithms, we assume that if there is an apparent deviation in the visual feature between two frames, a shot transition will occur there. So the difference between two successive frames is used as a measure of variation of video sequence. The key issues here are 1) what visual features are extracted from frames, ands 2) what similarity measure is adopted. We used different methods to deal with cut and gradual transition.

3.1 Cut Detection

The pixel-wise difference between two successive frames could be used as dissimilarity measure. However, it is very sensitive to the motion, including camera movement and object motion. To reduce the disturbance of motion, we employ a block based comparison. In RGB color space, let $c_{ij}^{(t)} = (r_{ij}^{(t)}, g_{ij}^{(t)}, b_{ij}^{(t)})$ denote the color of the pixel at the pixel (i, j) in the *t*-th frame. Then we divide each frame into $m \times n$ blocks and comparisons are carried out on blocks instead of pixels. The average color of block (p, q) in the *t*-th frame could be defined as follows:

$$\overline{c_{pq}^{(t)}} = \frac{1}{m \times n} \sum_k c_k^{(t)} = \frac{1}{m \times n} \sum_k (r_k^{(t)}, g_k^{(t)}, b_k^{(t)}) \tag{1}$$

where $c_k^{(t)}$ is the pixel color within a block field. Then the block-wise difference between *t* and *(t-1)*-th frame is defined:

$$d'^{(t)}_{pq} = | \overline{c_{pq}^{(t)}} - \overline{c_{pq}^{(t-1)}} | = | \overline{r_{pq}^{(t)}} - \overline{r_{pq}^{(t-1)}} | + | \overline{g_{pq}^{(t)}} - \overline{g_{pq}^{(t-1)}} | + | \overline{b_{pq}^{(t)}} - \overline{b_{pq}^{(t-1)}} | \tag{2}$$

Finally, we count the number of blocks in frame *t* whose difference $d'^{(t)}_{pq}$ exceeds a threshold T_b. The inter-frame difference value $D_c^{(t)}$ is thus decided by the proportion of the blocks which are sufficiently different between each other.

$$D_c^{(t)} = n_{d_{pq}^{(t)} \geq t} / (m \times n) \tag{3}$$

If the $D_c^{(t)}$ is larger than another threshold T_c, we will declare the current frame as a cut boundary. Experiment results show that this block-wise comparison based method lessens the influence of global and local motion effectively.

After we got the differences between every consecutive frame pair, we can put them along the time axis to observe the temporal distribution of the frame difference values. Fig.2. shows the inter-frame pixel-wise difference values of a video sequence, which includes two cut transitions. The cut position can be clearly observed.

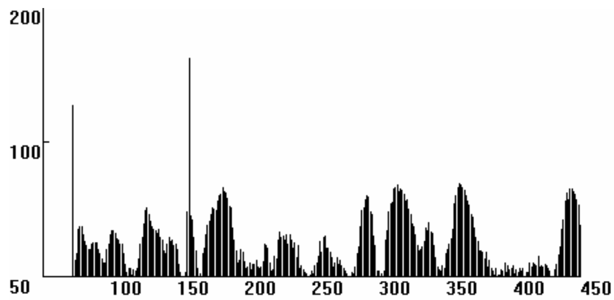


Fig.2. **Frame-to-frame Different based on Block-wise Comparison**
Two cut transitions occur at about frame #70 and frame #155 and the "jump up" is easy to figure out.

Since it is difficult for a global threshold to be suitable for any type of videos, even for different segments in one sequence, we adopt a sliding widow scheme for getting a self-adaptive threshold locally. It could be decided by (4) considering the local first several maximum values during previous *m* frames.

$$T^{(t+1)} = w_1 \times \frac{1}{n_0} \times \sum LMax \tag{4}$$

where $LMax$ denotes the first n_0 maximum values during that sliding window, $\frac{1}{n_0} \times \sum LMax$ is the average difference and w_1 is the weight factor. By self-adaptive threshold, our method could adapt to the variation of motion intensity in video. When the motion is intense, the threshold will become higher; and when the motion slows down, the threshold will also drop.

3.2 Gradual Transition Detection

Due to the complexity of gradual transition, we extract color histogram from each frame as visual feature. Let $His^{(t)}$ denotes the color histogram of the *t*-th frame in RGB color space. Then we define a dissimilarity measure of successive frames based on histogram intersection as (5):

$$D_g^{(t)} = 1 - \frac{\sum_{i=1}^{N} MIN(His^{(t)}[i], His^{(t-1)}[i])}{m \times n} \tag{5}$$

where $His^{(t)}[i]$ denotes the number of pixels falling into *i*-th bin, and $m \times n$ is the total number of pixels in one frame.

By observing the dissimilarity sequence in Fig. 3, we can see that the difference between the frames during the dissolve are higher, although only slightly, than those in the preceding and following frames. Moreover, it is a continuous region. Our task is to find the boundary of this region viz. the start and the end frame numbers. We propose a run-length based method to detect this kind of regions. First, we still need a self-adaptive threshold much lower than cut's threshold to detect the variation during gradual transition. If the dissimilarity of two successive frames is higher than this threshold, we count the frame number until this criterion is not met. We call this number as run-length. If the run-length reaches sufficient length, a gradual transition is declared. Otherwise, no gradual transition occurs. Unlike cut detection, we take into account the all of values in the previous sliding window except for those very high or very low values to decide threshold this time. It could be defined as (6)

$$T_g^{(t+1)} = w_2 \times \frac{1}{n_0} \times \sum LMedian \tag{6}$$

where $LMedian$ are the median values in the sliding window, $\frac{1}{n_0} \times \sum LMedian$ is the average value and w_2 is the weight factor.

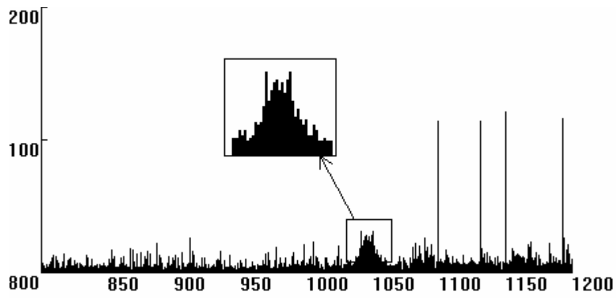
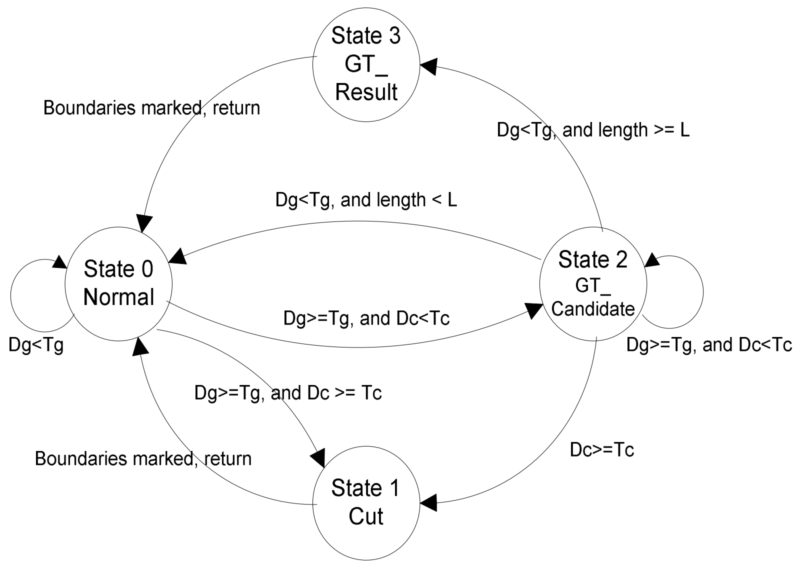


Fig. 3. Frame-to-frame Difference based on Histogram Intersection.
A graduate transition occurs from #1040 to #1060.

3.3 Integration

In the final decision module, the two detection algorithms above are integrated by finite state automata (FSA), shown as Fig. 4. There are 4 states in this FSA for each frame: State 0 is normal state, namely, no cut or gradual transition occurs in current frame; State 1 denotes a cut occurred in current frame; State 2 is a state during a suspect gradual transition; and State 3 denotes a gradual transition occurred from the frame enters State 2 to that enters State 3.



Dc: Inter-frame difference based on Block Pair-wise Comparison
Dg: Inter-frame difference based on Histogram Intersection
Tc: Cut threshold
Tg: Graduate transition threshold
L: Graduate transition minimum length

Fig. 4: **Integration in Finite State Automata**

When a detection process is started, the frame is assumed in State 0 by default. If $D_g<T_g$, FSA will keep in State 0. Otherwise we consider D_c. If $D_c >=T_c$, FSA will transfer to State 1, a cut is declared and FSA will go back to State 0. When $D_g>=T_g$ and $D_c<T_c$, FSA will transfer to State 2, entering a potential gradual transition, and the run-length detection process will be started. In this state, if the condition $D_g>=T_g$ and $D_c<T_c$ is satisfied, the state will be kept. When $D_g<T_g$, the run-length detection will finish. If the duration of run-length process is long enough, such as length>=L, FSA will jump to State 3, a gradual transition is declared, then FAS go back to State 0. When FAS is in State 2, there is another path to jump. If $D_c>=T_c$, FSA will jump to State 1, a cut will be declared, then FAS go back to State 0. The FSA will resume a new detection process, if only it goes back to Sate 0.

4. Experiments

Experiments are carried out on the test video sequences provided by NIST, and the results are also evaluated by NIST. NIST provided participants of video track about 11 hours video data. Among them, more than 5 hours' data are used as final test set including 42 video sequences. In this section, we pick out 16 video sequences with the bigger size from the NIST evaluation results to analysis the performance of our algorithms. As we can see, Table. 1 lists the general evaluation results considering cut and gradual transition as a whole. Table. 2 and Table. 3 list the evaluation results of cut and gradual transition detection respectively. Besides, we also test processing speed of our algorithm on PC with the configuration of PIII 450MHz, 256MB. The results are listed in Table. 4.

Table. 1 Evaluation Results: as a whole

Name	Frame Number	Reference Transition Count	Deletion rate	Insertion rate	Recall	Precision	Correction probability
ahf1.mpg	15679	107	0.102	0.168	0.897	0.842	0.948
anni005.mpg	11364	65	0.153	0.107	0.846	0.887	0.922
anni009.mpg	12307	103	0.368	0.155	0.631	0.802	0.814
bor03.mpg	48451	237	0.067	0.139	0.932	0.870	0.965
bor08.mpg	50569	528	0.094	0.140	0.905	0.865	0.951
bor12.mpg	24550	135	0.340	0.140	0.659	0.824	0.829
bor17.mpg	49801	246	0.337	0.150	0.662	0.815	0.830
eal1.mpg	16048	81	0.271	0.370	0.728	0.662	0.863
nad28.mpg	52927	298	0.161	0.177	0.838	0.825	0.918
nad31.mpg	52405	239	0.175	0.213	0.824	0.794	0.911
nad33.mpg	49768	214	0.046	0.168	0.953	0.85	0.976
nad53.mpg	25783	158	0.107	0.183	0.892	0.829	0.945
nad57.mpg	12781	67	0.208	0.208	0.791	0.791	0.894
pfm1.mpg	14686	82	0.134	0.231	0.865	0.788	0.932
senses111.mpg	86789	308	0.090	0.142	0.909	0.864	0.954
ydh1.mpg	22276	119	0.168	0.252	0.831	0.767	0.915
Average	34136	187	0.176	0.184	**0.823**	**0.817**	**0.910**

Table. 2 Evaluation Results: Cut

Name	Frame Number	Reference Transition Count	Deletion rate	Insertion rate	Recall	Precision	Correction probability
ahf1.mpg	15679	62	0.080	0.161	0.919	0.850	0.959
anni005.mpg	11364	38	0.026	0.052	0.973	0.948	0.986
anni009.mpg	12307	38	0.157	0.105	0.842	0.888	0.920
bor03.mpg	48451	226	0.061	0.044	0.938	0.954	0.968
bor08.mpg	50569	375	0.034	0.128	0.965	0.882	0.982
bor17.mpg	49801	126	0.087	0.015	0.912	0.982	0.956
eal1.mpg	16048	61	0.131	0.0	0.868	1.0	0.934
nad28.mpg	52927	181	0.160	0.077	0.839	0.915	0.919
nad31.mpg	52405	183	0.087	0.060	0.912	0.938	0.956
nad33.mpg	49768	188	0.010	0.037	0.989	0.963	0.994
nad53.mpg	25783	81	0.024	0.024	0.975	0.975	0.987
nad57.mpg	12781	44	0.227	0.022	0.772	0.971	0.886
pfm1.mpg	14686	61	0.098	0.081	0.901	0.916	0.950
senses111.mpg	86789	292	0.061	0.006	0.938	0.992	0.969
ydh1.mpg	22276	67	0.014	0.194	0.985	0.835	0.992
Average	34776	135	0.084	0.067	**0.915**	**0.934**	**0.957**

Table. 3 Evaluation Results: Gradual Transition

Name	Frame Number	Reference Transition Count	Deletion rate	Insertion rate	Recall	Precision	Correction probability
ahf1.mpg	15679	45	0.133	0.177	0.866	0.829	0.933
anni005.mpg	11364	27	0.333	0.185	0.666	0.782	0.833
anni009.mpg	12307	65	0.492	0.184	0.507	0.733	0.753
bor03.mpg	48451	11	0.181	2.090	0.818	0.281	0.908
bor08.mpg	50569	153	0.241	0.169	0.758	0.816	0.878
bor12.mpg	24550	135	0.340	0.140	0.659	0.824	0.829
bor17.mpg	49801	120	0.6	0.291	0.4	0.578	0.699
eal1.mpg	16048	20	0.7	1.5	0.3	0.166	0.649
nad28.mpg	52927	117	0.162	0.333	0.837	0.715	0.918
nad31.mpg	52405	56	0.464	0.714	0.535	0.428	0.767
nad33.mpg	49768	26	0.307	1.115	0.692	0.382	0.845
nad53.mpg	25783	77	0.194	0.350	0.805	0.696	0.902
nad57.mpg	12781	23	0.173	0.565	0.826	0.593	0.912
pfm1.mpg	14686	21	0.238	0.666	0.761	0.533	0.880
senses111.mpg	86789	16	0.625	2.625	0.375	0.125	0.687
ydh1.mpg	22276	52	0.365	0.326	0.634	0.66	0.816
Average	34136	60	0.346	0.714	**0.652**	**0.571**	**0.826**

Table. 4 Processing Speed Comparison

Name	Frame Number	Test time (s)	Normal time (s)	Speed up
ahf1.mpg	15679	367	540	1.47
anni005.mpg	11364	254	379	1.49
anni009.mpg	12307	287	410	1.43
bor03.mpg	48451	1115	1616	1.45
bor08.mpg	50569	1171	1687	1.44
bor12.mpg	24550	565	819	1.45
bor17.mpg	49801	1189	1661	1.40
eal1.mpg	16048	378	540	1.43
nad28.mpg	52927	1177	1766	1.50
nad31.mpg	52405	1250	1748	1.40
nad33.mpg	49768	1152	1660	1.44
nad53.mpg	25783	605	860	1.42
nad57.mpg	12781	297	426	1.43
pfm1.mpg	14686	342	495	1.45
senses111.mpg	86789	1998	2986	1.45
ydh1.mpg	22276	518	743	1.43
Average	34136	792	1146	**1.44**

By observing the evaluation results, it is concluded that:

1) The average probability of correct cut detection is more than 95% with more than 90% average recall and precision. This result indicates that our cut detection algorithm is very effective.
2) The average probability of correct gradual transition detection is more than 80% with about 60% average recall and precision. This result is satisfying considering the complexity of gradual transition. Because gradual transition consists of all kinds of non-cut transitions.
3) The average correction probability of all of transitions is more than 90% with more than 80% average recall and precision. It proves that integration method with finite state automata is much effective and efficient.

4) Comparing with the video playback time, the processing speed of our algorithm is much faster. Without any optimization, the processing speed could approach about 1.5 times of real-time on the PIII 450MHz 256MB personal computer.

On the other hand, we also find some mismatching between our ground truth and those provided by the organizer in the case of graduate transition. These mismatching affect our evaluation results to a certain degree. Some examples are listed in Table.5.

Table. 5. Some Mismatched Samples

Video Sequences	Mismatching in ground truth
bor08.mpg	#49326 to #49349
bor12.mpg	#16497 to #16520
bor17.mpg	#9679 to #9686; #9745 to #9752
nad28.mpg	#327 to #339, #2035 to #2057, #26591 to #26609, #37696 to #37709, #52197 to #52211
… …	……

5. Conclusions

In this paper, we described our work in TREC-10 video track shot boundary detection task. We also reported and analyzed the evaluation results by NIST. The experimental results indicate that our shot boundary detection algorithm based on uncompressed domain is effective and much faster than real-time. By some optimizations, the speed of processing can be further improved.

However, there still is much room to improve our algorithm, especially for gradual transition. For example, some tolerances should be added into run-length based method. Because if the potential gradual transition state ends when only one inter-frame difference drops below the threshold Tg, some gradual transitions would be truncated. Another shortcoming is that the spans of gradual transition we detected are much longer than the real transitions sometime. Besides, how to integrate two detection algorithms also can still lead to additional improvements.

Acknowledgement

The authors would like to thank Dong Zhang for his helpful work on implementing some useful experimental tools.

References

[1] G. Ahanger and Thomas D.C. Little, "A Survey of Technologies for Parsing and Indexing Digital Video", *Journal of Visual Communication and Image Representation*, 7, 1, pp. 28043, 1996.

[2] H.J. Zhang, A. Kankanhalli, and S.W. Smoliar, "Automatic partitioning of full-motion video", *Multimedia Systems*, 1, pp.10-28, 1993.

[3] J. Meng, Y. Juan, S.-F. Chang, "Scene Change Detection in a MPEG Compressed Video Sequence", *Journal*, Publisher, Location, pp. 1-10, Date.

[4] B.L. Yeo and B. Liu, "Rapid scene analysis on compressed video", *IEEE Transactions on Circuits and Systems for Video Technology*, 5, 6, pp. 533-544, 1995.

[5] S.-B. Jun, K. Yoon, H.-Y. Lee, "Dissolve transition detection algorithm using spatio-temporal distribution of MPEG macro-block types", ACM Multimedia 2000, pp. 391-394, 2000.

[6] A. Nagasaka and Y. Tanaka, "Automatic Video Indexing and Full-Video Search for Object Appearances", *Visual Database Systems,* II, pp. 113-127, Elsevier Science Publishers, 1992.

[7] C.F. Lam, M.C. Lee, "Toward some innovative video scene change detection techniques", A CM Multimedia 1998, 1998.