Rapid Cut Detection on Compressed Video

Jurandy Almeida, Neucimar J. Leite, and Ricardo da S. Torres*

Institute of Computing, University of Campinas – UNICAMP
13083-852, Campinas, SP – Brazil
{jurandy.almeida,neucimar,rtorres}@ic.unicamp.br

Abstract. The temporal segmentation of a video sequence is one of the most important aspects for video processing, analysis, indexing, and retrieval. Most of existing techniques to address the problem of identifying the boundary between consecutive shots have focused on the uncompressed domain. However, decoding and analyzing of a video sequence are two extremely time-consuming tasks. Since video data are usually available in compressed form, it is desirable to directly process video material without decoding. In this paper, we present a novel approach for video cut detection that works in the compressed domain. The proposed method is based on both exploiting visual features extracted from the video stream and on using a simple and fast algorithm to detect the video transitions. Experiments on a real-world video dataset with several genres show that our approach presents high accuracy relative to the state-of-the-art solutions and in a computational time that makes it suitable for online usage.

Keywords: video analysis, temporal segmentation, shot boundary, cut detection, compressed domain.

1 Introduction

Recent advances in technology have increased the availability of video data, creating a strong requirement for efficient systems to manage those materials.

Making efficient use of video information requires that the data be stored in an organized way. For this, it must be divided into a set of meaningful and manageable units, so that the video content remains consistent in terms of camera operations and visual events. This has been the goal of a well-known research area, called video segmentation [9].

Different techniques have been proposed in the literature to address the temporal segmentation of video sequences [5, 7, 10, 12–16]. Many of those research works have focused on the uncompressed domain. Although existing methods provide a high quality, they are extremely time-consuming and require a huge amount of space.

In this paper, we present a novel approach for temporal segmentation of video sequences that operates directly in the compressed domain. It relies on exploiting visual features extracted from the video stream and on a simple and fast algorithm to detect the video transitions. The improvement of the computational efficiency makes our technique suitable for online tasks.

* Thanks to Brazilian agencies FAPESP, CNPq, and CAPES for funding.

C. San Martin and S.-W. Kim (Eds.): CIARP 2011, LNCS 7042, pp. 71–78, 2011.
© Springer-Verlag Berlin Heidelberg 2011

We evaluate the proposed algorithm on a real-world video dataset with different video genres and compare our technique with the state-of-the-art approaches for temporal video segmentation. Results from an experimental evaluation over several types of video transitions show that our method presents high accuracy and computational speed.

The remainder of this paper is organized as follows. Section 2 describes related work. Section 3 presents our approach and shows how to apply it to segment a video sequence. Section 4 reports the results of our experiments and compares our technique with other methods. Finally, we offer our conclusions and directions for future work in Section 5.

2 Basic Concepts and Related Work

A video shot is a series of inter-related frames captured from a single camera. In the editing stage of video production, video shots are joined together to form the complete sequence. They represent a continuous action in time and space, where no changes in scene content can be perceived [8].

There are two different types of transitions that can occur between shots: abrupt (discontinuous) transitions, also referred as cuts; and gradual (continuous) transitions, which include camera movements (e.g., panning, tilting, zooming) and video editing effects (e.g., fade-in, fade-out, dissolving, wiping) [9].

A comprehensive review of methods to address the problem of identifying the boundary between consecutive shots can be found in [9, 11]. Most of existing research works have focused on the uncompressed domain. Although those techniques provide a high quality, they spend lots of time and space for decoding and analyzing a video sequence. For this reason, such approaches are unsuitable for online tasks.

The most common approach relies on the definition of similarity metrics between consecutive frames. Usual metrics are based on pixel-wise differences [16] and color histograms [13]. Tracking of image features (e.g., edges [14]) can also be used to detect the shot boundary, since they tend to disappear in a cut. In a different approach, patterns are detected in a bi-dimensional subsampling of the video, called video slice or visual rhythm [5, 7].

Since video data are usually available in compressed form, it is desirable to directly process the compressed video without decoding. It allows us to save high computational load in full decoding the video stream.

Several methods for video segmentation that directly manipulate compressed have been proposed for specific domains, such as sports, music, and news [10, 12, 15]. Focusing on a particular domain helps to reduce levels of ambiguity when analysing the content of a video by applying prior knowledge of the domain during the analysis process [9].

Different from all of the previous techniques which operate directly in the compressed domain, our approach is designed to segment generic videos and, hence, it does not use any specific information beyond the video content.

3 Our Approach

Video data have a lot of redundant information. For saving computational time, the video stream is divided into a set of meaningful and manageable units. Most video codecs (e.g., MPEG-1/2/4) are based on GOPs as basic units. The I-frame contains enough information to characterize the content of the whole GOP.

The compression of the I-frames of a MPEG video is carried out by dividing the original image into 8x8 pixel blocks and transforming the pixels values of each block into 64 DCT coefficients. The DC term $c(0,0)$ is related to the pixel values $f(i,j)$ via the following equation [15]:

$$c(0,0) = \frac{1}{8} \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y).$$

In other words, the value of the DC term is 8 times the average intensity of the pixel block. If we extract the DC term of all the pixel blocks, we can use those values to form a reduced version of the original image. This smaller image is known as the DC image [15]. Fig. 1 illustrates an original image of size 384 × 288 and its DC image with 48 × 36.

Fig. 1. Original image at 384 × 288 and its DC image at 48 × 36. Frame extracted from the video I of the test set.

Initially, we discard a lot of GOPs by computing the pairwise dissimilarity of consecutive I-frames. For this, we convert each DC image to a 256-dimensional feature vector by computing a color histogram. It is extracted as follows: the YCbCr color space is divided into 256 subspaces (colors), using 16 ranges of Y, 4 ranges of Cb, and 4 ranges of Cr. The value for each dimension of the feature vector is the density of each color in the entire DC image.

Let $\mathcal{H}^i$ be the i-th bin of the color histogram $\mathcal{H}$. We measure the dissimilarity between the I-frames by using the well-known histogram intersection, which is define as

$$d(\mathcal{H}_{t_1}, \mathcal{H}_{t_2}) = \frac{\sum_i \min(\mathcal{H}^i_{t_1}, \mathcal{H}^i_{t_2})}{\sum_i \mathcal{H}^i_{t_1}},$$

where $\mathcal{H}_{t_1}$ and $\mathcal{H}_{t_2}$ are the color histograms extracted from the I-frames taken at the times t_1 and t_2, respectively. This function returns a real value ranging

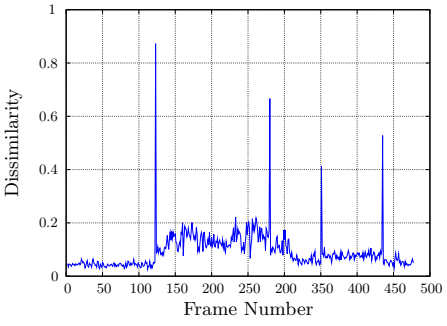


Fig. 2. Pairwise dissimilarities of between frames of the video I of the test set

from 0 for situations in which those histograms are not similar at all, to 1 for situations in which they are identical.

Fig. 2 shows an example of how those values are distributed along time. It can be observed that there are instants of time in which the dissimilarity value varies considerably (corresponding to peaks), while there are longer periods in which the variance is small (corresponding to very dense regions). Usually, peaks correspond to sudden movements in the video or to shot boundaries.

We analyze only the GOPs for which the histogram intersection is below 0.85. If they are completely intra-coded (i.e., only I-frames), we compute the normalized pixel-wise difference of the luminance (Y) between the DC-images. Then, an abrupt cut is declared every time the dissimilarity value is greater than 0.3 and the normalized pixel-wise difference is greater than 0.1. The choice of those values is detailed in Section 4.

Otherwise, we exploit the motion compensation algorithm to detect shot boundaries. For this, we examine the number of inter-coded macroblocks inside each P or B-frame. The main idea is that the motion compensation algorithm cannot find a good match in the nearest past and/or future I and/or P-frames if the GOP are in the shot boundary.

This causes most of the macroblocks of the P-frames to be intra-coded instead of inter-coded. If the ratio of the number of intra-coded macroblocks to the total number of macroblocks is greater than 0.1, there is a high probability to exist a cut in the neighborhood of this P-frame. In this case, we analyze its precedent B-frames to detect both type and location of this video transition. For GOPs that do not contain B-frames, an abrupt cut is declared if the percentage of intra-coded macroblocks in this P-frame is greater than 60%.

Three possible behaviours for the macroblocks of the B-frames are shown in Fig. 3. The width of the arrows indicate the dominant direction for the motion compensation. In this work, a B-frame has a dominant direction if the number of macroblocks with motion compensation in a given direction is the double of that in the opposite direction.

If most of the B-frames are encoded with forward motion compensation, an abrupt transition is detected between the last B-frame and its subsequent anchor frame (I or P). On the other hand, if most of the B-frames are encoded with backward motion compensation an abrupt transition is detected between the

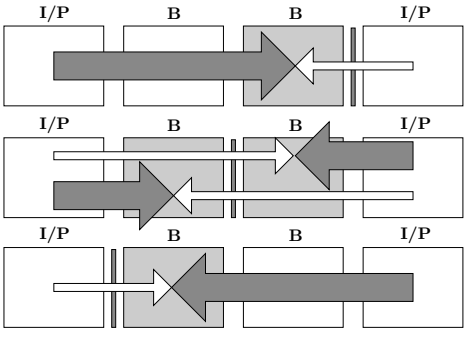


Fig. 3. The possible behaviours for the macroblocks of the B-frames

first B-frame and its precedent anchor frame (I or P). Finally, if the forward direction is dominant in the first half of the B-frames and the backward direction is dominant the last half of the B-frames, then an abrupt cut is detected in the middle of those sequence. In order to avoid false alarms, a video transition is declared only if the percentage of intra-coded macroblocks in such B-frames is greater than 50%.

If none of the above conditions is satisfied, we check if there exists a possible gradual transition. For this, we examine the variation of the percentage of inter-coded macroblocks with forward motion compensation along the frames of the GOP. In the case of gradual transitions, those values form a *plateau* (i.e., an isosceles trapezoid), as first observed by Yeo and Liu [15]. Since a gradual transition has a certain duration, at least 7 frames should be involved to declare a shot boundary.

4 Experiments and Results

Experiments were carried out on a real-world video dataset with known ground-truth data. For benchmarking purposes, we used the test set[1] presented in [14]. This benchmark contains 10 video sequences, including a variety of genres and quality levels, as shown in Table 1.

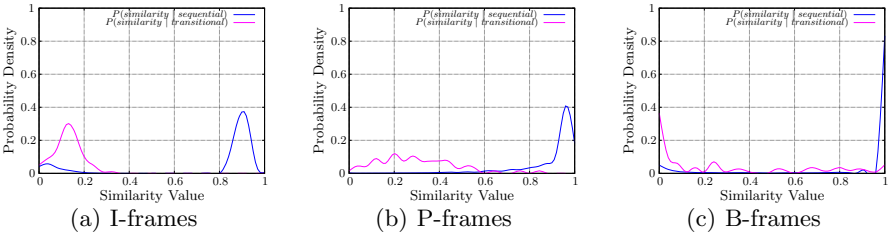
For selecting the threshold values used to detect video transitions, we computed the pairwise similarities of consecutive frames. Remember that the similarity value between the I-frames is measured using the histogram intersection, whereas for P- or B-frames the similarity value relies on the ratio of the number of inter-coded macroblocks to the total number of macroblocks.

Fig. 4 presents the probability density function (PDF) for the distribution of the similarity value of sequential and transitional frames. Notice that the PDF curves for the different types of video frames are well-separated, evidencing the high discriminating capability of our strategy.

[1] All the video sequences and the ground-truth data are available at `http://www.site.uottawa.ca/~laganier/videoseg/` (last accessed on 20 July 2011).

Table 1. The main characteristics of each video sequence in test set

Video	Genre	Dur. (s)	Dim.	GOP size	# Frames
A	cartoon (low quality)	21	192 × 144	6	650
B	action (motion)	38	320 × 142	6	959
C	horror (black/white)	53	384 × 288	2	1619
D	drama (high quality)	105	336 × 272	15	2632
E	science-fiction	17	384 × 288	2	536
F	commercial (effects)	7	160 × 112	15	236
G	commercial	16	384 × 288	1	500
H	comedy	205	352 × 240	12	5133
I	news	15	384 × 288	1	479
J	action	36	240 × 180	12	873



(a) I-frames (b) P-frames (c) B-frames

Fig. 4. The probability density function (PDF) for the distribution of the similarity value of different types of video frame

We assess the effectiveness of the proposed method using the metrics of precision and recall. Precision (P) is the ratio of the number of temporal positions correctly identified as cuts to the total number of temporal positions identified as cuts. Recall (R) is the ratio of the number of temporal positions correctly identified as cuts to the total number of cuts in the video sequence. However, there is a trade-off between precision and recall. Greater precision decreases recall and greater recall leads to decreased precision. So, we also employ the F-measure for assessing the quality of the temporal segmentation. The F-measure (F) combines both precision and recall into a single measure by a harmonic mean:

$$F = \frac{2 \times P \times R}{P + R}.$$

The experiments were performed on a machine equipped with an Intel Core 2 Quad Q6600 processor (four cores running at 2.4 GHz) and 2 GBytes of DDR3-memory. The machine run Ubuntu Linux (2.6.31 kernel) and the ext3 file system.

Table 2 compares our technique with four different approaches: (1) histogram-based method [13], (2) feature-based method with automatic threshold selection [14], (3) visual rhythm with longest common subsequence (LCS) [5], and (4) visual rhythm with clustering by k-means [7]. The average rates of each quality measure correspond to the weighted mean of individual results, whose weights are the total number of cuts in each video. In addition, the weighted standard deviations reveal the amount of dispersion with respect to those values.

Table 2. Comparison of precision (P), recall (R), and F-measure (F) achieved by different approaches for each video of the test set

Video	Our proposal (compressed)			Visual rhythm with k-means [7]			Visual rhythm with LCS [5]			Feature tracking method [14]			Histogram (MOCA) [13]		
	P	R	F	P	R	F	P	R	F	P	R	F	P	R	F
A	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.857	0.923	1.000	1.000	1.000	1.000	1.000	1.000
B	1.000	1.000	1.000	0.500	1.000	0.667	0.096	1.000	0.176	1.000	1.000	1.000	1.000	0.375	0.545
C	0.891	0.907	0.899	0.662	0.907	0.766	0.635	0.870	0.734	0.595	0.870	0.707	0.936	0.536	0.682
D	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.971	0.985	1.000	1.000	1.000	1.000	0.941	0.969
E	0.815	0.786	0.800	0.828	0.857	0.842	0.676	0.821	0.742	0.938	1.000	0.968	0.955	0.700	0.808
F	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
G	0.938	0.833	0.882	0.950	1.000	0.974	1.000	0.842	0.914	0.810	0.944	0.872	1.000	0.667	0.800
H	0.974	0.949	0.961	0.949	0.974	0.961	0.943	0.868	0.904	0.895	0.895	0.895	0.971	0.895	0.932
I	1.000	1.000	1.000	1.000	1.000	1.000	0.667	0.500	0.571	1.000	1.000	1.000	1.000	0.500	0.667
J	0.776	0.506	0.612	0.683	0.869	0.765	0.639	0.885	0.742	0.497	0.897	0.637	0.850	0.395	0.540
Avg.	0.882	0.786	0.825	0.793	0.923	0.848	0.745	0.878	0.789	0.730	0.924	0.803	0.932	0.621	0.730
Dev.	0.095	0.213	0.164	0.156	0.061	0.111	0.205	0.067	0.153	0.220	0.054	0.157	0.063	0.229	0.178

The results indicate that the proposed method is robust to several conditions (e.g., frame rate, frame size, total duration, etc.), showing high accuracy compared to the state-of-the-art solutions. Notice that our approach provides the best achievable F-measure for the majority of the video sequences (7 of 10).

The key advantage of our technique is its computational efficiency. Since the time required to segment video sequences is hardware dependent (with faster hardware the computational speed increases and the production time decreases) and the source codes of all the compared methods are not available, it is impossible to perform a fair comparison of performance in relation to our technique.

In order to evaluate the efficiency of our approach, we analyze the time per frame spent for processing all the steps of our algorithm, excluding the time for the partial decoding of each frame. We performed 10 replications for each video in order to guarantee statistically sound results.

According to those experiments, all the steps of our technique takes a mean time equals to 112 ± 38 microseconds per frame (confidence higher than 99.9%). For online usage, by considering a maximum waiting time of 39 seconds [6], the proposed method can be used for videos up to 349515 frames (about 194 minutes at 29.97 frames per second). It is important to recall that those values depend on the computational power of the employed hardware.

5 Conclusions

In this paper, we have presented a novel approach for video cut detection that works in the compressed domain. Our technique relies on exploiting visual features extracted from the video stream and on using a simple and fast algorithm to detect the video transitions. Such combination makes our technique suitable for online usage.

We have validated our technique using a real-world video dataset with different video genres and compared our technique with the state-of-the-art approaches for temporal video segmentation. Results from an experimental evaluation over several types of video transitions show that our method presents high accuracy and computational speed.

Future work includes the evaluation of other visual features and similarity metrics. In addition, the proposed method can be augmented to consider local features [4] and/or motion analysis [2, 3]. Finally, we want to investigate the effects of integrating our technique into a complete system for search-and-retrieval of video sequences [1].

References

1. Almeida, J., Leite, N.J., Torres, R.S.: Comparison of video sequences with histograms of motion patterns. In: Int. Conf. Image Processing (ICIP 2011) (2011)
2. Almeida, J., Minetto, R., Almeida, T.A., Torres, R.S., Leite, N.J.: Robust estimation of camera motion using optical flow models. In: Bebis, G., Boyle, R., Parvin, B., Koracin, D., Kuno, Y., Wang, J., Wang, J.-X., Wang, J., Pajarola, R., Lindstrom, P., Hinkenjann, A., Encarnação, M.L., Silva, C.T., Coming, D. (eds.) ISVC 2009. LNCS, vol. 5875, pp. 435–446. Springer, Heidelberg (2009)
3. Almeida, J., Minetto, R., Almeida, T.A., Torres, R.S., Leite, N.J.: Estimation of camera parameters in video sequences with a large amount of scene motion. In: Proc. of Int. Conf. Syst. Signals Image (IWSSIP 2010), pp. 348–351 (2010)
4. Almeida, J., Rocha, A., Torres, R.S., Goldenstein, S.: Making colors worth more than a thousand words. In: Int. Symp. Applied Comput. (ACM SAC 2008), pp. 1180–1186 (2008)
5. Bezerra, F.N., Leite, N.J.: Using string matching to detect video transitions. Pattern Anal. Appl. 10(1), 45–54 (2007)
6. Bouch, A., Kuchinsky, A., Bhatti, N.T.: Quality is in the eye of the beholder: meeting users' requirements for internet quality of service. In: Int. Conf. Human Factors Comput. Syst. (CHI 2000), pp. 297–304 (2000)
7. Guimarães, S.J.F., Patrocínio Jr., Z.K.G., Paula, H.B., Silva, H.B.: A new dissimilarity measure for cut detection using bipartite graph matching. Int. J. Semantic Computing 3(2), 155–181 (2009)
8. Hanjalic, A.: Shot-boundary detection: Unraveled and resolved? IEEE Trans. Circuits Syst. Video Techn. 12(2), 90–105 (2002)
9. Koprinska, I., Carrato, S.: Temporal video segmentation: A survey. Signal Processing: Image Communication 16(5), 477–500 (2001)
10. Lee, S.W., Kim, Y.M., Choi, S.W.: Fast scene change detection using direct feature extraction from MPEG compressed videos. IEEE Trans. Multimedia 2(4), 240–254 (2000)
11. Lienhart, R.: Reliable transition detection in videos: A survey and practitioner's guide. Int. J. Image Graphics 1(3), 469–486 (2001)
12. Pei, S.C., Chou, Y.Z.: Efficient MPEG compressed video analysis using macroblock type information. IEEE Trans. Multimedia 1(4), 321–333 (1999)
13. Pfeiffer, S., Lienhart, R., Kühne, G., Effelsberg, W.: The MoCA project - movie content analysis research at the University of Mannheim. In: GI Jahrestagung, pp. 329–338 (1998)
14. Whitehead, A., Bose, P., Laganière, R.: Feature based cut detection with automatic threshold selection. In: Enser, P.G.B., Kompatsiaris, Y., O'Connor, N.E., Smeaton, A., Smeulders, A.W.M. (eds.) CIVR 2004. LNCS, vol. 3115, pp. 410–418. Springer, Heidelberg (2004)
15. Yeo, B.L., Liu, B.: Rapid scene analysis on compressed video. IEEE Trans. Circuits Syst. Video Techn. 5(6), 533–544 (1995)
16. Zhang, H., Kankanhalli, A., Smoliar, S.W.: Automatic partitioning of full-motion video. Multimedia Syst. 1(1), 10–28 (1993)