Fast Frame-Based Scene Change Detection in the Compressed Domain for MPEG-4 Video

Jens Brandt Jens Trotzky Lars Wolf
IBR, Technische Universität Braunschweig
Mühlenpfordtstraße 23, 38106 Braunschweig, Germany
$\{$brandt$|$trotzky$|$wolf$\}$@ibr.cs.tu-bs.de

Abstract

Detection of scene changes is an elementary step in automatic video processing like indexing, segmentation or transcoding. Video indexing and segmentation allow fast browsing without decoding the complete video. In the case of transcoding, the information about scene changes such as cuts and fades as well as about special movements like rotations or zooms in video frames is helpful to determine suitable transcoding parameters. In the compressed domain only information about DCT values as well as motion information can be used to determine such scene changes and movements. Therefore we defined different measures which use the encoded DCT values and motion vectors of each compressed frame. Based on these measures as well as on motion vector histograms we present a fast approach to detect different kinds of scene changes and special movements in MPEG-4 videos in the compressed domain.

1. Introduction

Scene change detection is an important mechanism for automatic video indexing and segmentation. For video adaptation the information about scene changes can be used to determine suitable adaptation parameters. When working in the decompressed domain (i. e., the pixel domain), scene changes can be detected by inspecting the pixel values of the video frames as well as the changes between successive frames. However, video analysis in the compressed domain is much more favourable as most video streams are stored in a compressed format, which makes working in the decompressed domain computationally expensive. In the case of MPEG-compressed video, this can be achieved by analysing the discrete cosine transform (DCT) values and motion information of each frame, which are directly accessible in the compressed domain. Typically scene changes can be detected by inspecting the differences of successive frames. Due to the nature of MPEG compression, the encoded DCT values and motion vectors of a frame already contain information about these differences. Thus, when working in the compressed domain, we can reuse this existing information to decide about the existence of a scene change on a frame by frame basis. When analysing the DCT values as well as the motion information we can also detect special movements like rotations or zooms within the video stream. For the adaptation of compressed video, detection of scene changes and special movements can provide helpful information to determine suitable adaptation parameters as well as to decide whether a certain frame should be skipped.

This paper describes a fast frame-based scene change detection algorithm for MPEG-4 video which is working in the compressed domain. By analysing DCT values and motion information contained in the compressed video stream we define different measures and motion vector histograms for each frame. These measures and histograms are used to decide whether a frame is part of a scene change or not and can be computed very fast. By detecting scene changes and special movements within a video stream we expect to automatically chose transcoding parameters more exactly, which enhances the produced quality of the transcoder. Further we expect to be able to use the presented metrics to decide whether to skip a certain frame or not. For instance, in case of a scene cut none of the frames immediately before or after the cut should be skipped. In our previous work we presented a transcoding architecture which can be used in such an environment [5]. The implementation of our scene change detection mechanism, which we used for the evaluation of the algorithm is based on this architecture.

In this paper we concentrate on P-frames within the video streams. The videos which we used to develop and to evaluate our algorithm were encoded by using only one key frame (i. e., I-frame) at the beginning and solely P-frames for the rest of the stream. Because I-frames do not contain any motion information our algorithm cannot definitely decide whether such a frame belongs to a scene change or not.

However, most encoders already use basic scene change detection mechanisms and for those video streams which are encoded with a minimum number of I-frames it is very likely that an I-frame is a cut between two scenes. The primary objective of our algorithm design is speed and minimal workload for analysis and decision making. The mechanisms described here are meant to provide information on digital video scenes in an automatic transcoder environment providing on-demand video adaptation for mobile devices such as mobile phones and personal digital assistants.

The remainder of this paper is organized as follows: First, section 2 gives a general idea of previously published papers in the research area of scene change detection for digital video. Following this introductory section on related work section 3 provides the basic tools for coded MPEG-4 video analysis which are used within our proposed algorithm for scene change detection. The actual algorithm and its components are described in section 4. Following the theoretical description and the implementation of the algorithm section 5 evaluates the capabilities in practical tests. Finally we conclude this paper in section 6.

2. Existing Approaches

As described by Boreczky et al. [3] there has been a lot of research work on video segmentation and scene change detection. Several approaches are working in the uncompressed domain such as those described in [2, 6, 8, 9, 10]. It is important to mention that some of these approaches use very specific knowledge about the video content, such as for detecting scene changes in video coverage of sporting events by Han et al. [10]. Some papers can be found which are decoding only a low resolution version of each frame [12, 13, 16, 18] and therefore avoid the complexity of a complete decoding. Another set of papers deal with analysing video sequences in the compressed domain by inspecting the differences of successive frames [1, 14] and by using statistical tests [11, 17] for scene change detection.

Pei et al. proposed an algorithm which uses the macro block type for detection of wipe effects in the compressed domain [15]. However, due to the highly artificial nature of wipe effects this approach cannot be used for general scene change detection. Analysing information from the motion vectors in the compressed domain has been done for several versions of MPEG video [4, 7]. The idea of using a combination of multiple cues, which are based on motion vectors and macro block information, was presented in [19]. But the computations needed to derive the cues still use information from the decompressed domain. Our work incorporates this idea of combining motion vector and DCT value analysis but it completely avoids the use of decoded information and therefore saves the processing time needed for video decoding.

In a way our approach is of minimalistic nature so that the video in question does not need to be decoded at all. Furthermore, we use metrics and motion vector histograms that are easy to compute and therefore our algorithm needs only a minimum of processing time for detecting scene changes and special movements. The video format in question shall be MPEG-4 and the following sections of this work will only refer to the macro blocks, the motion vectors, and the AC and DC values from the DCT in their coded form. As quality and accuracy of the scene change detection shall be of interest as well, speed and easy computation of a decision are foremost important, because our approach is targeted for solutions that are providing video adaptation for mobile devices with limited resources.

3. Frame Analysis

3.1. Macro Block Coding

In MPEG-compressed video each frame is divided into 8×8 pixel blocks. Each four neighbouring blocks build a macro block (MB) which holds the motion information in the form of one or four motion vectors (MV). In the compressed domain each block contains 8×8 DCT values, i. e., one DC value and 63 AC values. Depending on the frame type three different macro block types exist: i) inter-coded macro blocks use motion information in the form of motion vectors pointing into one or two reference frames ii) intra-coded macro blocks only contain DCT values without any motion information; and iii) not coded macro blocks do not contain any information. The latter type of macro blocks should be treated by the decoder as inter-coded with a zero length motion vector. I-frames may only consist of intra-coded macro blocks, whereas P- and B-frames may contain both, intra- and inter-coded macro blocks.

Since neither the initial settings nor the performance of the encoder which was used to encode a video are known when processing a compressed video stream, one cannot conclude a typical underlying content or specific scene change based on the coding type of the macro blocks only. However, typical encoders prioritize size as a major factor when encoding a video into MPEG-4 video. During this process motion compensation supports the idea of referring to the same content in the vicinity of a macro block in a preceding or succeeding frame. If no macro block with a similar content is found in the surrounding area the macro block has to be coded as an intra-coded macro block.

For further analysis of the DCT values we define a complexity measure for the whole frame which is based on the number of non-zero DCT values $n_{\neg 0}$ of all macro blocks:

$$c = \frac{n_{\neg 0}}{64 \cdot n_B} \quad (1)$$

with 64 being the number of DCT values in each 8×8 pixel block and n_B being the number of blocks within the frame. The complexity c can therefore range between 0 and 1 and the lower c the less complex the frame is.

In the case of inter-coded frames, i. e., P- and B-frames, macro blocks may be inter- or intra-coded depending on the decision at encoding time. The ratio of intra- and inter-coded macro blocks of inter-coded frames is of interest, as one can use the information that a block was initially intra-coded as a hint that motion estimation was unsuccessful. Thus we define a second measure for each frame, based on the macro block types within the frame, which we call the intra-ratio:

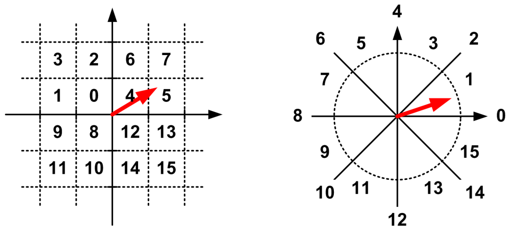
$$r_{INTRA} = \frac{n_{MB,INTRA}}{n_{MB}} \qquad (2)$$

with $n_{MB,INTRA}$ being the number of intra-coded macro blocks in the frame and n_{MB} being the total number of macro blocks in the frame. Based on this hint about unsuccessful motion estimation the intra-ratio is very handy for detecting scene cuts, which will be seen later on in this paper.

3.2. Motion Vector Analysis

Motion in videos shall be interpreted as the change or variation of a reference image over time. In this instance time is split into discrete time codes assigned to individual video frames. By working with a set of full frames motion can be tracked as the difference between succeeding images. MPEG-4 makes use of those differences in the encoding process and can track motion by identifying similar content in the vicinity of the previous frame for each macro block. This in return allows for more detailed analysis of the frame. During this search two major cases can be differentiated:

1. The tracking of the same content in a reference frame was successful and the motion can be described using a motion vector. This can be the result of two cases:
   - The content looked up is in fact the same as the content from the reference frame.
   - The content matches an area with similar content in the reference frame but from a different scene.
2. The tracking was unsuccessful within the analysed area, which can have several causes:
   - The content is not available as the scene has changed or the content has changed too much.
   - The content is out of range, meaning it moved too far from its origin position.



(a) Cartesian (b) Polar coordinate

Figure 1. MV classification schemes

3.3. Motion Vector Ratio

As we have seen before, not all macro blocks contain motion information in the form of motion vectors. Intra-coded macro blocks do not contain any motion information and inter-coded macro blocks may contain 1 or 4 motion vectors. MPEG-4 defines 1 or 4 motion vectors per inter-coded macro block. Thus we can define a third measure for each frame, the motion vector ratio (MV-ratio):

$$r_{mv} = \frac{n_{MV}}{n_{max,MV}}, \qquad with \quad n_{max,MV} = 4n_{MB} \qquad (3)$$

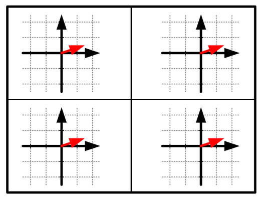
with n_{MV} being the number of non-zero motion vectors in the frame and $n_{max,MV}$ being the maximum possible number of motion vectors in the frame, calculated by 4 times the number of coded macro blocks (n_{MB}). The motion vector ratio provides information about how many of the possible motion vectors are actually used to describe movements between the previous and the current frame and it is therefore also a measure for the amount of motion in the frame.

3.4. Motion Vector Classification

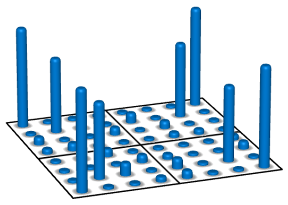
In order to analyse motion vectors quickly we define two classification schemes of the non-zero motion vectors within a frame:

- The Cartesian classification scheme maps each motion vector to one rectangular sector which corresponds to its length and direction.
- The polar coordinate classification scheme maps each motion vector to one polar sector which corresponds solely to its direction.

Figure 1 illustrates both, the Cartesian classification approach as well as the polar coordinate classification approach. The granularity of both approaches may be extended by introducing more detailed scanning directions and multiple motion vector length classes. However, for our approach we observed that 16 different vector classes are sufficient to get reasonable results.



(a) Four quadrants



(b) Potential zoom

Figure 2. Frame histogram of a full frame

Based on this classification schemes we can count the number of motion vectors in each class and build one histogram for each scheme of all motion vectors of a frame. However, for detecting special movements in the frames we should also take into account the origin of each motion vector, since such movements contain motion vectors pointing in opposite directions when observing from the centre of the frame. Therefore we split up the frame into four quadrants resulting in one histogram per quadrant of the frame as shown in figure 2(a). The motion vectors of all macro blocks located in one of the four quadrants of the frame are counted in the corresponding histogram. A resulting histogram of a full frame for a potential zoom movement is shown in figure 2(b). One can see how most of the motion vectors are orientated in a way, so one would suggest that this frame is part of a zoom movement, since nearly all motion vectors are pointing outwards.

There are a couple of basic scene changes and special movements that can be observed in many videos. Figure 3 illustrates some of them. These basic scene changes and movements are not always present as stand-alone movements but rather as a combination of multiple changes. The basic changes and movements considered in this paper are:

a) Scene cuts, which result in different content compared to the previous frame.

b) Scene fades, which are frames providing a transition between two reference frames A and B. Each of the fade frames is a frame $C = x_a \cdot A + x_b \cdot B$ with x_a, x_b being an intensity measure to describe the portion being added from each of the reference frames. $0 \leq x_a, x_b \leq 1$ with $x_a + x_b = 1$.

c) Movements in the form of a zoom, which means that a portion of a frame is magnified or demagnified.

d) Movements in the form of a translation in either x- or y-direction, like a camera turn left or right

e) Movements in the form of a rotation, which may result from rotating the camera or from a rotating object.

4. Scene Change Detection

We developed our scene change detection algorithm by inspecting the previously defined measures and histograms for several video sequences with different genre and content. By analysing the values of the complexity, the intra-ratio, and the motion vector ratio of each frame we got different thresholds for existing scene changes and special movements. In the following sections we describe our scene change detection algorithm for all of the aforementioned scene changes and movements. For all scene changes and movements together we observed that frames with a motion vector ratio below 0.3 do not belong to any of these.

4.1. Scene Cut Detection

In the case of scene cuts the content of the frames before and after the cut differ completely, which is schematically illustrated in figure 3(a). Therefore, a cut within a video is quite easy to detect by inspecting the intra-ratio of a frame as described in equation 2. The first frame of the new scene is normally identified by a high ratio of intra-coded macro blocks. Typical numbers are around $0.50 - 0.60$ but might range up to 0.99. It is important to mention that it is always above the average intra-ratio of previous frames. Additionally the complexity is also much higher than that of an average frame.

For most inter-coded frames in our test videos identifying a cut with an intra-ratio of more than 0.5 and a complexity of more than 0.04 was very promising. However, there were a couple of cut-related frames with a much higher complexity but slightly lower intra-ratio. For this purpose it has been found that allowing an intra-ratio of 0.4 and a complexity of more than 0.08 identifies the remaining cuts. Therefore two thresholds are being proposed, which each frame's parameters have to be above, to be considered a cut. Additionally, we observed a motion vector ratio higher than 0.95 in almost all frames of scene cuts. Only in a few situations this was not true but we observed that the motion vector ratio was still at least 40% above the average ratio calculated for the previous five frames. Thus, the detection of scene changes can be summarized as:

$$\begin{aligned} isCut = &((c > 0.04 \wedge r_{INTRA} > 0.5) \vee \\ &(c > 0.08 \wedge r_{INTRA} > 0.4)) \quad \wedge \\ &(r_{mv} > 0.95 \vee r_{mv} > 1.4\, r_{mv,avg}) \end{aligned} \tag{4}$$

with c, r_{INTRA} and r_{mv} as described in section 3 and the average MV-ratio $r_{mv,avg}$ of the previous five frames.

4.2. Fade Detection

Fades are transitions within a video that stretch over a couple of frames. A fade can basically be described as the

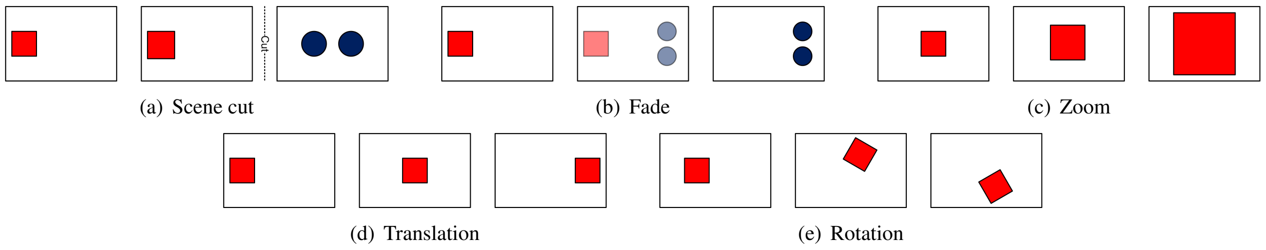


Figure 3. Different scene changes and special movements

addition of two images, namely the source and target images, with different factors. While the first image shows 100% of the source image, following images show a higher and higher percentage of the target image intensity-wise until reaching 100% of the target image. This case is schematically illustrated in figure 3(b). The content of the source and target image, however, might be a moving image itself, which makes it more complicated to define a fade as such.

Observations showed that, compared to average parameters of the whole video, in situations of a fade the motion vector ratio is above average values and the complexity is not that high, if the source and target videos are still images and do not move much themselves. The only motion resulting is therefore the transition between the images. It has also been observed that a typical pattern in the described parameters as well as in the motion vector histograms can be observed. Nevertheless, due to the nature of natural video this pattern is not continuous over the period of the whole fade. Typical fades last between 10 and 50 frames, and identifying patterns such as a high MV-ratio and medium complexity can only be observed in the majority but not in all of the frames. As the detection needs to be done on a frame basis, fades are therefore quite hard to detect.

In most frames the complexity is around the range of 0.01 to 0.16. This is one of the main reasons which makes it difficult to come to a qualified decision. Naturally the complexity changes with increased quantization factors as well, so the values shown here are only valid for the range of quantization factors between 1 and around 5. However, it is possible to reduce the threshold in the same way as for any other complexity measure, which is decreasing with increasing quantization factors. Additionally, the MV-ratio as well as the intra-ratio provide some additional hints for fade detection. The MV-ratio is typically around 0.9 but not lower than 0.8 and the intra-ratio ranges from 0.1 to 0.4 for frames belonging to a fade. The best results could be achieved by using the following equation for fade detection with c, r_{INTRA} and r_{mv} as described in section 3.

$$\begin{aligned} isFade =& (c + r_{INTRA}) > 0.22 \wedge \\ & r_{INTRA} > 0.15 \ \wedge \ r_{mv} > 0.8 \end{aligned} \tag{5}$$

Observations in some of our test videos showed that some special movements such as explosions or sparkling bubbles in a water tank are also detected as fades. Nevertheless, in both cases we have similar divergent movements within the video frames as in the case of fading from one scene to another.

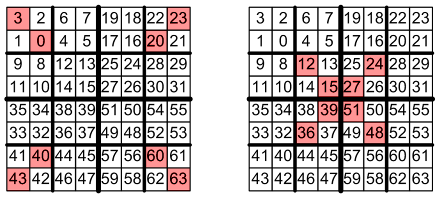
4.3. Zoom Detection

A zoom is a camera operation that enlarges an area of an image, as illustrated in figure 3(c), or in the opposite direction, gets more information about the area surrounding the original image, where the original image is a subset of the zoomed image. Thus, we can assume that most of the motion vectors of a frame which belongs to a zoom are pointing outwards or inwards. In order to detect such a zoom we use the motion vector histograms and count the number of motion vectors in classes of outwards or inwards pointing vectors respectively.

Figures 4(a) and 4(b) each show the 8 zoom indicative classes of outwards and inwards pointing vectors in the Cartesian histogram which we used to detect zooms. For the polar coordinate histogram we have 12 classes for each case. If the motion vector ratio is above 0.4 we consider these vectors pointing outwards or inwards to be zoom indicative. If the number of these zoom indicative motion vectors is more than 30% higher than the expected value for both the Cartesian and the polar coordinate histogram or if it is 100% higher than the expected value for one of the histograms, then we consider the scene movement a zoom. The expected value can hereby be calculated as $\frac{n_{classes}}{64} n_{MV}$ with the number of zoom indicative classes $n_{classes}$, the number of all classes being 64 and the number of non zero motion vectors n_{MV}.

4.4. Translation Detection

Translations are simple camera movements within a video. In this specific case they are considered to be one directional movements in either x or y direction. Those movements can be observed very easily using motion vector histograms, and are characterized by a very high motion vector ratio ranging around 0.9 or even higher. This is due to the fact that most blocks can be referenced easily within the previous frame as the whole content just moves slightly.



(a) Outwards (b) Inwards

Figure 4. MV classes for zoom detection

The fact that it does not reach 1.0 is easily explained: New content which previously was not visible enters the scene with every following frame, whose content cannot be found in the previous frame. Those blocks need to be encoded without motion vectors resulting in a slightly lower motion vector ratio.

4.5. Rotation Detection

Rotation is a movement that is predominantly directed clock-wise or anti-clockwise. Mathematical vector analysis would provide a tool to determine the rotation of a vector field. However, our proposed method of analysing motion vector histograms is capable of providing similar results especially for small video resolutions. The approach is similar to the one for detecting a zoom movement but is solely focussing on those motion vector classes that indicate clockwise or anti-clockwise motion.

5. Evaluation

The objective of our scene change detection algorithm is the use of information about scene changes and special movements within a video stream for determining suitable transcoding parameters for automatic video adaptation for mobile devices. In [5] we presented a transcoding architecture which can be used for such video adaptation. Based on this architecture we implemented our presented methods as one transcoding module for our multidimensional transcoder. This module analyses each frame by building the presented motion vector histograms and computing the aforementioned measures. Based on this information it decides for each frame whether it belongs to a scene change or to a special movement in the video stream. This decision is sent to the controller which can react by determining the according transcoding parameters.

For the evaluation of our scene detection algorithm we selected video sequences which differ from those we used for the development of this algorithm. The selected sequences differ in spatial resolution as well as in the amount and type of motion within the sequences. We used two movie trailers with a high number of cuts and fades, a sequence from TV news with a low number of cuts, fades and zooms and a sequence from a soccer game with only a few cuts and fades but a higher number of zooms. The spatial resolution ranges from 320×240 pixels to 1280×720 pixels and the duration of each sequence was 90 seconds. All evaluation sequences were encoded by using a slightly modified version of the MPEG-4 codec included in FFmpeg. FFmpeg has a rudimentary build-in scene change detection algorithm which we disabled to prevent the use of I-Frames.

The scene changes and special movements which we considered in our work are idealized and simplified. In existing video sequences there are often combinations of such basic scene changes or they cannot be clearly identified by the viewer. Therefore we concentrated our evaluation solely on scene cuts, zooms and fades. Table 1 shows the results of our evaluation. For each of the test videos the number of existing, detected and falsely detected scene changes and movements are shown. The news sequence, for instance, contained 14 scene cuts, from which 13 were detected by our algorithm, and in two cases our algorithm detected a cut although no scene change was present. All existing fades and zooms were detected correctly and there were a few single frames which were falsely detected as such movements. However, assuming that fades and zooms last for at least two or three frames such detection of single frames which belong to a fade or zoom can be easily eliminated and therefore are not shown in the table.

Video	Cut	Fade	Zoom
	existing/detected/false positives		
news	14/13/2	1/1/2	2/2/0
soccer	4/4/1	5/5/3	9/7/0
movie-1	51/48/1	8/7/4	3/3/0
movie-2	49/38/11	14/13/2	7/7/0

Table 1. Evaluation Results

Although there are some false positive detections for cuts and fades, the results are very promising. Moreover, most of the falsely detected frames can be explained quite easily, because they either belong to one of the two other types of scene cuts and movements or they belong to another special movement not covered by our algorithm. For instance 9 out of 11 falsely detected scene cuts of the movie-2 sequence actually belong to a fast fade and in 2 frames the colour of the background changes from one frame to another which could also be interpreted as a cut. On the other hand, 8 of the undetected scene cuts in movie-2 were detected as a fade and only 3 frames were not detected as any type of scene change. Another example are the 3 falsely detected fades in the soccer sequence. They result from a high amount of movement of the background whereas the moving person in the foreground remains in the focus of the camera and therefore relatively fixed in the middle of the frame.

However, the objective of our work was to provide a method for detection of scene changes and special movements to be used for automatic video adaptation. For this purpose the most important information is that a frame does belong to such movement or scene change and the type of special movement is somehow secondary. At the bottom line it can be stated that the presented algorithm does not always detect the type of movement or scene change correctly but the existence of such a special movement is detected quite reliably. The average processing time per frame of our proposed scene change detection algorithm ranges between 0.6 and 6.5 ms for the test videos which is about 18% to 33% of the processing time needed for parsing the video bit stream.

6. Conclusion & Future Work

In this paper we have presented a fast frame-based algorithm for compressed domain scene change detection in MPEG-4 video streams. The presented algorithm analyses the DCT values and motion vectors of each frame in order to decide whether this frame belongs to a scene change or special movement. For this decision we defined three different measures as well as two different motion vector classification schemes which are used by our algorithm. The easy computation of the used metrics and histograms makes the whole scene change detection algorithm very fast which is very important for real time video processing.

The highly statistical nature of the scene change detection results in a statement that by design cannot be absolutely correct. Therefore the scene change detection can only be a tool to indicate a stronger or weaker correlation towards one or the other scene movement. However, the evaluation results show that the algorithm detects a high number of scene changes and movements. Based on these promising results we are currently developing a method to use the information about detected scene changes and special movements within a video stream for determining suitable transcoding parameters. This method will be integrated in our previously presented multidimensional transcoder for automatic video adaptation in the compressed domain.

References

[1] F. Arman, A. Hsu, and M. Chiu. Image processing on compressed data for large video databases. In *ACM Multimedia Conference*, pages 267–272, 1993.

[2] J. Bescos. Real-time shot change detection over online MPEG-2 video. *IEEE Trans. on Circuits and Systems for Video Technology*, 14(4):475–484, April 2004.

[3] J. S. Boreczky and L. A. Rowe. Comparison of video shot boundary detection techniques. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 170–179, 1996.

[4] P. Bouthemy, M. Gelgon, and F. Ganansia. A unified approach to shot change detection and camera motion characterization. *IEEE Trans. on Circuits and Systems for Video Technology*, 9(7):1030–1044, Oct 1999.

[5] J. Brandt and L. Wolf. Multidimensional Transcoding for Adaptive Video Streaming. In *Proceedings of the 17th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'07)*, Urbana, Illinois, June 2007.

[6] Z. Cernekova, C. Nikou, and I. Pitas. Shot detection in video sequences using entropy-based metrics. In *International Conference on Image Processing*, volume 3, pages III–421–III–424 vol.3, June 2002.

[7] C. Dorai and V. Kobla. Generating Motion Descriptors from MPEG-2 Compressed HDTV Video for Content-Based Annotation and Retrieval. In *IEEE Third Workshop on Multimedia Signal Processing (MMSP)*, Sept. 1999.

[8] R. Fablet and P. Bouthemy. Motion Recognition Using Non-parametric Image Motion Models Estimated from Temporal and Multiscale Co-Ocurrence Statistics. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 25(12):1619–1624, Dec. 2003.

[9] J. M. Gauch, S. Gauch, S. Bouix, and X. Zhu. Real Time Video Scene Detection and Classification. *Information Processing and Management*, 1999.

[10] J. Han, D. Farin, P. H. de With, and W. Lao. Automatic Tracking Method for Sports Video Analysis. In *Int. Symposium on Information Theory in the Benelux*, pages 309–316, May 2005.

[11] R. Jin, Y. Qi, and A. Hauptmann. A Probabilistic Model for Camera Zoom Detection. *16th International Conference on Pattern Recognition*, 3:859–862 vol.3, 2002.

[12] R. Joyce and B. Liu. Temporal segmentation of video using frame and histogram space. *IEEE Trans. on Multimedia*, 8(1):130–140, Feb. 2006.

[13] D. Lelescu and D. Schonfeld. Statistical sequential analysis for real-time video scene change detection on compressed multimedia bitstream. *IEEE Trans. on Multimedia*, 5(1):106–117, March 2003.

[14] J. Meng, Y. Juan, and S. Chang. Scene change detection in a MPEG compressed video sequence. In *Storage and Retrieval for Image and Video Databases (SPIE)*, volume 2419, pages 14–25, 1995.

[15] S.-C. Pei and Y.-Z. Chou. Effective wipe detection in mpeg compressed video using macro block type information. *Multimedia, IEEE Transactions on*, 4(3):309–319, 2002.

[16] X. Qian, G. Liu, and R. Su. Effective fades and flashlight detection based on accumulating histogram difference. *IEEE Trans. on Circuits and Systems for Video Technology*, 16(10):1245–1258, Oct. 2006.

[17] I. Sethi and N. Patel. A statistical approach to scene change detection. In *Storage and Retrieval for Image and Video Databases (SPIE)*, volume 2420, pages 329–338, 1995.

[18] K. Shen and J. Delp. A fast algorithm for video parsing using MPEG compressed sequences. In *IEEE Internatuional Conference on Image Processing*, pages 252–255, 1995.

[19] M. S. Toller, P. H. Lewis, and M. S. Nixon. Video Segmentation using Combined Cues. In *Storage and Retrieval for Image and Video Databases (SPIE)*, pages 414–425, Dec. 1997.