Fast Segmentation of H.264/AVC Bitstreams for On-Demand Video Summarization

Klaus Schöffmann and Laszlo Böszörmenyi

Department of Information Technology (ITEC), University of Klagenfurt
Universitätsstr. 65-67, 9020 Klagenfurt, Austria
{ks,laszlo}@itec.uni-klu.ac.at

Abstract. Video summarization methods need fast segmentation of a video into smaller units as a first step, especially if used in an on-demand fashion. We propose an efficient segmentation algorithm for H.264/AVC bitstreams that is able to segment a video in appr. 10% of the time required to decode the video. This is possible because our approach uses features available after entropy-decoding (which is the very first stage of the decoding process) only. More precisely, we use a combination of two features, especially appropriate to H.264/AVC, with different characteristics in order to decide if a new segment starts or not: (1) L1-Distance based partition histograms and (2) ratio of intra-coded macroblocks on a per-frame basis. Our results show that this approach performs well and works for several different encoders used in practice today.

Key words: video segmentation, video analysis, shot detection, shot boundary detection, cut detection, H.264, AVC, compressed domain

1 Introduction

Due to many improvements in several areas of computing, the amount of digital videos has been dramatically increased in recent years. It is often not easy for users to decide whether they are interested in one particular video or not. To solve this problem researchers have started to work on video summarization methods. The goal of video summarization is to create abstracts of digital videos. Many articles have been published on this topic in recent years. While the proposed solutions differ in the methods used to create the abstracts, most of them are based on an elementary video segmentation, where a whole video sequence is divided into small units, called *shots*. Although there has been intensive research on shot-detection methods in the last two decades and this topic is considered to be a solved problem, it seems to reobtain importance with video summarization. Since practically every video is stored in compressed form and video coding standards have reached a very high level of compression efficiency, decoding a video can be a time-consuming process. With applications where on-demand video abstracts should be created, a user might not accept a long delay since he/she wants to see the abstract as soon as possible. In such cases it is obviously a better solution for video summarization methods to rely on

shot-detection methods which can operate directly in the compressed domain in order to enable fast processing, although they might not be as precise as algorithms working in the decompressed domain. Even though there have been many proposed algorithms for performing video shot segmentation in the compressed domain, only a few of them are suited for the most recent state-of-the-art video coding standard H.264/AVC [3], frequently used in practice due to its high compression efficiency.

With this article, we propose a novel shot detection algorithm for H.264/AVC bitstreams that completely operates in the entropy-decoding stage, which is the very first one in the entire decoding process. As shown in Fig. 1, our shot-detection approach is composed of two main processes and uses some further refinement steps. The details of these post-processing steps are not described in this article, but can be found in [10]. Our algorithm allows segmentation of a video in about 10% of the time required to decode the video. This makes it perfectly suited for the usage in the video summarization area. Our approach has been evaluated with the baseline profile of H.264/AVC since we used our own decoder for evaluation which currently supports baseline profile only, but we expect similar results for higher profiles. In the further sections we show that our algorithm works for three very popular encoders with quite equal results and uses simple threshold parameters whose settings can be easily understood by a non-expert end-user as well.

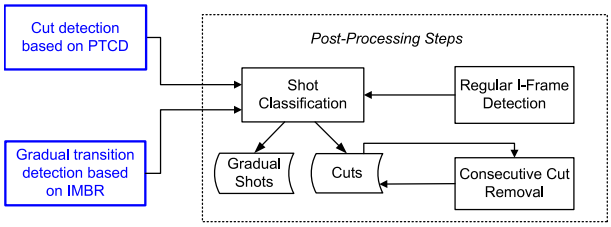

Fig. 1. Overview of our shot-detection approach

2 Shot Detection in H.264/AVC

2.1 Macroblock Types and Partitions

The H.264/AVC standard [3] allows encoding different *types of macroblocks* within a particular type of frame. For instance, a predicted (P) frame may consist of intra-coded macroblocks (I-MBs), predicted macroblocks (P-MBs), bidirectional predicted macroblocks (B-MBs), and skipped macroblocks (S-MBs). Macroblocks are further partitioned into smaller blocks whereby the size of a partition can be: 16x16 Pixel, 16x8 Pixel, 8x16 Pixel, 8x8 Pixel, 8x4 Pixel, 4x8 Pixel, or 4x4 Pixel. Which *macroblock type* and which *partitioning* is applied to

a specific block, is decided by the encoder. Which type of macroblock an encoder chooses mainly depends on the residual. The encoder tries to find the corresponding block(s) in a previous or future frame which will give the smallest residual. If even the smallest residual is too high, the encoder will use intra-coding. Which partitioning is chosen for a macroblock mainly depends on the strategy of the encoder. However, an encoder will typically apply partitioning depending on the "*noise/energy*" of a block, the shapes and contours in the block, the location of "*homogeneous*" areas in the block. The chosen type of a macroblock does also constraint the partitioning[1]. An encoder will use such a partitioning that will result in the lowest residual for the entire block. More details about H.264/AVC can be found in [3], [9], and [11].

2.2 Related Work

In difference to algorithms operating on decompressed data, video shot detection algorithms working in the compressed domain have the advantage that no full decoding (which costs time and memory space) is necessary. Several shot detection algorithms operating on compressed videos have been proposed in the last decade. However, only a few of them use features newly introduced by the H.264/AVC coding standard.

Usually, the performance of a shot detection algorithm is measured in *Recall* and *Precision*. Recall denotes the proportion of the number of detected shots to the number of existing shots. Precision quantifies how many of the detected shots are correct shots. In many cases we are also interested in the correct detection of the starting and the ending location of a shot, nevertheless in this article we use recall and precision as the only metric of evaluation. The reason for this is that this is the most comparable metric since many other published shot-detection algorithms use it as well. We wished to compare the performance of our algorithm to the already proposed algorithms, but the test-videos are, unfortunately, not available for public use[2].

In [5] the authors proposed a shot detection approach for H.264/AVC which is based on the prediction modes of macroblocks. Their approach works in two stages. First, the *dissimilarity distance* of inter-prediction histograms in two adjacent I frames is calculated. If this distance is above a predefined threshold, the GOP starting with the corresponding I frame is further analyzed. This first stage is used to limit the number of GOPs for the second stage. In the second stage, they use *Hidden Markov Models* (HMM) to detect shots based on seven types of inter-prediction of 4x4 partitions. If used with both stages, their approach is able to detect up to 94.7% of the shots with a precision of 98.16%. If only the second stage is used, the precision decreased down to 88.7%. However, their approach takes some assumptions which may not hold in practice.

[1] I-MBs may only be composed of one 16x16/four 8x8/sixteen 4x4 partitions; S-MBs may only be composed of one 16x16 partition

[2] We use this opportunity to raise a plea for unrestricted publication of material needed for reconstruct any published experiments. This is a basic req. for scientific work.

First, many H.264/AVC encoders with default settings do only sparsely insert I frames into the bitstream. Instead of an I frame rather, a P or B frame with many I macroblocks is inserted. This means, that the first stage of the described approach would miss many shots if they do not start with an I frame, and the whole sequence must be analyzed with the technique of the second stage. Next, the inter-prediction type of a macroblock is strongly dependent on the settings of the encoder. For instance, a video encoded with baseline profile does only use backward prediction. The authors of [1] proposed a shot detection approach which is based on SAD values (SAD = Sum of Absolute Differences). The SAD value is calculated by the encoder at encoding time and sums up the Pixel-differences between the current block and the referenced block. With dynamic thresholding their approach achieved a recall of 97% with precision of 97% for a sports video. However, while this is a good result, the proposed approach is only applicable for video bitstreams that include SAD values calculated by the encoder at encoding time. Another shot-detection algorithm for H.264/AVC bitstreams has been presented in [12]. However, their approach separately considers I, P, and B frame coded videos, rather than one single video consisting of several frame types. A detailed summary of related work can be found in [10].

3 IBR Approach

If the residual of a P- or B-MB tends to be too high, an encoder will typically use intra-coding instead of predictive coding for that macroblock. Such situations occur frequently at shot boundaries especially if backward prediction is used. The reason is, that the first frame of a shot is usually very different to earlier frames in the video. Our shot detection algorithm exploits this behavior as it uses the *I-MB Ratio* (IBR) to create a set of candidate-cuts. A candidate frame i is added to the candidate set CS by the following rule, where TH_I is the threshold [0-1] for the required ratio of I-MBs in a frame:

$$\mathrm{CS} = \mathrm{CS} \cup \mathrm{Frame_i}, \mathrm{IBR_i} \geq \mathrm{TH_I} \tag{1}$$

A frame will be added to the candidate set if and only if it contains more I-MBs than the specified threshold (as a percentage value). Although this algorithm successfully detects shot boundaries (e.g. cuts), it needs some further refinement. Those post-processing steps are described in Section 5.

4 PHD Approach

The shot detection process solely based on IBR achieves good performance, as our results show. However, typically the algorithm lacks of accuracy which results in many false positives and, thus, a low precision value. Therefore, we have developed another shot detection algorithm that exploits the partitioning-decisions taken by the encoder.

In Section 2.1 the partitioning of macroblocks has been briefly discussed. Intuitively, an encoder will use uniform partitioning for macroblocks of a slow-changing scene, and differ from that partitioning scheme if a shot-change occurs and the homogenous areas, shapes, contours, noise etc. significantly change. As our results show, this assumption is true for several encoders, thus, the change of the *partitioning scheme* is another well-suited feature for shot detection of H.264/AVC encoded bitstreams. We specify a weighted value, called *Partition Histogram Difference* (PHD), which expresses how much the partitioning scheme changes between two consecutive frames. Therefore, we use a partition histogram consisting of 15 bins according to the macroblock-related partition sizes defined in the H.264/AVC standard (with except to partitions of I-MBs) [3]:

$$H = \{P^{16\times16}, P^{16\times8}, P^{8\times16}, P^{8\times8}, P^{8\times4}, P^{4\times8}, P^{4\times4},\\ B^{16\times16}, B^{16\times8}, B^{8\times16}, B^{8\times8}, B^{8\times4}, B^{4\times8}, B^{4\times4}, S^{16\times16}\}$$

The reason we do not use partition-types of intra-coded macroblocks is that we do not want to detect candidate-cuts in consecutive frames with changing I-MBs. This relies on the observation that such I-MBs could have been inserted by the encoder due to fast motion. Furthermore, if from one frame to another many I-MBs have been inserted by the encoder, this fact will influence the number of other partition-types (i.e. S/P/B partitions). Thus, if a frame consists primarily of many I-MBs it cannot consist of many P-,S-, or B-macroblocks. Hence, the number of such partitions will go dramatically down and increase at the next P/B frame, if this frame will contain only a few I-MBs.

The PHD is calculated as the sum of differences of the histogram bins between the current frame i and the preceding frame $i-1$:

$$PHD = \sum_{b=1}^{15} \left(H_i^b - H_{i-1}^b\right) \tag{2}$$

The histogram difference defined in Equation 2 was originally specified as an L_1 *Difference* (i.e. sum of absolute differences). However, as our measurements have shown, absolute differences lead to much more false detections (i.e. lower precision). This correlation is shown in Figure 2. The reason for this behavior is, that there are situations where macroblocks at the same spatial position change their partitioning not due to real content change, but due to compression efficiency decisions of the encoder. Such a situation might occur if an encoder starts to use *Skipped* macroblocks instead of *Predicted* macroblocks. In this case, a sum of absolute differences will produce a high and double-evaluated PHD value, possibly causing false detections. On the other hand, using just differences instead of absolute differences has the problem, that the (e.g. negative) difference of one

[3] In our measurements, we have not used partition types of B-MBs, thus, we use in fact only eight partition-types due to the baseline profile limitation of our decoder.

bin may compensate the (e.g. positive) difference of another bin. However, as we have performed many measurements with many different video files encoded with different encoders and the sum of differences always produced better results than the sum of absolute differences, we can conclude that such situations do rarely happen.

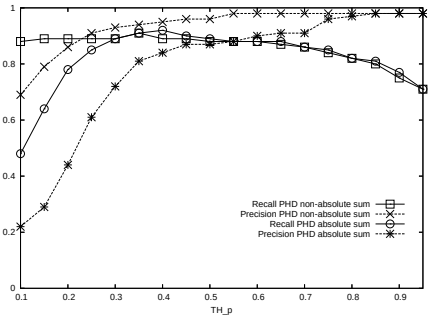

Fig. 2. PHD results when using L1-Distance or sum of differences for video I enc. with x264

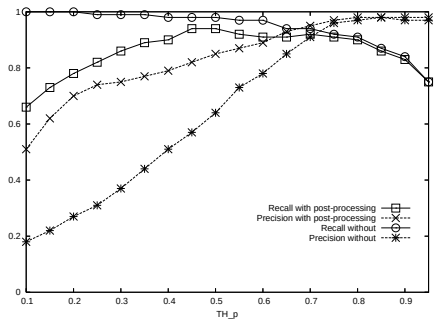

Fig. 3. Influence of post-processing in video I enc. with x264

To enable uniform rating of different partition types, a scaling vector for the histogram bins should be used. We use scaling to 4×4 blocks, thus, our scaling vector looks like this: $S = \{16, 8, 8, 4, 2, 2, 1, 16, 8, 8, 4, 2, 2, 1, 16\}$

Furthermore, we perform a weighting of the sum of scaled differences in order to enable an easier threshold setting. The refinement of Equation 2 including scaling of histogram bins and weighting of the result is shown here:

$$PHD = \frac{1}{W} \sum_{b=1}^{15} \left(H_i^b - H_{i-1}^b \right) S^b \tag{3}$$

The dynamic weight W is specified as the maximum number of 4×4 partitions (i.e. blocks) for frame i and frame $i-1$, whereas intra-coded partitions are not considered, as shown in Equation 4. This will produce PHD values in the range $[0-1]$[4]:

$$W = \frac{2}{16} Framesize - (P_i^{I16\times16} + P_{i-1}^{I16\times16}) S^{I16\times16} - (P_i^{I4} + P_{i-1}^{I4}) S^{I4\times4} \tag{4}$$

A candidate frame i is added to the candidate set CS if and only if the absolute PHD value of frame i is greater than or equal to a predefined threshold TH_P.

[4] $Framesize$ specifies the size of a frame in pixel dimension. It is assumed that this value is constant for the entire duration of a video.

$$CS = CS \ \cup \ Frame_i \ , \ |PHD_i| >= TH_P \tag{5}$$

5 Post-Processing Steps

In the previous two sections we described two simple shot-detection algorithms which rely on different features extracted from compressed data of H.264/AVC bitstreams. The algorithms described so far are quite simple and their general goal is to detect candidate cuts. However, since shot-boundaries are sometimes represented as *gradual transitions* rather than cuts, we need to detect them as one single shot remaining over several frames. Such consecutive candidate cuts are grouped together to one single shot (a *gradual transition*) by the *Shot Classification* Process which is described in [10]. The IBR algorithm described in Section 3 has another problem which needs to be solved: every I frame will be detected as a candidate cut due to its IBR of 100%, even if it has been inserted by the encoder due to Group-of-Picture (GOP) limitations[5]. To avoid this problem our approach removes such frames from the candidate set which seem to be inserted due to the regular end of a GOP. There are several possibilities to detect this correlation, the details are described in [10]. Another post-processing step, called *Consecutive Cut Removal* (CCR), is used to remove such "consecutive" candidate cuts which have not been grouped together by the process of *Shot Classification*.

Figure 3 shows the influence of the post-processing steps on the recall and precision values. For low thresholds the shot classification process will group together a lot of adjacent candidate cuts (which reduces the number of false detections when applying such evaluation rules as described in Section 7) and the consecutive cut removal will further remove many consecutive candidate cuts. This causes both, an increase of precision and a decrease of recall, especially for low threshold settings, as shown in the figure. More details about the post-processing steps can be found in [10].

6 Combining PHD and IBR

Figure 4 shows how the PHD and IBR approach performs if used solely. As shown in the figure, the PHD approach produces much better precision values, especially for low threshold settings, while the IBR approach produces better recall values. As our measurements have shown, the PHD approach works fine for detecting cuts in the video but lacks of the detection of gradual-transition. In contrast, the IBR approach works better for gradual-transition detection but produces many false positives in some specific situations (see [10]). As those observations hold for all of our test videos, we propose a combination of both

[5] Some videos are encoded with a constant GOP size which means that the distance between two adjacent I frames will be usually the same for the entire video. In addition to constant GOP sizes encoders do also manage a maximum GOP size.

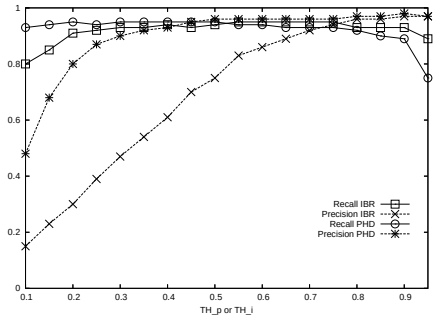

Fig. 4. Comparison of PHD and IBR for video II enc. with QT

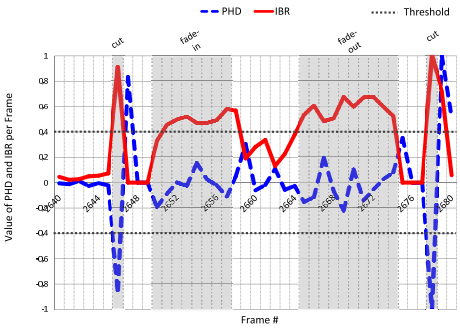

Fig. 5. PHD and IBR applied to frames 2640-2680 of video-I (enc. with x264)

approaches, whereas the IBR approach is used to detect gradual transitions which have not yet been detected by PHD (cuts detected by IBR are discarded). For the example presented in Figure 5, the cuts at frames 2646 and 2678 will be detected by the PHD algorithm while both gradual transitions from 2650-2657 and 2665-2674 will be detected by the IBR algorithm. In this example a threshold of 0.4 has been chosen for both, TH_I and TH_P. The light-gray vertical bars in the figure denote the positions of shot-changes.

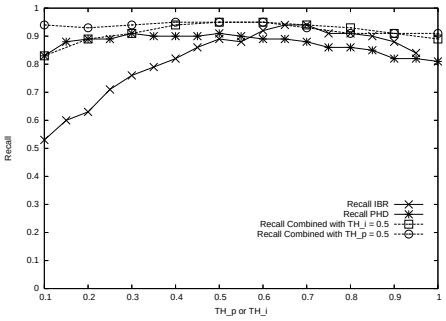

Fig. 6. Recall-Comparison of IBR, PHD, and IBR/PHD combined, video I enc. with Nero

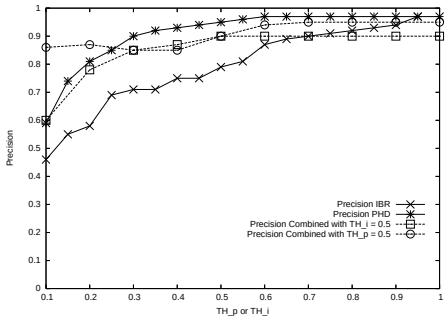

Fig. 7. Precision-Comparison of IBR, PHD, and IBR/PHD combined, video I enc. with Nero

Figures 6 and 7 show the results of solely used IBR and PHD in direct comparison with the IBR/PHD combined approach. The figures show that for most of the threshold values the combination of IBR/PHD produces both, better recall and better precision results. In Figure 8, the Recall/Precision graph of all three methods (IBR, PHD, and IBR/PHD) is presented. Please note, that in this graph lower recall values are not shown, since our results are of weak order

where several entries may have the same rank. It should be further noted, that very high recall values in this graph are avoided by our post-processing steps (as they avoid false-positives).

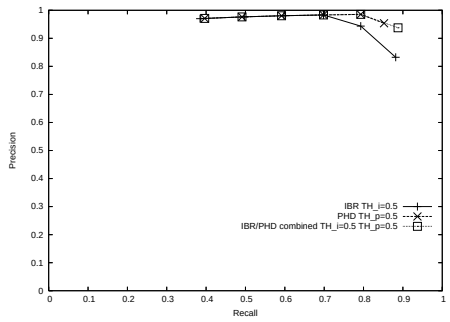

Fig. 8. Recall/Precision Graph for video I enc. with QT

7 Performance Measurements

Our measurements have been performed with several test-videos. In this article we present the results for three videos (more results can be found in [10]):

I. *James Bond - Casino Royale*, Movie Trailer[7]; 320x176 Pixel; 1024 kb/s
3650 Frames; 136 Cuts; 33 Gradual Transitions

II *Sex-and-the-City*, Season-1, Episode-1; 352x288 Pixel; 1024 kb/s
29836 Frames (only the first 20 minutes including intro); 114 Cuts; 117 Gradual Transitions

III. *TRECVID-BOR03*[8]; 320x240 Pixel; 512 kb/s
48451 Frames; 231 Cuts; 11 Gradual Transitions

At least two of the three videos above are publicly available, the first one has been downloaded from the website [7]. The second video is a popular sitcom recorded from TV. The last video has been taken from the public *TREC-2001 Video Track*[8]. For this video the shot-annotation specified by TRECVID has been used. For the other videos the types and locations of shots have been identified by ourselves in a time-consuming manual annotation process. All analyzed videos have been encoded and evaluated with the H.264/AVC *Baseline Profile* by using three popular encoders: (1) Apple QuickTime 7 Pro [2], (2) x264 [4], and (3) Nero Recode 7 [6]. For our implementations and measurements we used our self-implemented H.264/AVC decoder which currently supports baseline profile only. We defined the following rules in order to decide if a shot-change has been correctly detected or not:

– A cut happening at frame i has been correctly detected if our algorithm has detected a shot-change at frame $i \pm 1$.

– A gradual transition happening from frame i to frame $i+n$ has been correctly detected if a shot-change from frame k to frame j has been detected, where either $i \leq k \leq i+n$ or $i \leq j \leq i+n$. In other words, a detected gradual transition must either start or stop within frame i and $i+n$.

Table 1. Best results of the *IBR/PHD combined* approach

video	TH_P / TH_I	**detected cuts/grads**	**false**	**R**	**P**	**average shot detection time in relation to the decoding time**
I (Nero)	0.60 / 0.60	135/25	6	94%	96%	9.54%
II (QT)	0.50 / 0.60	112/108	14	95%	94%	8.41%
III(QT)	0.50 / 0.50	228/3	22	95%	91%	9.17%

Table 1 summarizes the best measurement results of the IBR/PHD combined approach applied to all videos. The combined approach produces better results than any of the single ones (IBR or PHD). For instance, with $TH_P = 0.6$ and $TH_I = 0.6$ the combined approach achieves 94% recall with 96% precision for video-I. With the same threshold value, the IBR-alone approach achieves 92% recall with 87% precision and the PHD-alone approach achieves 89% recall with 97% precision. From our results, we can conclude, that the IBR/PHD combined approach produces well results when both thresholds TH_P and TH_I are within 0.4 and 0.9, whereas 0.5 will be a good start value. This correlation is also shown in Figures 9, 10, 11 and 12, where the performance of the IBR/PHD combined approach with different threshold settings for video I (encoded with Nero Recode and QuickTime) have been summarized.

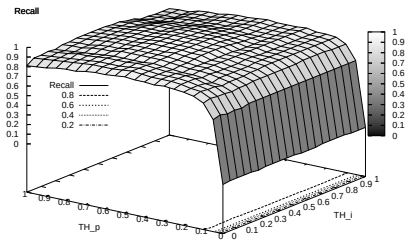

Fig. 9. Recall of the IBR/PHD comb. approach for video I (Nero)

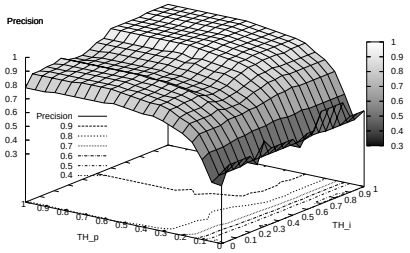

Fig. 10. Precision of the IBR/PHD comb. approach for video I (Nero)

In the last column of Table 1 the results of our measurements regarding runtime performance are given. We used our self-written decoder (implemented

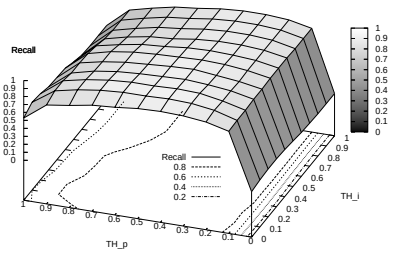

Fig. 11. Recall of the IBR/PHD comb. approach for video I enc. with QT

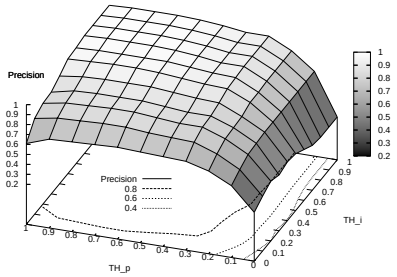

Fig. 12. Precision of the IBR/PHD comb. approach for video I enc. with QT

in C#) for all the measurements, which has currently suboptimal decoding-performance. Since the aim of our program was rather portability and extendibility, it does not use any platform specific optimization. As we can see, our approach requires an average runtime of 9.04% of the overall decoding time for a video. Furthermore, we have directly compared the influence of the chosen encoder for video-I. The results of the combined approach are very similar: while with x264 a recall of 95% with a precision of 95% has been achieved, with QuickTime-7 the algorithm achieved 91% recall with 95% precision and with Nero Recode 7 94% recall with 96% precision. Moreover, we have evaluated the influence of the chosen encoding bitrate. Although not illustrated here due to space limitations (please see [10]), our measurements have shown that IBR is highly dependent on the encoding bitrate and produces the more false alarms the higher is the encoding bitrate i.e. the quality of the video. PHD seems to be more independent from the encoding bitrate. In contrast, the recall of both methods is quite independent from the encoding bitrate although PHD is more resistant against higher bitrates than IBR.

8 Conclusion and Further Work

We have developed a simple and efficient shot detection method for H.264/AVC bitstreams. The algorithm uses information from the entropy decoding process and calculates the difference of partition histograms in combination with the ratio of intra-coded macroblocks. Our method uses a very simple and transparent threshold setting which can be understood by a non-expert user as well. Our approach does not depend on any specific video encoder and has only little dependence on the encoding-bitrate. Finally, our proposed method works fast, i.e. takes only about 10% of the entire decoding time for a video. This is especially advantageous in cases, where no successive decoding is needed. Our proposed algorithm can be used for applications which require very fast segmentation of

a video into shots, as for instance video summarization, video indexing, or shot-based adaptation. The next step is to evaluate our method with higher profiles of H.264/AVC too and to directly compare it with other shot detection methods which work in the compressed domain of H.264/AVC as well.

References

1. A. Dimou, O. Nemethova, and M. Rupp. SCENE CHANGE DETECTION FOR H.264 USING DYNAMIC THRESHOLD TECHNIQUES. In *Proceedings of 5th EURASIP Conference on Speech and Image Processing, Multimedia Communications and Service*, Jul 2005.
2. A. Inc. Quicktime 7 pro, 2007. Online (last accessed on: 04/01/07): `http://www.apple.com/quicktime/pro/`.
3. ISO/IEC JTC 1/SC 29/WG 11. ISO/IEC FDIS 14496-10: Information Technology - Coding of audio-visual objects - Part 10: Advanced Video Coding. March 2003.
4. Laurent Aimar and Loren Merritt and Eric Petit and Min Chen and Justin Clay and Mans Rullgard and Radek Czyz and Christian Heine and Alex Izvorski and Alex Wright. x264 - a free h264/avc encoder. Online (last accessed on: 04/01/07): `http://www.videolan.org/developers/x264.html`.
5. Y. Liu, W. Wang, W. Gao, and W. Zeng. A NOVEL COMPRESSED DOMAIN SHOT SEGMENTATION ALGORITHM on H.264/AVC. In *ICIP '04. 2004 International Conference on Image Processing*, volume 4, pages 2235–2238, Oct 2004.
6. G. Nero Inc. Nero recode 7, 2007. Online (last accessed on: 04/01/07): `http://www.nero.com/eng/index.html`.
7. J. Neuveglise. PocketMovies.net.
8. N. N. I. of Standards and Technology. Trec video retrieval evaluation. Online (last accessed on: 04/01/07): `http://www-nlpir.nist.gov/projects/trecvid/`.
9. I. Richardson. *H.264 and MPEG-4 Video Compression: Video Coding for Next Generation Multimedia*. John Wiley & Sons Ltd., The Atrium. Southern Gate, Chichester, West Sussex PO19 8SQ, England, 2003.
10. K. Schöffmann and L. Böszörmenyi. Early Stage Shot Detection for H.264/AVC Bitstreams. Technical Report TR/ITEC/07/2.04, Department of Information Technology, University of Klagenfurt, Austria, Jul 2007. Online: `http://www-itec.uni-klu.ac.at/~klschoef/papers/shotdetection.pdf`.
11. T. Wiegand, G. J. Sullvian, G. Bjontegaard, and A. Luthra. Overview of the H.264/AVC Video Coding Standard. In *IEEE Transactions on Circuits and Systems for Video Technology*, Jul 2003.
12. W. Zeng and W. Gao. Shot change detection on H.264/AVC compressed video. In *ISCAS 2005. IEEE International Symposium on Circuits and Systems*, volume 4, pages 3459–3462, May 2005.