FAST MULTI-FRAME REFERENCE VIDEO ENCODING WITH KEY FRAMES

Nukhet Ozbek[1] and A.Murat Tekalp[2,3]

[1] International Computer Institute, Ege University, Izmir, Turkey
[2] College of Engineering, Koc University, Istanbul, Turkey
[3] Department of Electrical Engineering, University of Rochester, Rochester, NY 14627

ABSTRACT

Enhanced reference picture selection in H.264 enables increasing compression efficiency at the expense of increasing complexity, in other words encoding time. We search how we can select the best multiple reference pictures by a fast, computationally efficient method. We propose a simple histogram-similarity based method for selecting the best set of multiple reference pictures. Out-of-order coding of these frames is implemented by means of pyramid encoding. Experimental results show that the proposed approach can provide encoding time saving up to 22% with similar picture quality and bitrate for selected video sequences.

1. INTRODUCTION

For complexity/speed optimization of H.264 encoding, several algorithms have been proposed for the case of single-reference frame motion compensation up to now [1-3]. The H.264 syntax also supports multiple reference frames so that more than one previously coded pictures can be used as reference for motion-compensated prediction. This is an important feature of the standard since it can provide considerable gain in terms of compression efficiency [4]. However, increased compression efficiency generally comes at the expense of dramatic increases in computational complexity. We hereby propose a new method that enables fast selection of the best multiple reference frames for H.264 video encoding.

The concept of B-pictures is generalized in H.264 to B-slices, which employ two distinct lists of reference pictures, *list 0* and *list 1*, containing short term and long term (LT) pictures. The default index order, based on picture order count (poc), of the pictures is as follows: *list 0*, starts with the closest past picture and is followed by other past pictures with decreasing poc, and then future pictures with increasing poc; *list 1*, starts with the closest future picture and is followed by other future pictures with increasing poc, and then past pictures with decreasing poc.

Unlike previous standards, the best reference is chosen on a macroblock basis. Macroblock mode decision and motion estimation are the most computationally expensive processes in H.264. RD optimized mode decision process calls for calculation of bitrate and distortion for each option by actually encoding and decoding the video. Brute force approach to obtain the best RD performance requires, for each macroblock, motion search to be done by considering all frames in the reference lists.

Key frames, which are the most representative frames for a video shot, are widely used in video summarization, indexing and retrieval [5]. We propose using key frame selection methods for fast selection of the best multiple reference frames for each group of pictures (GoP). To this effect GoP boundaries can be defined to match video shots, which can be determined by standard shot boundary detection methods [5,6]. In extracting key frames for each shot, an important issue is to determine an appropriate number of key frames to well represent the shot content. Existing approaches for key frame selection tend to be either cluster-based or sequential-based methods using some visual similarity measure [6].

In this paper, we present a computationally efficient, cluster-based method for key-frame extraction using a histogram similarity measure, and an alternative way of buffer management in H.264 by keeping the key references in the Decoded Picture Buffer (DPB) and adding them into *list0* and *list1* when needed. The performance results are drawn by comparing our proposed method and the H.264 reference software (JM 9.2) under the same conditions. The paper is organized as follows: In Section 2, key frame selection method is reviewed. How we changed the coding order for the key frames is discussed in Section 3. Experimental results are presented in Section 4, and conclusions are drawn in Section 5.

2. KEY FRAME SELECTION

Our key frame selection method follows the clustering approach in [5], where color histogram similarity is

employed as a measure. Shots with similar scene content are clustered together. Given shot boundaries, the mean color histogram of each cluster is calculated. The frame whose histogram has the minimum distance to the mean histogram is selected as the main key frame for that cluster. Since the proposed method use histogram bins (totally 256, due to 8-bit pixel values) for subtraction in similarity measure, instead of frame-size times pixel-by-pixel difference it is computationally efficient enough. Hence, the key-frame selection cost is negligible in total encoding time.

We used two clips, one with 199 frames (video1) and another with 121 frames (video2), from the movie Troy (640×272, 25 fps) that both include two clusters with two scene changes. The test clips are typical examples of alternating camera angles that switch back and forth between two different scenes. As shown in Figures 1-3, frames prior to scene change 1 and the frames after scene change 2 together correspond to Cluster 1, whereas frames in between two scene changes correspond to Cluster 2.

We define three different test cases for video1 and one case for video2, as follows:

- In the first case, each cluster has the main key frame together with other two frames which have the smallest difference to the mean histogram (See Fig. 1). The main key frame is 78 for cluster1 and 132 for cluster2 here.
- In the second case, the main key frame is along with two frames which have higher difference to the mean histogram such that the one has a smaller poc and the other has a bigger poc than the poc of the main key frame (See Fig. 2).
- In the third case, among last two key frames in the second case the one has lower difference is removed. Namely, pair (78,168) is selected as key frames for cluster1 and pair (84,132) is selected as key frames for cluster2.
- For video2, two key frames are selected as the ones have minumum and maximum difference to the mean histogram (See Fig. 3). 24 and 60 are the main key frames here.

Test results for all cases, which are called Test1, Test2, Test3 and Test4, respectively are reported in Section 4.

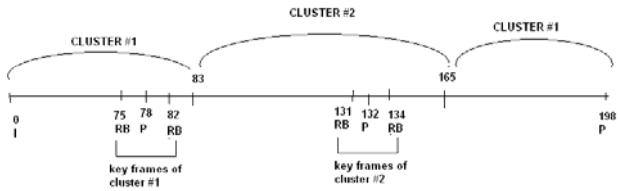

Fig. 1: Key Frame selection for case #1 of video1.

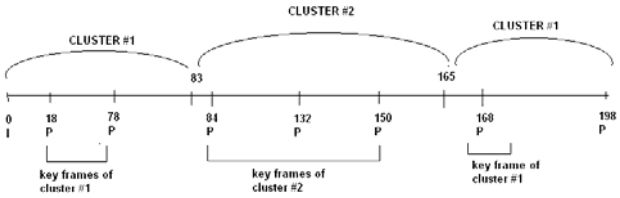

Fig. 2: Key Frame selection for case #2 of video1.

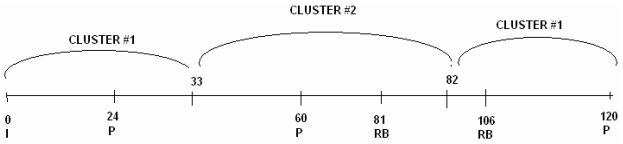

Fig. 3: Key Frame selection for video2.

3. CODING OF KEY FRAMES

Pyramid coding is a non-normative encoder model implemented in JM version 9.2 in order to support more flexible GOP structures. Although it is designed to achieve temporal scalability, it may also be useful for increasing compression efficiency due to its out-of-order coding nature. For our test cases the key frames need to be coded first and placed in the DPB before starting to code other frames. At this stage the pyramid coding serves us very well with the explicit format mode.

We use the same GOP structure for both the JM reference encoder and our modified encoder such that "I-RB-RB-RB-RB-RB-P-RB-RB-RB-RB-RB-P", where RB means reference B picture. For a fair comparison it is necessary to keep encoder parameters the same. The only difference must be identities of the frames in reference lists, for our case key frames always exist in the buffers. Depending on this difference, another difference has to turn out to be coding order. The reason why of different coding order lies under the requirement of that key frames should be coded before their normal orders came, so that they can be used as reference more.

According to three types of key frame selection mentioned in Section 2, three different coding order comes out. Key frame combinations and corresponding coding orders are given in Table 1 and 2 for video1 and video2, respectively. First brackets include Cluster 1's key frames and second brackets have Cluster 2's key frames.

Key Frames	Coding Order
{75,78,82}&{131,132,134}	I0-P78-B75-B82-P132-B131-B134-P6-B1-..
{18,78,168}&{84,132,150}	I0-P18-P78-P168-P84-P132-P150-P6-B1-..
{78,168}&{84,132}	I0-P78-P168-P84-P132-P6-B1-..
{78,168}&{84,132}	I0-P6-B1-..-P60-**P78**-B55-..-B77-**P168**-B79-..-B83-**P84**-P90-..-P114- **P132**-B109-..

Table 1. Key frames for Test1, Test2, Test3, Test3 v2

Key Frames	Coding Order
{24,106}&{60,81}	I0-P24-B106-P60-B81-..
{24,106}&{60,81}	I0-P6-B1-..-P12-**P24**-B7-..-P30-**B106**-B25-..-P48-**P60-B81**-B43-..

Table 2. Key frames for Test4 and Test4 v2

4. RESULTS

We have performed our experiments on a P4 3GHz PC with 1 GB RAM. The first test set is formed by the search range (SR) parameter to be used in motion estimation and its selected values 16, 32 and 64. When SR=32 and SR=64 results are compared, SR=32 outperforms in encoding time whereas bitrates are the same. On the other hand, if SR=16 is used significant time saving is obtained when compared to SR=32 with a negligible bitrate increase. Thus we continue our experiments with SR=16.

It is reported in [7] that if 5 reference pictures are used instead of 1, typical gains could be in 5% range, and for sequences with background that remains similar, gains could be as high as 10%. Being motivated from this point, high number of references is involved in this study. Therefore, (7, 7, 3) and (2, 6, 3) values are selected for the reference buffer triple. The triple is constructed such that the first element means the number of references in P *list0* and the second and third elements mean B *list0* and B *list1*, respectively.

Table 3 and 4 are given to state comparative results of video1 for buffer (7,7,3) and (2,6,3) conditions. Table 5 is given to state comparative results of video2 at buffer (2,6,3) condition. Encoding time, bitrate and luminance SNR values are taken from the output report of the encoder. Time saving is percentage of difference in encoding time calculated as:

Time_saving = [(JM_time – Test#_time)/JM_time] × 100.

Total Rbits, which is abbreviation of Remaining bits, is an important measure in order to analyse bitrate increase in the test cases and calculated as follows:

$$Total\_Rbits = Total\_Bits - \Sigma\, Keyframe\_Bits,$$

where Total_Bits is also taken from the codec output report.

In this study, the same type of buffer management is applied for all tests, which is based on keeping key frames together in the lists for specific intervals. For example in {78,168}&{84,132} case, key frames 78 and 168 continuously exist in the reference lists as long as the frames in between 1 and 83 and the frames in between 166 and 198 are coded. Similarly, key frames 84 and 132 are kept in the lists during the frames in between 85 and 165 should be coded.

Current buffer management style in the JM performs poc-based reordering in B lists and picture number-based in P list. In order to be sure that the key frames are used properly in the reference buffer for other frames, we use long-term reference syntax and MMCO (Memory Management Control Operation) commands. The key frames are marked as long-term just after encoded and added into the lists when needed.

Software	Encoding time(sec)	Bitrate (kbps)	Luma SNR(dB)	Total Rbits	Time saving(%)
JM v9.2	1127	267.15	35.52	2,062,024	-
Test1	956	287.0	35.65	2,082,888	15.17
Test2	975	276.8	35.67	1,888,096	13.50
Test3	958	266.2	35.63	1,875,472	15.00

Table 3. Test results of video1 for reference buffer is (7,7,3)

Software	Encoding time(sec)	Bitrate (kbps)	Luma SNR(dB)	Total Rbits	Time saving(%)
JM v9.2	1005	275.6	35.51	1,933,896	-
Test3	824	273.5	35.63	1,937,968	18.0
Test3 v2	798	267.7	35.62	1,871,136	20.6

Table 4. Test results of video1 for reference buffer is (2,6,3)

When compared to JM results, Test1 results, in which {75,78,82} and {131,132,134} key frame set is used, show that 15.17% time saving is achieved as keeping SNR the same but at the expense of considerable increase in bitrate. Besides bitrate increase there is significant increase in Total Rbits, as well. This means that the key frame combination in Test1 brings no advantage since we target to get time saving without loss in compression efficiency and PSNR.

Test2 gets smaller time saving than Test1 but bitrate is closer to the original one and Total Rbits is even lower. It shows that reference buffer organization like 3 key frames together with 4 recently coded frames is better than the regular one. On the other hand, in selection of key frames, choosing the other two with higher histogram difference (less similar to the main key) makes more sense.

Results show that Test3 outperforms Test2, since it has higher time saving and lower cost for the remaining frames. The reason for this is buffer organization, which is 3 key frames together with 4 recently coded frames for Test2, whereas in Test3 it is 2 key frames together with 5 recently coded frames.

In Table 4, it is seen that Test3 can reach 18% time saving and encode at a smaller bitrate than the original software when the buffer size is set to (2,6,3). Consequently, 2 key frames and 4 recently coded frames together with

{78,168}&{84,132} key references is the best of our experiments up to now.

Software	Encoding time(sec)	Bitrate (kbps)	Luma SNR(dB)	Total Rbits	Time saving(%)
JM v9.2	601	246.7	37.36	1,129,048	-
Test4	488	269.5	37.58	1,138,456	18.8
Test4 v2	471	255.5	37.56	1,108,336	21.6
Test4 v3	476	263.5	37.58	1,109,248	20.8
Test4 v4	471	253.7	37.56	1,099,344	21.6

Table 5. Test results of video2 for reference buffer is (2,6,3)

In most cases, it is observed that although Total Rbits values are smaller, bitrates are bigger than the original. To be able to explain this observation we plot and compare costs of key frames in both the modified and the original software. The comparisons result in that key reference costs are very high in the modified case due to out-of-order way of coding them. The huge distance between key frames and the I-frame causes weak correlation and so high costs.

In order to make any advance, we consider accumulating key frames one by one during encoding instead of coding them at the beginning. Namely, some key frames should be coded when there is a certain distance to the actual order of it. This type of coding orders are given in the last row of Table 1 for video1 and the second row of Table 2 for video2. Performance results of Test3 v2 and Test4 v2 show that while time saving increases, it is observed a considerable decrease in bitrate and Total Rbits. Therefore, by means of accumulating key frames, their bit costs are reduced. They are not allowed to exist in the lists of far frames, so Total Rbits are also reduced.

Searching any further improvement on Test4, we tried another type of buffer management, of which performance results are stated as Test4 v3, and v4. In this scheme, only one key frame can exist in the lists. Pictures of each cluster are ordered according to their distance to the mean histogram and then are grouped into two parts which have equal number of pictures. The main key reference is used for encoding the frames in the part that has smaller distance, and the other key reference is used for encoding the frames in the other part which has bigger distance.

Test4 v3 is single keyframe in the lists version of Test4 (coding keys at the begining) whereas Test4 v4 is the single key version of Test4 v2 (accumulating the keys). For the first case, in both time saving and bitrate an improvement achieved. For the second case, keeping time saving the same, a considerable decrease in bitrate is obtained.

5. CONCLUSIONS

Several algorithms on speed optimization are proposed with single-frame reference in the past. However, only few studies exist with multiple frame references, which propose fast motion estimation by various prediction methods for selecting the initial search point [1, 2, 3].

We selected two or three key frames for different scenes in the test videos according to color histograms, coded them as LT reference and kept them in the DPB till the end of encoding.

Test results show that fast H.264 video encoding with multiple frame references can be achieved at similar quality and bitrate as the brute force reference encoder. We note that the amount of time saving is closely related to how key frames are selected, the number of key frames, which other frames should exist in the reference buffer along with them and also the reference buffer size.

6. REFERENCES

[1] Y. Hsiao, T. Lee, P. Chang, “Short/long-term Vector Prediction in Multi-frame Video Coding System”, in Proceedings of the IEEE Int. Conf. on Image Processing 2004, Singapore, October 2004, pp. 1449-1452.

[2] X. Li, E. Q. Li, Y-K. Chen, “Fast Multi-Frame Motion Estimation Algorithm with Adaptive Search Strategies in H.264”, in Proceedings of the IEEE Int. Conf. on Acoustics, Speech, and Signal Processing 2004, Canada, pp. 369-372.

[3] M-J. Chen, Y-Y. Chiang, H-J. Li, M-C. Chi, “Efficient Multi-Frame Motion Estimation Algorithms for MPEG-4 AVC/JVT/H.264”, in Proceedings of the IEEE Int. Symp. on Circuits and Systems 2004, Canada, pp. 737-740.

[4] M. Flierl, B. Girod, “Generalized B Pictures and the Draft H.264/AVC Video-Compression Standard”, IEEE Trans. on Circuits and Systems for Video Tech., Vol. 13, no. 7, pp. 587-597, July 2003.

[5] M. Ferman, A.M. Tekalp, R. Mehrotra, “Robust Color Histogram Descriptors for Video Segment Retrieval and Identification”, IEEE Trans. on Image Processing, vol. 11, no. 5, May 2002.

[6] Y. Ho, W. Chen, C. Lin, “A Rate-constrained Key-frame Extraction Scheme for Channel-aware Video Streaming”, in Proceedings of the IEEE Int. Conf. on Image Processing 2004, Singapore, October 2004, pp. 613-616.

[7] A. Puri, X. Chen, and A. Luthra, “Video coding using the H.264/MPEG-4 AVC Compression Standard,” Signal Proc.: Image Communication, Elsevier, vol. 19, no. 9, Oct. 2004.