PERMUTATION-BASED LOW-COMPLEXITY ALTERNATE CODING IN MULTI-VIEW H.264/AVC

Daniel Socek, Dubravko Culibrk, Hari Kalva, Oge Marques and Borko Furht

Department of Computer Science and Engineering,
Florida Atlantic University, Boca Raton, FL, USA
[dsocek,dculibrk]@fau.edu, [hari,oge,borko]@cse.fau.edu

ABSTRACT

Low bitrates enabled by the H.264/AVC standard come at the cost of significant decoding complexity. The problem is exacerbated when multi-view video coding (MVC), presently being standardized by the MPEG committee, is considered. Increasing the number of reference views will improve the prediction but also increase the encoding and decoding complexity. In this paper we present an alternative to transform coding and arithmetic coding used to code motion compensation residuals in H.264/AVC. Proposed approach is sorting permutation based, resulting in significantly decreased decoding complexity. We evaluate the efficiency of the coding and present initial results obtained for some sequences in published literature.

1. INTRODUCTION

Rapid development and application of digital video and audio compression technologies has marked the last two decades. Continued decrease in the cost of memory and increase in processing speed enabled the use of progressively more efficient video encoding algorithms. The need to provide interoperability among products developed by different manufacturers and at the same time allowing flexibility for ingenuity in optimizing and molding the technology has given rise to the development of a series of international video coding standards such as H.261, MPEG-1, MPEG-2/H.262, H.263, and MPEG-4 (Part 2).

Among the recent works in this area, the H.264/AVC video encoding standard, also known as MPEG-4 AVC occupies a central place [1]. The H.264 standard, jointly developed by the ITU-T and the MPEG committees, is highly efficient offering perceptually equivalent quality video at about 1/3 to 1/2 of the bitrates offered by the MPEG-2 format. These significant bandwidth savings open the market to new products and services. However, these gains come with a significant increase in encoding and decoding complexity [2].

The development of video coding technologies has, over the last few years, made possible a new generation of video applications. Multi-view video coding (MVC) has been receiving significant attention among researchers [3-6]. Multi-view video coding (MVC) is also being standardized by the MPEG committee [7-8]. The goal of MVC is to allow coding of multiple camera views such that the user has the freedom of choosing the view point. The biggest challenge here is in developing compression technologies that can exploit the redundancies among the multiple views to achieve high compression ratio and allow fast decompression and playback of the selected views at the receiver.

The key components of a H.264 video decoder, currently used as the anchor in the MVC proposals under review by MPEG, are shown in Figure 1. The entropy decoder outputs the header and the quantized transform coefficients. The inverse quantization is followed by the 4x4 or 8x8 inverse transform. The intra or inter prediction is added to the decoded residual. The decoded picture is deblocked and saved in the frame store. The inverse quantization, inverse transform, and deblocking account for a large portion of the decoder complexity in H.264. The use of the context adaptive binary arithmetic coding (CABAC) in H.264 adds to the decoding complexity. A complete overview of the H.264/AVC can be found in [9].

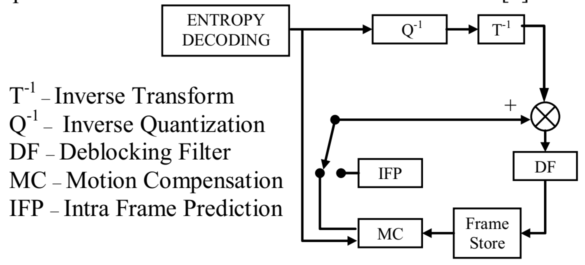

Figure 1 H.264 decoder.

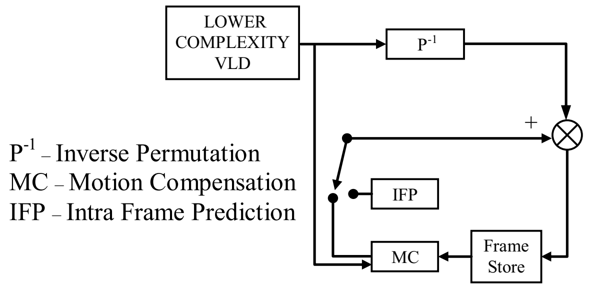

Figure 2 Proposed permutation-based decoder.

This work was supported in part by the Office of Naval Research (ONR) under the Center for Coastline Security Technology grant N00014-05-C-0031.

In this paper we present an alternative to integer transform and arithmetic coding used to code motion compensation residuals in H.264/AVC. Block diagram of the system is shovn in Figure 2. The proposed approach is sorting permutation based. The use of the proposed algorithm reduces the residual decoding step to two table lookup operations and one subtraction for minor part (<40%) of the pixels of the coded macro blocks (MBs) in the residual, followed by a padding ("fill-in") operation to set the value of the majority (>60%) of pixels in these blocks. This significantly decreases decoding complexity of the H.264/AVC decoder, by replacing computationally intensive quantization, transform and arithmetic coding. In addition, our results show that the artifacts introduced by the proposed approach are not block effects and can therefore render the deblocking filter unnecessary. Reduced complexity comes at the price of increased bitrate. However the increase is acceptable for the low quality applications (high H.264 quantization parameter values) that would benefit most from a low complexity decoder.

The lower complexity of the proposed encoder makes it possible to support a larger number of views on a given hardware configuration compared with the H.264 based multi view coding.

The rest of the paper is organized as follows: Section 2 is dedicated to the description of our approach and the analysis of the complexity of decoding. Section 3 presents the experiments conducted and the results obtained. Section 4 holds our conclusions. Section 5 is the list of references used.

2. PERMUTATION-BASED CODING OF MOTION COMPENSATION RESIDUALS

Permutation-based compression techniques were studied in the past for different domains (e.g. digital images [10-11]). We designed a low-complexity permutation-based compressor that outperforms other general purpose compressors when applied to data with a highly Laplacian frequency distribution. By "highly" Laplacian distribution, we mean a Laplacian distribution that has a single peak with more than 60% of the data contained in it. Generally speaking, motion compensation residuals of a multi-view H.264/AVC have such or similar distribution (large percentage of 0's), and hence, are suitable for the permutation-based compression. The following steps describe the proposed permutation-based encoder:

1. Calculate a sorting permutation P of a sequence S of length N and K distinct values.
2. Since P is a sorting permutation, it can be divided into K blocks, so that the permutation indices within each block point to the same value in S. Note that rearranging the elements within a block, as well as rearranging the blocks themselves, simply result in another sorting permutation for the sequence S. Thus, in Step 2, P is to be divided into such K blocks, and elements sorted within each block.
3. Drop the block with the highest number of elements. This block obviously corresponds to the peak of the Laplacian frequency distribution of sequence the S. Notice that if N is known, one can easily "fill-in" the remaining indices that are contained in the largest block.
4. Calculate the differentials of the elements within each remaining block, and apply a standard Huffman encoder. Transmit this along with the histogram of S.

To decode the data, one simply must apply the inverse transformations. First, apply Huffman decoder and invert the differentials to recover the sorting permutation P. Invert P and apply it to the data histogram. Notice that the method has low computational complexity, especially at the decoder side, since inverting P and permuting data is equivalent to indexed table lookup. The use of the proposed algorithm reduces the residual decoding step to two table lookup operations and one subtraction for minor part (<40%) of the pixels of the coded macro blocks (MBs) in the residual, followed by a padding ("fill-in") operation to set the value of the majority (>60%) of pixels in these blocks. Thus, most of the pixels will require just two elementary operations, while less than 40% of them will require 3. An average of 38.4 elementary operations per 4x4 block. In comparison, just the inverse integer transform in H.264 requires 224 elementary operations for each block with non-zero coefficients and 32 elementary operations for blocks that contain all zeros [12].

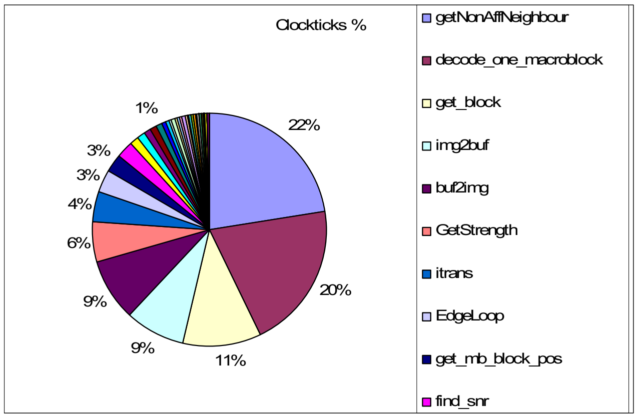

Figure 3 Cycles spent in specific routines of the H.264 reference code.

To evaluate the impact of the proposed method in terms of the complexity of the decoder, the number of cycles spent in routines that would be made obsolete in the reference H.264/AVC code (JM 10.1) has been analyzed. Figure 3 shows the break down of the number of cycles spent in specific routines of the standard. The chart indicates that some 56% of cycles is spent in transform decoding, deblocking and arithmetic coding, which would be rendered unnecessary when the proposed approach is used.

3. EXPERIMANTAL RESULTS

In order to evaluate the performance of our approach, we modified the current implementation of H.264/AVC (JM 10.1) and simulated a multi-view H.264 coder by composing predictions from both temporal and spatial camera views. We performed experiments on two publicly available multi-view sequences, whose camera settings are depicted in Figure 4.

Ballroom (MERL) sequence was filmed with 8 cameras arranged in a horizontal array, and we coded the sequence captured with camera #1 with one temporal view prediction and two spatial view predictions (from cameras #0 and #2). Akko&Kayo sequence was filmed with 15 cameras arranged in a 3×5 array so that we coded the sequence captured with camera #47 with one temporal view and three spatial view predictions (from cameras #27, #46 and #48).

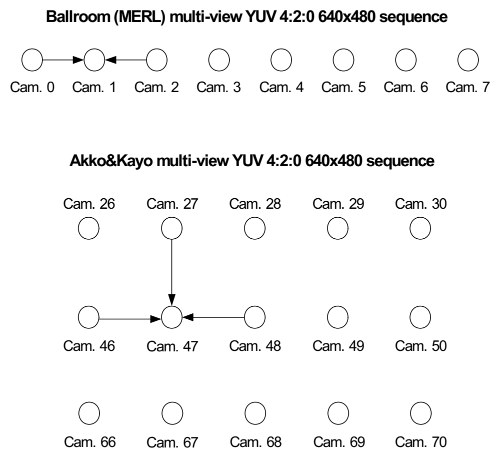

Figure 4 The camera settings of two multi-view sequences: Ballroom sequence (high motion) on the top, and Akko&Kayo (low-to-medium motion) on the bottom.

As expected, preliminary results showed that coding of multi-view sequences produces motion compensation residuals with more Laplacian distribution than that of any of the single camera views alone (temporal view prediction only). However, for our method we needed slightly better Laplacian distribution. Thus, we introduced an *adjustment threshold* (AT) parameter which is a small integer value used to artificially create highly Laplacian sequence by turning all coefficients that are smaller or equal to it (in the absolute value) into 0. For example, if AT=10, all motion compensation residuals that have value in the range [-10, 10] are turned into 0. As a result, all prediction errors greater then 10 (in absolute value) are fixed, and the ones in the range [-10, 10] are not.

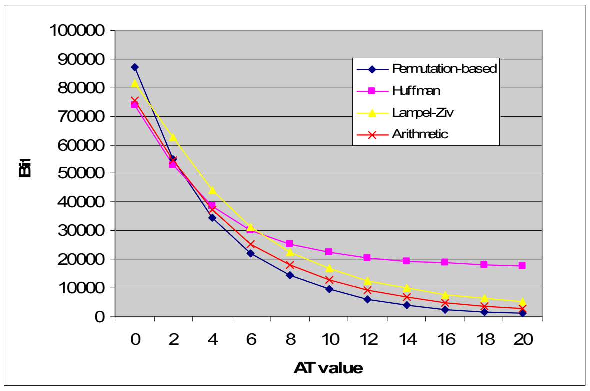

Figure 5 Compression Performance comparson plot for ballroom sequence..

Since the target applications include low-complexity decoders, we selected relatively high quantization parameters for P-slices (QPPSlice) in the H.264/AVC configuration file. In particular, we have used a value of 30 for QPPSlice in coding of the lower motion sequence Akko&Kayo, and a value of 35 for QPPSlice in coding of the high motion sequence Ballroom. Also, we considered only coding of motion compensation residuals for luminance (Y component), assuming that similar results would hold for the chrominance components.

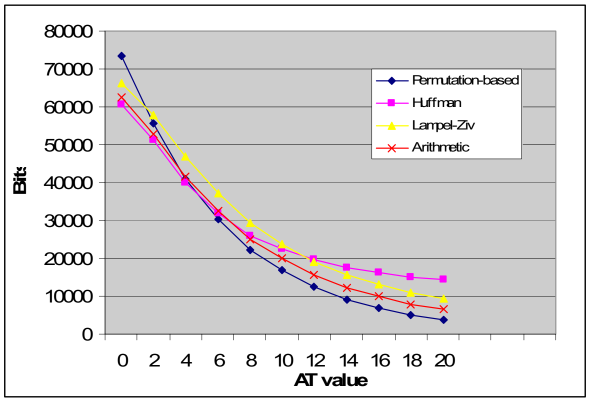

Figure 6 Compression Performance comparson plot for Akko&Kayo sequence.

This process is similar to quantization in H.264 codec but does not result in blocking artifacts. Figures 5 and 6 show comparison of our method as opposed to the general purpose compressors (Huffman, Lampel-Ziv, and Arithmetic coders). Notice that after a certain threshold value, the data becomes highly Laplacian and our method outperforms all other methods in terms of compression ratio. Tables 1 and 2 show performance of the standard H.264 coder on the two sequences, while Tables 3 and 4 show the performance of our approach on the same sequences in terms of MSE, PSNR, and the bit-size of the encoded motion compensation residuals (for Y component only). The results indicate that as AT parameter increases, so does the MSE of the reconstructed frames, but at relatively low rate. At the same time, for higher AT values, the proposed method approaches the performance of standard H.264 encoder in terms of bit-size.

Table 1. H.264 statistics for the Akko & Kayo sequence.

QPPSlice	MSE	PSNR	BITS
20	3.187327	43.09654	74817
25	5.231266	40.94474	14511
30	6.935267	39.72017	3943
35	8.851979	38.66040	1245
40	11.72056	37.44132	468

Table 2. H.264 coder statistics for the ballroom sequence.

QPPSlice	MSE	PSNR	BITS
20	4.469606	41.62811	173776
25	9.195218	38.49518	41382
30	12.65426	37.10844	9656
35	15.37748	36.26195	3678
40	19.49729	35.23106	1573

Table 3. Permutation coder statistics for the Akko&Kayo sequence with multi-view prediction and a base QPPSlice parameter of 30.

AT	MSE	PSNR	BITS
0	6.274971	40.15469	87150
2	6.393887	40.07315	54889
4	6.722376	39.85558	34581
6	7.128421	39.60087	22128
8	7.505329	39.37711	14478
10	7.848965	39.18268	9555
12	8.171735	39.00766	6198
14	8.435472	38.86971	4036
16	8.653115	38.75908	2522
18	8.817894	38.67715	1641
20	8.952979	38.61113	1014

Table 4 Permutation coder statistics for the Ballroom sequence with multi-view prediction and a base QPPSlice parameter of 35.

AT	MSE	PSNR	BITS
0	14.41585	36.54240	73466
2	14.47926	36.52334	55493
4	14.70824	36.45520	40858
6	15.06642	36.35070	30249
8	15.51329	36.22376	22328
10	15.96519	36.09906	16889
12	16.46461	35.96529	12420
14	16.92329	35.84596	9112
16	17.33773	35.74088	6835
18	17.71555	35.64726	5026
20	18.00961	35.57576	3857

Therefore, the choice of AT parameter should depend on the particular application requirements in terms of complexity/bit-size target ratio. In addition, as Figure 7 depicts, our permutation-based approach introduces a slightly different type of visual artifacts (salt-and-pepper noise) than the standard H.264 (blocking artifacts), so that the deblocking filter could be either simplified or removed at the decoder side.

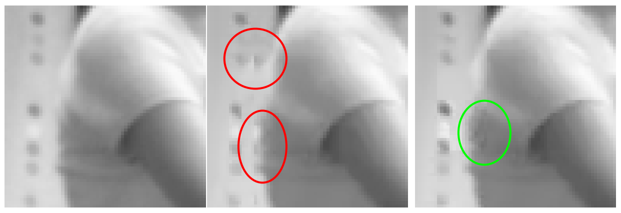

Figure 7 The comparison of visual artifacts on a part of the Akko&Kayo frame for near identical PSNR: (a) the subframe of the original frame, (b) the subframe of the standard H.264 with QPPSlice=35 reconstructed frame showing blocking artifacts due to transforms [PSNR=38.66040], and (c) the subframe of our permutation-based multi-view approach (a reconstructed frame) with QPPSlice=30 and AT=18 showing a salt-and-pepper type of artifacts [PSNR=38.67715].

4. CONCLUSIONS

In this paper we propose an alternate, permutation-based low-complexity coding of motion compensation residuals in a multi-view H.264 codec. Our approach is suitable for use in applications that require low computational complexity at the decoder side. As preliminary experiments suggest, for certain parameter values our method produces slightly larger number of bits representing the motion compensation residuals of Y component than the standard (single-view) H.264 does for the same or similar PSNR, but at lower complexity. We also observed different artifacts in the reconstructed frame (salt-and-pepper artifacts vs. blocking artifacts). Based on our initial observations, we believe that the proposed approach could be more effective when lossless MVC is concerned. Further studies are necessary to fully exploit the benefits of permutation-based approaches to video compression.

5. REFERENCES

[1] ITU-T RECOMMENDATION H.264 "Advanced Video Coding for Generic Audiovisual Services". May 2003.

[2] Implementation Studies Group, "Main Results of the AVC Complexity Analysis," MPEG Document N4964, ISO/IEC JTC11/SC29/WG11, July 2002.

[3] T. Matsuyama, "Exploitation of 3D video technologies," Informatics Research for Development of Knowledge Society Infrastructure, 2004.ICKS 2004.International Conference on, pp. 7-14.

[4] O. Schreer, P. Kauff, and T. Sikora, edts., "3D Video Communications, " Wiley 2005.

[5] M. Ollis and T. Williamson, "The future of 3D video," Computer, vol. 34, pp. 97-99, 2001.

[6] A. Smolic and P. Kauff, "Interactive 3-D video representation and coding technologies," Proceedings of the IEEE, vol. 93, pp. 98-110, 2005.

[7] ISO/IEC JTC1/SC29/WG11, "Call for Proposals on Multi-view Video Coding (MVC)," MPEG Document MPEG2005/N7327, July 2005.

[8] ISO/IEC JTC1/SC29/WG11, "Survey of Algorithms used for Multi-view Video Coding (MVC)," MPEG Document MPEG2005/N6909, January 2005.

[9] T. Wiegand, G. Sullivan, G. Bjontegaard, and A. Luthra. "Overview of the H.264/AVC Video Coding Standard," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, No. 7, July 2003.

[10] Z. Arnavut, "Lossless Compression of Color Mapped Images", Journal of Optical Engineering, vol. 38, no. 6, June 1999, pp.1001-1005.

[11] Z. Arnavut and S. Narulmalani, "Lossless Compression of Multispectral Images Using Permutations", In Proc. of the International Geoscience and Remote Sensing Society Symposium (IGRASS-96), Lincoln, Nebraska, IEEE Press, Piscatway, NJ, 1996, pp. 460-463.

[12] M. Horowitz, A. Joch, F. Kossentini and A. Hallapuro. "H.264/AVC Baseline Profile Decoder Complexity Analysis", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, No. 7, July 2003.