PARALLEL ENCODING - DECODING OPERATION FOR MULTIVIEW VIDEO CODING WITH HIGH CODING EFFICIENCY

Kemal Ugur[1], Hui Liu[2], Jani Lainema[1], Moncef Gabbouj[3], Houqiang Li[2]

[1]: Nokia Research Center, Tampere, Finland
[2]: University of Science and Technology of China, Hefei, China
[3]: Tampere University of Technology, Tampere, Finland

ABSTRACT

Multiview Video Coding (MVC) standardization is an on-going effort aiming to extend H.264/AVC by developing novel tools optimized for 3D and multiview video use cases. One of the key identified requirements for the MVC standard is its ability to support parallel processing of different views. The parallel operation is known to be especially important for 3DTV applications, where the display needs to output many views simultaneously to support head-motion parallax. In this paper, we present a novel coding structure that enables parallel encoder/decoder operation for different views, without compromising from the coding efficiency. This is achieved by systematically restricting the reference area of each view, so that encoding and decoding of macroblocks from different views could be efficiently pipelined and parallel operation of separate views becomes possible. As the inter-view prediction is still used, proposed structure achieves up to 0.9 dB gain compared to simulcast, maintaining very similar desirable parallelism characteristics.

1. INTRODUCTION

With recent advances in capture and display technologies, 3D video communication and entertainment services are becoming a reality and will enter into consumer domain in the near future. To enable these upcoming applications that range from stereoscopic video broadcasting to 3DTV, an efficient Multi-view Video Coding (MVC) standard is necessary. Recently, the Joint Video Team (JVT) comprising VCEG and MPEG standardization groups undertook the effort to standardize an MVC standard by extending H.264/AVC and including novel tools optimized for MVC.

One of the key identified requirements for the MVC standard is its ability to support parallel processing of different views [1]. The parallel processing of different views is especially important for 3DTV use cases, where the displays need to output many views simultaneously to support head-motion parallax. However, inter-view dependencies between pictures may impose serious parallelism issues to the video system, because two pictures at different views need to be decoded sequentially. Let's consider a 3DTV system displaying simultaneously two views, and views are coded with the coding structure as illustrated in Figure-1. In order to decode a picture in view-1 at any temporal instant, the picture in view-0 at the same temporal instant shall be decoded first. Only way to display two views at the same time is by having an MVC decoder running two times faster than a regular single-view decoder. Even though two independent decoders running on different platforms might be available, both decoders need to run twice faster than the single-view decoder because decoding has to be performed sequentially. The situation gets worse, with the increasing number of views that is supported by the 3D display. Currently there are commercial displays which can display 100 views simultaneously, and if all the views depend on each other, then the decoder must run 100 times faster, which is very challenging.

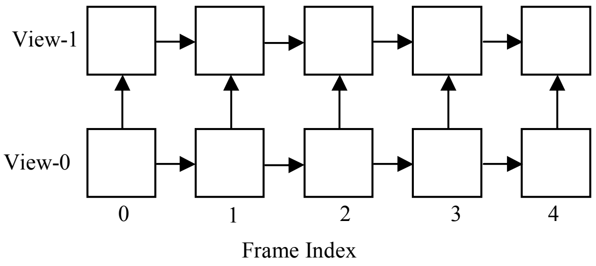

Figure 1. Sample Prediction Structure for Two Views

One way to increase the parallelism is to code each view independently. However, this kind of simulcast approach results in a significant penalty in coding efficiency as inter-view redundancies are not exploited at all. In this paper, we present a coding structure that enables parallel encoder/decoder implementation for different views, even though there are dependencies between views. This is achieved by encoding views with some constraints, so that any macroblock in a certain view is allowed to depend only on reconstruction values of a subset of macroblocks in other views. Resulting bitstreams generated with the proposed structure could be decoded in parallel, because macroblocks belonging to different views could be efficiently pipelined. As the inter-view prediction is still used, significant gain compared to simulcast is achieved, while maintaining almost the same desirable parallelism characteristics. Also, because the parallelism is achieved by encoding constraints, no low-level changes are introduced to the MVC decoder. Experimental results show that, when compared to simulcast approach, proposed system achieves 0.9 dB gain on average with similar parallelism requirements.

This paper is organized as follows: Section 2 provides the details of the proposed coding scheme and details a a parallel decoder operation for two views as an example. Section 3 presents some design considerations and trade-offs possible with the scheme. Section-4 presents the experimental results. Conclusions are presented in Section 5.

2. PARALLEL DECODING OPERATION FOR MVC

In order to better understand the parallel processing requirements for MVC, consider the structure illustrated in Figure-1, where two views are encoded using both temporal and view prediction. The decoding of a frame in view-1 cannot start before all its references are fully decoded and corresponding reconstructions are made available in the memory. This means that, before the decoding of any frame in view-1 could start, both its temporal and view references must be decoded first. Let's assume the decoding time for view-0 and view-1 pictures are τ_0 and τ_1 respectively. Then, the time needed to decode two pictures captured at the same time instant from both views would be $\tau_0 + \tau_1$. Even if there are resources for having separate decoders for each view, time needed to decode views at the same time instant cannot be reduced, as decoders cannot run in parallel, but have to wait for each other. This need to perform sequential decoding in view direction constitutes a serious problem with the increasing number of dependent views, and may become a very challenging task for implementation.

The main concept behind our proposal is to use certain encoder restrictions so that any macroblock in a certain view is allowed to depend only on reconstruction values of a subset of macroblocks in other views. Let's consider a similar example as above, where two pictures from view-1 and view-0 are going to be decoded, and view-1 picture references view-0 picture as illustrated in Figure 2 (for simplicity the size of the frames are five macroblocks both horizontally and vertically). Assume the video is encoded in a way that macroblocks in view-1 picture could only use reconstruction values of macroblocks that belong to certain rows in view-0 picture. For example, the macroblocks in the first row of view-1 picture could only use reconstruction values from the first two macroblock rows in view-0 picture. In other words, the available reference area for the first macroblock row of view-1 picture constitutes only data from the first two macroblock rows of view-0 picture (i.e. the motion vectors for the view-1 macroblocks are restricted). Similarly, the second macroblock row of view-1 picture only uses reconstruction values of the first three macroblock rows of the View-0 picture This systematic restriction of reference area enables parallel decoding of first row of view-1 with any row below the second of view-0, as they are not referring each other (the details of the parallel decoding operation is presented in the next section).

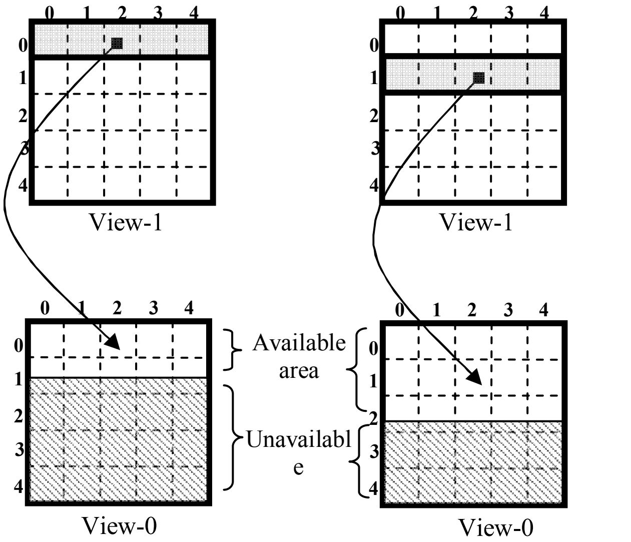

Figure 2. Reference Area Restriction

It should be noted that the available reference area for the first macroblock row in view-1 does not include all the pixels of first two macroblock rows of view-0, but cropped by three pixels from bottom. The reason for cropping is to avoid using pixels that depend on reconstruction values from unavailable reference area, due to deblocking and interpolation. In H.264/AVC interpolation is used to obtain sub-pixel values and is defined using a 6-tap FIR filter. Because of the 6-tap filtering step, some fractional pixels in the second macroblock row would need pixels from third macroblock row in the vertical sub-pixel interpolation process. To avoid this, the available reference area is restricted by cropping three pixels from the boundary. Second reason for cropping is related to the adaptive deblocking process. The deblocking filter in H.264/AVC modifies the reconstruction pixels close to the macroblock boundary by utilizing data from neighboring macroblocks. The number of pixels affected depends on many things such as macroblock mode, but at most three pixels from the macroblock boundary could be changed in deblocking. By cropping the available reference area by three pixels, those pixels that depend on macroblocks from the unavailable reference area are left out. For more details of deblocking filter and sub-pixel interpolation in the H.264/AVC standard, the reader is referred to [2][3]

In order to illustrate the parallel decoding operation, let's assume there are two decoders for each view running on separate platforms. Through proper signaling, the MVC decoder knows which macroblocks of view-0 and view-1 pictures could be independently decoded. Figure-3 illustrates the parallel decoding operation of macroblocks for the bitstream generated with the above example.

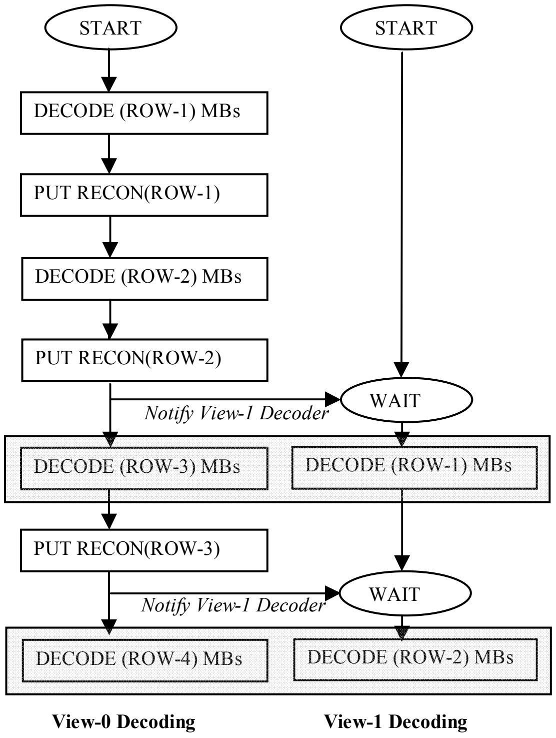

Figure 3. Sample Parallel Decoding Process for Two Views

As shown in Figure-3, decoding for both views start simultaneously, but decoding of first row of macroblocks in view-1 picture does not start before view-0 notifies the view-1 decoder. This notification is done after all the macroblocks in the first two rows in view-0 are decoded, and their reconstruction data are placed in the memory. This notification tells decoder of view-1 that, all data required to decode macroblocks of first row in view-1 are ready. This way, decoder of view-1 could start decoding of macroblocks of first row, while decoder of view-0 proceeds with decoding macroblocks in the third row and two decoders run in parallel (this parallel operation is indicated with a shaded rectangle in Figure-3). This parallel operation continues with two rows of delay between two views till the decoding of all the macroblocks are finished.

3. DESIGN CONSIDERATIONS

In the previous section, we presented encoder restrictions and details of the parallel decoding process. The total decoding time for a parallel implementation as illustrated above could be given as

$$\tau_0 + \mathit{delay\_overhead} + \mathit{synch\_overhead} \qquad (1)$$

where, τ_0 is the time needed to decode the frame for the single-view case. *delay_overhead* is the time spent waiting for the initial macroblock(s) to be decoded, and *synch_overhead* is the time spent in the WAIT states. If the initial delay between views is large (*delay_overhead* is large), coding efficiency of the system would increase because of the increased area available for referencing, with negative impact on parallelism. So, there exists a trade-off between coding efficiency and the parallelism of the multi-view system. Therefore, it is advantageous to allow an encoder to utilize this trade-off efficiently. For this purpose, we use the parameter *initial_delay* to adjust how many delays in macroblock rows exist between views. For the example illustrated in Figure 2, *initial_delay* is given as 2.

4. EXPERIMENTAL RESULTS

In order to test the performance of our scheme, we have followed the test conditions advised by the JVT standardization committee [4]. The proposed scheme was implemented on the Joint Multi-view Test Software (JMVM) [5]. As explained in Section 3, proposed scheme allows trade-offs between coding efficiency and the parallelism that could be adjusted using the parameter *initial_delay*. For this experiment, *initial_delay* is set to 2 (same as the example illustrated in Figure 3). For the simulcast case, where each view is coded independently, each decoder could run at the same speed as the single-view case to output pictures simultaneously, as there are no inter-view dependencies. For coding structures that utilize inter-view redundancy, it was assumed that decoder needs to run N times faster than a single view decoder (N being number of dependent views), to simultaneously output pictures from different views. For the bitstreams generated with the proposed method, there exists a lot less speed requirement. Table 1 compares the decoder speed requirement for the proposed method with different approaches for different sequences. The data for anchor in Table-1, utilizes inter-view prediction structure using hierarchical B as shown in [7].

Sequence	**Anchor**	**Proposed**
	Decoder Speed Requirement	Decoder Speed Requirement
Ballroom	5xsingle	1.3xsingle
Exit	5xsingle	1.3xsingle
Race1	5xsingle	1.3xsingle
Flamenco2	2xsingle	1.03xsingle
BreakDance	5xsingle	1.2xsingle
Rena	8xsingle	1.5xsingle
Uli	5xsingle	1.2xsingle
Akko&Kayo	6xsingle	1.4xsingle

Table 1. Decoding Speed Increase Compared to Single-View Decoder for Three Approaches

The coding results for the proposed scheme are compared against two approaches, simulcast and anchor. The anchor data is generated without any encoder restrictions and simultaneous output of views is possible only by running the decoder N times faster, N being the number of dependent views. Comparison against anchor is useful to see the penalty on the coding efficiency to enable parallel decoding of different views. We also compared our method against simulcast approach. The encoding details for simulcast are exactly the same as the proposed scheme and anchor, only difference being the inter-view prediction is disabled for simulcast. Comparison against simulcast indicates the amount of improvement proposed scheme brings with similar parallelism requirements as simulcast. Table-2 presents the coding efficiency comparison of the proposed approached against anchor and simulcast. The differences in PSNR are calculated using the method described in [6]. It is shown in Table-2 that the proposed scheme achieves up-to 0.9 dB gain compared to simulcast, with very similar parallelism requirements. Compared to anchor approach, where the parallel operation is not possible, the penalty to achieve parallelism is negligible (less than 0.025 dB on average). Similar results could be seen in Figure 3, where the RD curve for the sequence Flamenco is plotted. As also seen in Figure 3, proposed scheme has significant gains compared to simulcast with similar requirements on parallelism.

Sequence	**PSNR Results of Proposed Scheme**	
	Δ_{dB} relative to anchor	Δ_{dB} relative to simulcast
Ballroom	-0.005 dB	+ 1.17 dB
Exit	-0.004 dB	+ 0.39 dB
Race1	-0.06 dB	+ 0.92 dB
Flamenco2	-0.07 dB	+ 1.18 dB
BreakDance	-0.002 dB	+ 0.41 dB
Rena	-0.003 dB	+ 1.55 dB
Uli	-0.01 dB	- 0.19 dB
Akko&Kayo	-0.08 dB	+ 1.44 dB

Table 2. Coding Efficiency of the Proposed Scheme relative to Anchor and Simulcast Approaches

5. CONCLUSIONS

Parallel encoding/decoding operation for the MVC standard is especially important for 3DTV systems that support head-motion parallax, where the receiving end needs to decode and display multiple views simultaneously. In this paper, we presented a coding structure for the upcoming MVC standard that enables parallel encoder/decoder operation for different views, even though there are dependencies between views. This is achieved by coding views with some constraints, so that any macroblock in a certain view is allowed to depend only on reconstruction values of a subset of macroblocks in other views. Because of the systematic restriction of dependencies, a decoder could efficiently pipeline different views and achieve parallel operation. As the inter-view prediction is still used, significant gain is achieved compared to simulcast, while maintaining almost the same desirable parallelism characteristics. Experimental results show that, when compared to simulcast, proposed system achieves 0.9 dB gain on average with similar parallelism requirements. The proposed algorithm has also been proposed to MVC standardization [8], and has been adopted in the form of Supplemental Enhancement Information (SEI) message.

It should be noted that, utilizing simpler prediction structures that only use inter-view prediction for frames where temporal prediction is not available (i.e. I-frames in temporal direction) also helps to improve the parallelism of the system significantly, without hurting the coding efficiency [7]. The proposed method could be used in conjunction with those simpler prediction structures, so that I frames in different views could also be decoded in parallel.

6. REFERENCES

[1] “Requirements on Multi-view Video Coding v.5”, document *N7539*, MPEG, Nice, France, Oct. 2005

[2] P. List, A. Joch; J. Lainema.; G. Bjontegaard. M. Karczewicz, "Adaptive deblocking filter,", IEEE Transactions on CSVT, vol.13, no.7pp. 614- 619, July 2003

[3] M. Karczewicz, A. Hallapuro “Interpolation solution with low encoder memory requirements and low decoder complexity.”, VCEG-N31, 24-27 Sept. 2001.

[4] “Common Test Conditions for Multiview Video Coding”, *JVT-U221*, Hangzhou, China, Oct. 2006

[5] “Joint Multiview Video Model (JMVM) 1.0”, *JVT-T209*, Klagenfurt, Austria, July 2006

[6] G. Bjontegaard, “Calculation of Average PSNR Differences between RD-curves” ITU-T SGI 6/Q.6 Doc. Doc. VCEG-M33, April 2001.

[7] P. Merkle, A. Smolic, K. Mueller, T. Wiegand, “Comparative Study of MVC Structures”, *JVT-V132*, Marrakech, Morocco, January 2007

[8] K. Ugur, J. Lainema, H. Liu, Y.K. Wang, “Parallel Decoding Info SEI Message for MVC”, *JVT-V098*, Marrakech, Morocco, January 2007

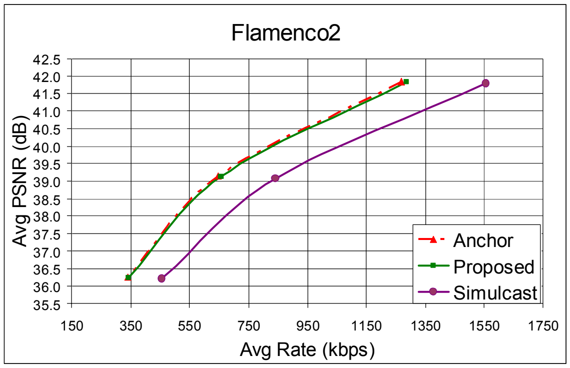

Figure 3. Comparing the Proposed Scheme with Anchor and Simulcast