A Novel Parallel Video Coding Framework for AVS+ Real Time Encoder

Xiaochen Jiang, Guoping Li, Yake Li, Haiwu Zhao, Guowei Teng, and Guozhong Wang

Communication and Information Engineering Department,
Shanghai University, Shanghai, 200072, P.R. China
j-xiaochen@hotmail.com

Abstract. Audio Video coding Standard + (AVS+) has made a high progress in coding efficiency, but the algorithm complexity also increased. Parallel coding techniques are efficient solutions to cope with high-complexity coding. A novel parallel video coding framework was proposed to break the traditional video coding back loop and realize efficient parallelism. The framework was applied to AVS+ real time video encoder. Experimental results demonstrate that the proposed framework makes full use of the computing ability of multi-core processor, and improves the code rate greatly, especially in the case of high resolution video.

Keywords: video coding, AVS+, parallel coding, parallel motion estimation.

1 Introduction

AVS+ Promotion Group has formulated the state-of-the-art video coding standard AVS+ that increases four key techniques to further improve the coding efficiency on the basis of AVS [1]. Though AVS+ greatly improves the coding efficiency for the demands of 3D and high definition video coding, computational complexity of the codec has an increment as well. With the development of multi-core hardware, using multi-core processor to parallelly accelerate video coding has become the main trend in standards setting and codec designing. On the condition of fast development of multi-core hardware and parallel processing tools [2], better software and algorithm of parallel coding are needed to realize effective video compress. The research of parallel coding based on H.264 has a focus in recent years [3-5]. For the new generational codec standard High Efficiency Video Coding (HEVC), Joint Collaborative Team on Video Coding (JCT-VC) proposed many valuable proposals for parallel coding, including WPP (Wavefront Parallel Processing) and Tiles [6-7]. These two technologies can only be applied to HEVC standard and simply base on spatial parallelism which has lowered parallelism degree. What's more, the accurate bit rate allocation has been a problem for developers. AVS+ doesn't support of the parallel technologies of WPP and Tiles, only the structure of slices can be used in data parallel computing. Also, the simple use of intra slice parallel coding not only reduces the coding efficiency but brings strip effects [8].

Considering the analysis above, we find that the traditional video coding framework is a main barrier for parallel coding. This paper proposes a novel parallel coding framework to break the parallel limitation of traditional coding framework. The new framework includes three main modules: parallel video preprocessing, parallel motion estimation and parallel coding. The new framework is applied to AVS+ real time encoder and the experimental results demonstrate that this coding framework eliminates the inter-frame dependency and accelerate the encoder efficiently.

2 Traditional Parallel Coding Algorithms

2.1 Intra-frame Parallel Coding

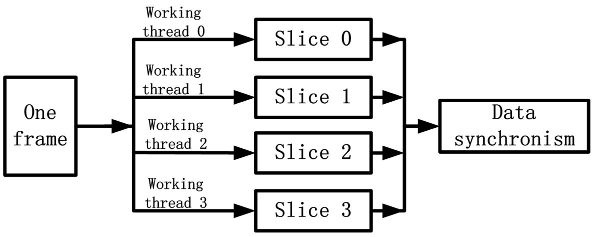


Fig. 1. Slice-level intra-frame parallel coding

Fig.1 shows the fundamental of the intra-frame slice-level parallel coding. One frame is divided into four slices and each slice can be coded within one of the four working threads which work concurrently. There will be a data synchronism after all the threads finish their working task. Though slice-level parallel coding uses the syntactic structural feature to achieve the parallelism and is realized easily, it does not have high coding efficiency and causes strip effects.

2.2 Inter-frame Parallel Coding

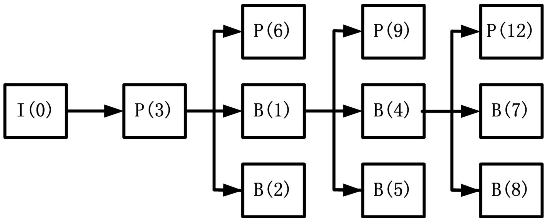


Fig. 2. Inter-frame parallel coding

As shown in Fig. 2, supposing the input sequence type is IPBBP, the first I frame I(0) and P frame P(3) are coded serially. The second P frame-P(6) and the prior two B frames-B(1) and B(2) can be coded concurrently because they don't correlate with each other. For P(6), B(1) and B(2) all refer to I(0) and P(3), three threads are created to process these three frames after I(0) and P(3) are coded parallelly. Then the program returns to the main thread and prepares to code P(9), B(4) and B(5). The main defect of this strategy is that the parallel degree is not high enough for it doesn't eliminate the dependency of inter-frames completely, and may cause system delay.

2.3 Inter-frame Parallel Coding with Reference Dependence

Macroblock's search range is limited because of the continuity of moving object in consecutive frames. The moving object will not beyond a certain range in near reference frames. There is no need for the later frame to wait for the current frame being coded totally. Once a slice of later uncoded frame which refers to the current frame has obtained necessary reference information, the later frame will be coded immediately.

As shown in Fig.3, the input sequence type is IPBBP. Slice-level parallel intra-frame coding is used in intra-frames and the traditional inter-frame coding is employed in inter-frames. Each frame level (a group of frames being coded parallelly in Fig.2) has a time delay, rather than the serial mode of each frame level showed in section 2.2. Fig.3 demonstrates that when the first I (0) is being coded, as long as the slice of P (3) has got the necessary reference information, it can be coded. In a similar way, B (1), B (2) and P (6) as well can start coding, no need to wait for the prior two frames being coded. But this method brings the dependency between reference frames, and reduces the parallel degree.

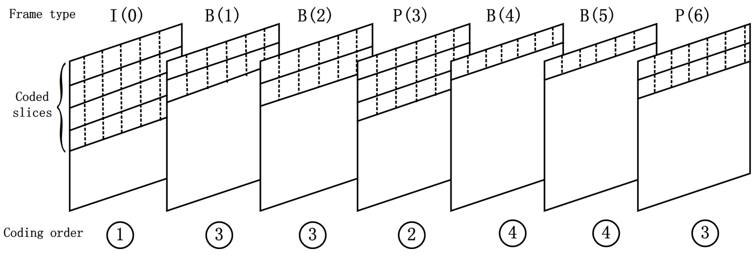


Fig. 3. Inter-frame coding with reference dependence

3 A Novel Parallel Coding Framework

3.1 Limitation of Traditional Video Coding Framework for Parallel Processing

Traditional video coding framework includes two coding paths: forward and reconstructed path. The forward path includes prediction (intra or inter), integer transformation, quantization and entropy coding, and the reconstructed path uses inverse quantization, inverse transformation and motion estimation to reconstruct frames that could be used as the reference frame in prediction. The forward and reconstructed paths constitute the whole coding feedback loop [9].

Owing to this feedback loop, it is difficult to implement the coding parallelism under such structure. A proposed new parallel coding framework breaks this traditional coding framework.

3.2 A New Parallel Coding Framework

As shown in Fig.4, the proposed framework mainly includes three modules: parallel preprocessing module, parallel motion estimation module and parallel coding module. We consider that each module has a time delay of some frames' coding time. In this occasion each module can work concurrently and independently: After the first module has processed the data of M frames, the second module can start to work. After the second module has processed the data of N frames, the third module can start to process.

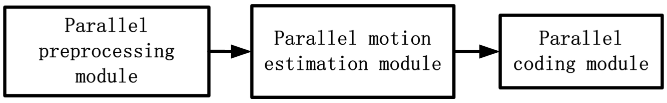

Parallel preprocessing module
Parallel motion estimation module
Parallel coding module

Fig. 4. Proposed parallel coding modules

Fig.5 shows the structure diagram of the entire parallel coding system. The framework breaks the feedback loop and adds the preprocessing module and parallel integer-pixel motion estimation module compared with traditional structure. In parallel coding module, fractional-pixel motion estimation substitutes for the original motion estimation. There are three parts in Fig.5 divided by dotted lines. The three parts are corresponding to the three modules in Fig.4.

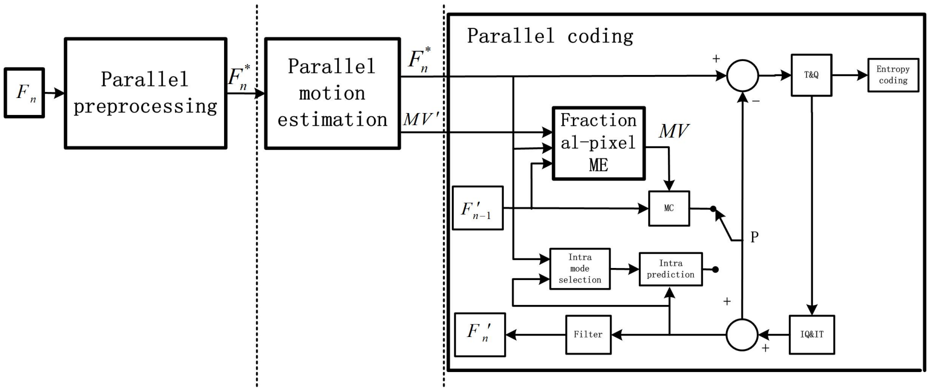


Fig. 5. Structure diagram of the proposed parallel coding framework

(1) Parallel Preprocessing Module

The first part of Fig.5 shows the preprocessing module. Firstly the original video sequence passes through some preprocesses. All preprocesses are conducted based on the original frames spatially and temporally, thus there is no dependency between each other and they can work concurrently. The most important thing in this module is scenes detection and frame types judgment of the sequence by figuring out correlation parameters, which could provide necessary correlation data for the later motion estimation module and parallel coding module.

(2) Parallel Motion Estimation Module

Motion estimation consumes most of time in video coding and integer-pixel motion estimation occupies maximum computational amount and time under the mode of multi references in motion estimation. Moreover, as is mentioned in 3.1, owing to the feedback loop in traditional coding framework, it is hard to make parallel implementation in traditional coding framework. To cope with the problems above, a method that employs the original images as reference images, which separate the most time-consuming integer-pixel motion estimation to conduct the motion estimation parallelly and independently is adopted.

Parallel motion estimation could be conducted under three conditions: (1) There is no dependency between reference frames; (2) Both estimated frame and reference frame could be obtained; (3) The relation between estimated frame and reference frame can be confirmed. The proposed coding framework uses frame types obtained in the preprocessing module and the feature that frames can be buffered by time delay. So it meets all the three conditions listed above and the motion estimation can be performed independently and parallelly in this module. The parallel motion estimation module greatly improves the parallelism of the coding framework and the utilization of multi-core hardware resource.

(3) Parallel Coding Module

Frame type parameters and motion vectors have been obtained through the prior modules, which makes the computational amount in this parallel coding module decrease greatly. We use a method of interlaced coding and each field's process is independent. The inter-frame coding in section 2.3 is adopted. The procedure is showed in the third part of Fig.5. After the necessary selections of macroblock mode, the integer-pixel motion vector MV' is served as the starting point in the fractional-pixel motion estimation. After the fractional-pixel motion estimation we get the final motion vector MV. Next integer transform, quantization and entropy coding are performed to get the AVS stream. This strategy overcomes the disadvantage of method mentioned in section 2.3 and decreases the parallel inter-frame number.

3.3 Parallel Coding Framework for AVS+ Real Time Encoder

AVS+ does not support WPP and Tiles which are parallel techniques in HEVC. Only slice structure coding can be used in parallel coding. AVS+ is mainly applied to 3D, high definition, and interlacing video coding. It can also handle a certain delay time in practical. Considering the features of AVS+ listed above, the new parallel coding framework can be used into AVS+ real time encoder.

Main steps of the parallel real time encoder can be described as follows:

1. Establish L parallel processing threads. Because all preprocessings are implemented using the original frames and the complexity is not high, there is no dependency between preprocessing threads. The thread number commonly sets 2 at most and each thread corresponds to the processing of an original frame.
2. Establish 2×M integer pixel parallel motion estimation threads. After procedure 1, the types of each frame have been confirmed. Parallel integer motion estimation can be conducted according to each original reference frame. AVS+ is mainly applied to interlacing coding. Each frame can be divided into two fields and each of the two fields works as an independent thread when conducting motion estimation.
3. Establish 2×N coding threads. Each thread conducts fractional-pixel motion estimation according to the integer MV from procedure 2. This can efficiently lower the coding complexity. The parallel coding with reference dependency which proposed in 2.3 is adopted in inter-frame coding. Considering AVS+ is mainly applied to interlacing frames coding, each frame can be divided into 2 fields and each field can be treated as a slice coded independently. Thus we not only improve the parallelism, but avoid the loss of coding efficiency and strip effects.
4. Rate control of parallel threads. Because motion vectors and sum of absolute differences (SAD) of coding frames can be obtained after motion estimation in procedure 2, the obtained SADs can be served as the level of complexity of coding frames, which provide basic information for the bit distribution of coding frames.
5. Delay control of the encoder. The time delay of the encoder inferred from above coding procedures is L+M+N frames' coding time.

According to the above procedures and the availability of encoder hardware resources, we should first ascertain time delay of the encoder and the bottleneck of thread management and then distribute L、M、N reasonably.

4 Experimental Results and Analysis

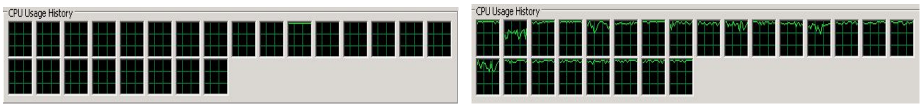
In order to test the performance of the proposed parallel coding structure, a DELL server was used as multi-core test platform. The server has an Intel Xeon CPU with 2×12 cores and each core's basic frequency is 3.06HGz. Six 1920×1080 high definition sequences-parterre, dial, basketball, volleyball, leaf, birdcage-and a spliced sequence by the above 6 sequences are used as test sequences. Coding succession format is IPBB. GOP is 30. Search range is ±32. The testing software is AVS+ high definition real time encoder. The encoder not only sustains the traditional video coding mode (serial coding), but supports the parallel video coding in the framework proposed in this paper. In the experiment, the preprocessing threads parameter L is set 1, parallel motion estimation threads parameter M is set 4, N means parallel inter-frame umber and other parameters' meaning can be seen in Table 1.

Table 1. Descriptions of parameters

Parameter	Description
ME2	UMHexagonS
ME1	Simplified UMHexagonS (remove 5×5 square and big hexogen search)
Fractional-search 1	Sub-pixel MV search 2 times and quarter-pixel MV search 4 times
Fractional-search 2	Sub-pixel MV search 4 times and quarter-pixel MV search 10 times
Fractional-search 3	Sub-pixel MV search 6 times and quarter-pixel MV search 12 times
Parallel inter-frame umber (N)	Number of frames in parallel coding in Fig.3

4.1 CPU Utilization

When computers are doing parallel computation utilizing multi-core processor, CPU utilization is an important feature to reflect the efficiency of algorithm. If the program can make use of all CPU cores to work simultaneously, the speed rate would surely be improved.



(a) (b)

Fig. 6. Comparison of CPU utilization between serial and parallel coding

Fig.6 shows the comparison of CPU utilization between serial and parallel coding. Figure (a) shows the CPU utilization of serial coding. We can see that only the 11th thread is working, which demonstrates that serial coding cannot make full use of CPU cores. Figure (b) shows the CPU utilization of using the proposed parallel coding algorithm, setting motion estimation parameters ME2 and Sub-search 3, and Parallel inter-frame number 16. The results show that all CPU cores take part in the process and all are of high-efficiently use.

4.2 Speedup Ratio

Speedup ratio is an important index in measuring the performance of the parallel computing. It reflects an increasing degree of the program computing speed from serial process to parallel optimization. In this paper the speedup ratio is defined in formula (1). Coding frame rate can be represented by parameter fps, which also reflects the speed of an encoder.

Table 2. Speedup ratio and fps in different motion estimation

Motion estimation	Fractional-pixel search	Parallel fps	Serial fps	Average speedup ratio
ME1	Fractional-search 1	51.4	4.2	12.24
ME2	Fractional-search 1	43.7	3.7	11.81
ME1	Fractional-search 2	50.4	4.2	12.00
ME2	Fractional-search 2	42.4	3.6	11.78
ME1	Fractional-search 3	50.5	4.2	12.02
ME2	Fractional-search 3	41.3	3.5	11.80

$$speedup = \frac{frame\ rate(fps)\ after\ prallel\ optimization}{frame\ rate\ before\ prallel\ optimization} \tag{1}$$

In formula (1), speedup ratio is presented by speedup, the larger the value, the better the parallelism of the algorithm. Also the coding speed is faster.

Table 2 shows the speedup ratio of testing the 1500 frames of the spliced sequence in ME1, ME2 and Fractional-search 1, 2, 3. It can be seen clearly that encoder employing the proposed parallel framework can get a sufficient speedup ratio that completely meets the practical demands.

4.3 Inter-frame Parallelism

As is mentioned before, Parallel inter-frame number (N) in inter-frame prediction is a key issue to determine coding parallelism. The parameter cannot be too small or too large; otherwise parallelism of the encoder will be negatively affected. What's more, there will be some problems in contributing bit rate. So it is an issue of importance to determine a suitable N in inter-frame prediction.

Table 3. Fps for different index N

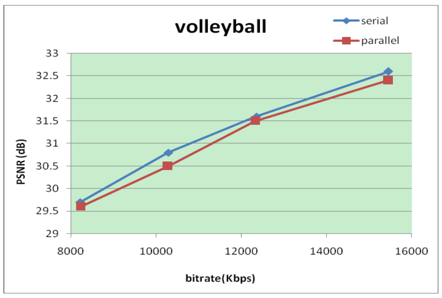
N	fps(ME2)	fps(ME1)
4	24.9	24.7
5	29.0	30.4
6	34.0	35.4
8	40.1	43.3
9	40.6	46.1
10	40.8	48.3
11	41.0	49.7
12	41.3	50.5
16	40.9	49.4

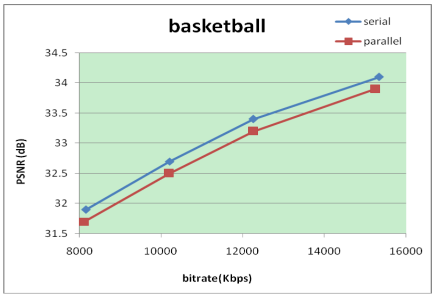
Table 3 gives the experimental results for different index N. It is easy to understand that the larger the index N is, the better improvement of the subjective quality of the decoded frames will have. However we should consider that whether the frame rate can improve as well. From Table.3 we can see that when N reaches 8, fps is high enough to meet the practical demands of encoder. Under this circumstances, the delay of encoder is 13 frames' time (L=1, M=4, N=8). Performance of bit-rate control in the encoder will be improved on the condition of low N. It can be seen from Table 3 that fps improves largely as N increases from 4 to 12, but when N arrives 12, fps starts to fluctuate and stops increasing. From the analysis above, we can reduce N if the coding speed is acceptable, which can provide advantages for better realization of rate control.

4.4 Rate-Distortion

After testing 6 high definition sequences, we show rate-distortion curves in different bitrates by serial and parallel coding. As shown in Fig.7, two curves of different color in each figure represent the result of serial and parallel coding. PSNR in different bitrates reflects the quality of decoded frames. The average PSNR loss of the 6 sequences is 0.13 dB and the minimum loss is 0 dB. The largest loss is about 0.2 dB, which can be seen in Fig.7.

We pay more attention on the practical application whose video may be hard to encode, just like volleyball.yuv and basketball.yuv. After testing these two sequences we can see even in the worst condition, there is only a loss about 0.2 dB. Making the original frames as reference frames in motion estimation is feasible and remains little negative impact for encoder. For the quality of decoded frames, using the proposed parallel encoder gets the same satisfactory objective and subjective effects as serial coding does. In addition, the proposed encoder has a remarkable advance in coding rate, which can further display its advantages: good performances and low losses.





(a) (b)

Fig. 7. The Rate-distortion curves with worst performance in 6 curves

5 Conclusion

We designed a novel parallel coding framework, breaking the traditional video coding structure and making use of the encoder's feature of time delay. Combining the coding framework with AVS+, we proposed a parallel coding framework for AVS+ real time encoder. The experimental results showed that the parallel encoder can make full use of the computational ability of the multi-core processor, and greatly improve the coding speed on the basis of least loss and controllable coding delay (within 0.5 s).

References

1. Zhang, X.L.: The promotion and application of Terrestrial digital TV industrialization. Video Engineering 36(22), 3–4 (2012)
2. Chen, D., Singh, D.: Fractal Video Compression in OpenCL: An evaluation of CPUs, GPUs, and FPGAs AS Acceleration Platforms. In: 18th Asia and South Pacific Design Automation Conference (ASP-DAC), pp. 297–304. IEEE Press, Yokohama (2013)
3. Pang, Y., Zhang, F.Y., Sun, L.F.: Survey of Parallel Acceleration Algorithms of Video Coding on Multi-core Processor. Journal of Frontiers of Computer Science and Technology 3(4), 337–346 (2009)
4. Ge, S., Tian, X., Chen, Y.-K.: Efficient multithreading implementation of H.264 encoder on Intel hyper-threading architectures. In: IEEE Proceedings of the 2003 Joint Conference of the Fourth International Conference, pp. 469–473. IEEE Press (2003)
5. Tian, X., Chen, Y.-K., Girkar, M., et al.: Exploring the use of hyper-threading technology for multimedia applications with Intel OpenMP compiler. In: Proceedings of the Intel Parallel & Distributed Processing Symposium, pp. 36–43. IEEE Press, Nice (2003)
6. JCTVC-G1025: Tiles and Wavefront Parallel Processing. 7th Meeting. Geneva CH, November 21-30 (2011)
7. JCTVC-F274: Wavefront Parallel Processing for HEVC Encoding and Decoding. 6th Meeting. Torino, IT, July 14-22 (2011)
8. Xu, C.M., Li, G.P., Wang, G.Z.: Study and Implementation of AVS Encoding algorithm Based on Slice Parallel. Journal of Image and Graphics 14(6), 1108–1113 (2009)
9. Chiang, C.-K., Lai, S.-H.: Fast H.264 Encoding Based on Statistical Learning. In: Qiu, G., Lam, K.M., Kiya, H., Xue, X.-Y., Kuo, C.-C.J., Lew, M.S. (eds.) PCM 2010, Part II. LNCS, vol. 6298, pp. 179–189. Springer, Heidelberg (2010)