PARALLELIZATION TECHNIQUES OF THE DECODING ALGORITHM OF THE NEW VIDEO COMPRESSION STANDARD H.265/HEVC

R. I. Chernyak

Tomsk State University, Elecard Devices
Tomsk, Russia
E-mail: r.chernyack@gmail.com

In our paper we consider parallelization techniques of the decoding algorithm of a new standard for digital video compression. Two methods of the decoder parallelization are compared in terms of speed and their strengths and weaknesses. The paper presents experimental data on the speed of the proposed algorithms at different inputs. The maximum possible acceleration within the proposed methods is computed theoretically.

Keywords: digital video compression, HEVC decoder parallelization, H.265/HEVC, parallel video decoding.

The newest video compression standard H.265/HEVC [1] was developed in 2013 by Video Coding Expert Group (VCEG) and Motion Picture Expert Group (MPEG). H.265 is the next step of the video compression technologies, and it is expected to replace the current video compression standard H.264/AVC [2]. The main requirement for HEVC is a double bitrate reduction (compared to AVC) with quality preservation. To meet this requirement, standards developers offered completely new and significantly improved the existing video compression methods. As a result, the goal was achieved, but at the same time codec logics became more sophisticated, and it caused a significant decrease in speed. But speed is a critical factor in an application, so effective implementation of HEVC standard is still a problem to solve. One of the most effective ways to achieve it is using parallelization. The overwhelming majority of modern devices, both desktop and mobile, contain multicore processors which allow to perform parallel calculations. Thus it is desirable to develop coding and decoding algorithms within HEVC standard that can be effectively executed by modern devices. In this paper we propose methods of decoding algorithms parallelization.

Briff description of H.265/HEVC

Like the previous video compression standards, HEVC performs video sequence coding frame by frame. The basic coding element, called *coding unit (CU)*, is a square block of pixels sized 16×16, 32×32 or 64×64. Very tentatively a coding algorithm can be described in the following way: an input frame is divided into CUs and then each CU is coded. To speak more formally, the codec implements the classical *hybrid coding scheme* [3]. Not going into its details, we will note only that the last stage of CU coding is entropy compression by a *Syntax-based context-adaptive Binary Arithmetical Coder (SBAC)*. Thus, the first step of the decoding process is to extract data, which correspond to the current CU, from the bitstream, then decode these data entropically, and, finally, *reconstruct* the block of pixels. We must keep in mind that SBAC is a context adaptive entropy coder, which means that to code (or decode) each of the following CU, the codec will use information about its state at a given moment (*context*), which updates after processing each CU. Thus we can see that entropy frame coding (or decoding) is a strictly sequential process.

Parallelization techniques

In this paper we suggest two methods of decoder parallelization within HEVC. The first method is universal for all HEVC-valid bitstreams. The second one uses new standards feature and therefore it works only with specific coding bitstreams.

Lines parallelization

The following are the results of decoder profiling of a typical Full HD video (video with frame size 1920×1080 pixels):

- 10% of time using entropy decoding;
- 65% - using reconstruction;
- 25% - using auxiliary operations (e.g. memory initialization, etc).

As it has been mentioned before, entropy decoding is a sequential process, but reconstruction can be processed parallely. There is only one requirement to observe.

Requirement 1.

CUs located in (i, j) position can be reconstructed correctly only if its four "neighbors" located on position: $(i-1, j)$; $(i-1, j-1)$; $(i, j-1)$ and $(\mathrm{i}-1, \mathrm{j}+1)$ have already been reconstructed.

Fig. 1 shows this condition. Dark color presents CUs which have already been reconstructed, light grey CUs are undergoing reconstruction, and the white ones are waiting for their turn. In this figure reconstruction is performed parallely by three calculators.

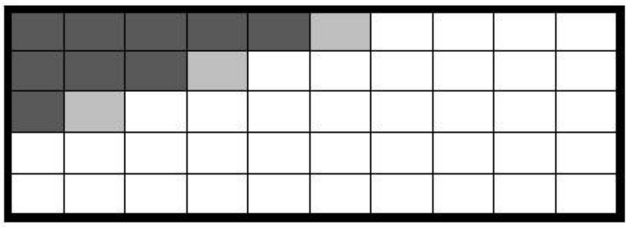

Fig. 1. Example of parallel frame reconstruction

The proposed method of HEVC decoder parallelization is based on the following ideas:

1) separation of entropy decoding process from reconstruction process;
2) the reconstruction process parallelization is performed observing requirement 1.

The proposed algorithm can be described in the following way. There is one main thread and a pool $(n-1)$ of working threads. Let the main thread sequentially perform entropy decoding of the current frame. As soon as one line of CUs is decoded, it is "handed over" to the pool of working threads for reconstruction. Each working thread from the pool gets a decoded line and reconstructs it, considering requirement 1. If requirement 1 is not observed, the thread falls asleep and sleeps until another thread awakes it. Otherwise, the thread reconstructs a current CU, sends notification to other thread and proceeds to the next CU. Meeting requirement 1 is checked before each CU reconstruction.

Tiles parallelization

One of the new video compression features in HEVC is the possibility of parallel entropy frame decoding. In general, SBAC is a strictly sequential algorithm, but in HEVC developers offered several ways to parallelize it. In this paper we describe one of such ways – *tiles parallelization*.

The frame is presented as an $n \times m$ matrix of blocks, each consisting of a set of CUs. These blocks are called *tiles*. The entropy codec context for each tile beginning is default. Tiles are coded and decoded independently, therefore the coding and decoding procedure can be parallelized when working with tiles. Fig. 2 shows one of possible ways to divide a frame into tiles.

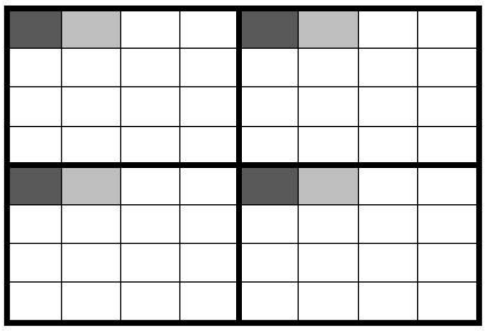

Fig. 2. Possible frame partition into tiles

In this figure a frame is divided into 4 tiles using a 2×2 matrix. Decoding in each tile is performed parallely. Note that tile parallelization allows to parallelize both entropy decoding and reconstruction. Thus the tiles method is more effective than the lines one, but a tiled frame is a specific codec feature, which can be not presented in a usual bitstream.

It should be noted that between the tiles SBAC context always resets, which leads a slight decrease of the codec quality.

Experiments

A series of experiments was conducted within the proposed subject. We used three different video sequences as input data: animated films Big Buck Bunny [4] and Sintel [5], which are customarily distributed by Blender Foundation [6], and a video from the Russia Serfing Cup and International Kitesurfing Completion in Vung Tau in 2013 [7]. Every video was presented in several resolutions:

- Big Buck Bunny – 1920×1080 (1080p), 1280×720 (720p) and 640×360 (360p);
- Sintel – 1920×816 (816p), 1280×408 (408p) and 640×272 (272p);
- Serfing – 1280×720 (720p) and 640×360 (360p).

Durations of the videos are 14315, 8883 and 21313 frames respectively.

Experiments were carried out for three decoding algorithms:

1) sequential;
2) lines parallelization;
3) tiles parallelization.

The hardware platform was a personal computer with an Intel Core I7 3770 processor and 8Gb RAM. The results are presented in Table 1. The decoding speed is given in frames per second (*fps*).

Table 1

Decoding speed by different algorithms with different inputs

Input data	Sequential	Lines parallelization	Tiles parallelization
Big Buck Bunny 1080p	20.4	38.7	43.1
Big Buck Bunny 720p	44.6	79.1	89.5
Big Buck Bunny 360p	159.0	201.6	242.6
Sintel 816p	23.4	47.5	52.6
Sintel 408p	83.3	128.4	154.4
Sintel 272p	172.3	217.5	263.1
Serfing 720p	23.8	50.5	56.2
Serfing 360p	80.0	123.4	155.8

The results of the experiment show that the greatest acceleration occurs in the tiles model with high resolution videos. With the decrease of the frame size the acceleration also decreases.

Let us estimate theoretical acceleration for the lines and tiles parallelization models when decoding a Full HD video. 10%, 65% and 25% of the decoding time will be used for entropy decoding, reconstruction and auxiliary operations respectively. Let t be the time for sequential decoding, then $0.65t$ will be the time for sequential reconstruction. For the parallel lines model with four threads, reconstruction time will be four times less. Thus, the total decoding time can be estimated by the following formula: $0.64t/4 + 0.1t + 0.25t = 0.5125t$. As the speed of decoding is inversely proportional to time, theoretically the acceleration coefficient equals to 1.95. Acceleration coefficient for the tile parallelization model estimated in the same way theoretically equals to 2.29. It should be noted that theoretically calculated coefficients are a bit less than the experimental ones. This happens because in both parallelization models data depends on each other. In lines model this dependency is defined by requirement 1; in tiles model - by potential difference of tiles complexity.

We have already mentioned that there is a slight decrease in the SBAC quality with the growing number of tiles. An experiment to calculate the degree of this quality reduction was conducted within the presented research. The same video was coded into a different amount of tiles with other parameters fixed. Table 2 shows changes in the video quality, presented as a *Peak signal-to-noise ratio (PSNR),* and compression effectiveness (*bitrate*) depending on the number of tiles.

Table 2

Dependence video quality and compression ratio of the number of tiles

Amount of tiles	1	2	3	4	5	6	7	8
Bitrate (kbps)	1456.2	1466.2	1474.4	1486.2	1494.9	1501.7	1507.7	1519.6
PSNR (dB)	41.448	41.444	41.442	41.440	41.437	41.434	41.433	41.432

As we can see, the reduction of quality is negligible even when the frame is divided into eight tiles.

Conclusion

To sum up, both theoretical and experimental data prove that tiles parallelization model is better that the lines model. However, in the general case using a tiles model is not possible. It is more efficient to use the tiles model for decoding tile bitstreams, otherwise the lines model should be used. It should be born in mind that the lines model can be used within the tiles model (within each tile). This approach is of interest when decoding is performed with modern multicore devices. It gives an opportunity to use each core to the fullest.

The results of research were obtained in Tomsk State University of Control Systems and Radioelectronics with the financial support from The Ministry of Education and Science of the Russian Federation.

Bibliography

1. *Bross B., Han W. J., Ohm J. R*. High efficiency video coding (HEVC) specification draft 10 (for FDIS & Last Call). 12Th Meeting : Geneva, CH, 14–23 Jan. 2013.
2. ISO/IEC 14496-10:2003. Information technology. Coding of audio-visual objects. Part 10: Advanced Video Coding.
3. *Sullivan G. R., Ohm J. R., Han W. J., Wiegand T*. Overview of the Hight Efficiency Video Coding (HEVC) standard // IEEE Trans. Circuits and System for Video Technologiy. 2012. V. 22. №. 12. P. 1649–1668.
4. http://www.bigbuckbunny.org/ Big Buck Bunny. 2013.
5. http://www.sintel.org/ Sintel, the Durian Open Movie Project. 2013.
6. http://www.blender.org/ blender.org – Home of the Blender project – Free and Open 3D creation software. 2013.
7. http://www.vungtau-city.com/?p = 1558 Russia Surfing Cup and International Kitesurfing Competitions in Vung Tau | Vung Tau City Portal. 2013.