International Journal of Hybrid Information Technology
Vol.8, No. 5 (2015), pp. 293-302
http://dx.doi.org/10.14257/ijhit.2015.8.5.32

DCT-JPEG Image Coding Based on GPU

Rongyang Shan[1], Chengyou Wang[1*], Wei Huang[2] and Xiao Zhou[1]

[1] School of Mechanical, Electrical and Information Engineering, Shandong University, Weihai 264209, China
[2] School of Electronic Information, Wuhan University, Wuhan 430072, China
sdusry@163.com, wangchengyou@sdu.edu.cn, 258357656@qq.com, zhouxiao@sdu.edu.cn

Abstract

In this paper, the parallel algorithm of JPEG coding based on GPU is proposed, most image compression systems have efficiency problem and the real-time of wireless multimedia sensor networks (WMSN) which used in image compression and transmission is also an issue need to be solved, so in this paper parallel computation is used in JPEG coding, it is an effective ways to solve these problems. The system of JPEG coding system mainly has eight parts: discrete cosine transform (DCT) and inverse DCT, quantization and inverse quantization, Zig-zag ordering and inverse Zig-zag ordering, Huffman coding and decoding. The proposed parallel algorithm of JPEG coding makes all the parts of JPEG system run on GPU, so the speed of JPEG coding is improved significantly. DCT and Huffman coding are wildly used in image processing; therefore the proposed parallel algorithm can be used in many fields about image compression and processing. We use the CUDA toolkit based on GPU which is released by NVIDIA to design the parallel algorithm of DCT-JPEG algorithm. The experimental results show that compared with conventional JPEG coding, the maximum speedup ratio of parallel algorithm of JPEG coding can reach more than 120 times, and the reconstructed image has almost the same performance with the serial algorithm in terms of objective quality and subjective effect.

Keywords: *parallel computing, GPU, discrete cosines transform (DCT), JPEG, Huffman coding*

1. Introduction

Due to the increasing demand on transfer and storage of image data, image compression has become a hot research area for many years. The most useful and successful image coding standard should be joint photographic experts group (JPEG) standard, also known as ITU-81 [1]. Because JPEG encoding can be implemented quickly, around 80% of the images on the Internet observe JPEG standard. The JPEG system mainly makes up of discrete cosine transform (DCT) [2] and Huffman coding, DCT has developed quite mature in recent years, DCT is applied to the standards of the international image compression and video compression, like JPEG, MPEG-2 [3], MPEG-4 [4], H.264/AVC [5], and H.265/HEVC [6], and DCT is also widely used in many other fields of image processing. Huffman coding is one of cores in JPEG system, which is the key of data compression. But the JPEG system still has disadvantages. For example, computational complexity of DCT and Huffman is real high, they occupy most of memory resource and computing powering in the whole JPEG system, that make the efficiency of JPEG system low.

Image compression and transmission [7] are widely used in wireless multimedia sensor networks (WMSN). The image data could be transferred Ultra Wideband (UWB) or WI-

[*] Corresponding Author

FI. The way they work is compressing the image at sending end, and the WMSN transfer image data from sending end to receiving end as packets of data. The data is uncompressed and reconstructed in receiving end. But the compression algorithm always has high time complexity for WMSN. Parallel algorithm provides an effective way to improve the efficiency and real-time of WMSN.

The scaling rate of the computational capability of individual processing units has slowed dramatically. The bottlenecks of CPU's development are already appearing. People are not able to improve the efficiency of computation by improving the frequency of processor, and then the parallel computation based on GPU becomes more and more popular. In the field of parallel computation, GPU has more advantages than CPU. Because GPU have more ALU, it can launch thousands of threads at the same time, so GPU is more suitable for parallel computing. Currently parallel computing based on GPU has been widely used in scientific research. Compute unified device architecture (CUDA) which is released by NVIDIA makes parallel program easily. Parallel computation has wildly used in image procession [8] and video compression [9]. In 2011, Tokdemir and Belkasim [10] used parallel DCT algorithm in data compression, and they got a satisfying result in efficiency. Liu and Fan [11] designed parallel program for DCT in 2012, they used parallel DCT algorithm in JPEG coding, and the parallel DCT algorithm gains 20 times acceleration than serial DCT algorithm in the experiment, but they did not make all parts of JPEG algorithm run on GPU.

The rest of this paper is organized as follows. Section 2 introduces CUDA. Section 3 describes the design of DCT-JPEG's parallel algorithm. Experimental results of the proposed method are presented in Section 4. Conclusions and remarks on possible further work are given finally in Section 5.

2. CUDA

CUDA is released by NVIDIA in 2007. it is an easy-to-use programming interface which is added to graphics card. The CUDA application which runs on GPU is consist of three main components CUDA libraries, CUDA runtime API and CUDA device API (as shown in Figure1). Figure 2 displays the programming model of CUDA. In this model, CPU could be seen as host and GPU should be seen as device which has a memory. In this system, it can have one host and multiple devices. Under this model, host and device work together and fulfill their proper function. At first, the data should be copied to memory by CPU, after that the CPU is responsible for preparing data and driving kernel, while the GPU is focused on the implementation of highly threaded parallel processing tasks. CPU and GPU have their own memory and they cannot visit each memory directly [12]. Data need to be copied from host to device at the beginning of the program and copied back at the end of the program; it will have some latency in the CUDA application. But it will be solved in the future, because the CUDA toolkit will support unified memory in next version.

After developers analyze the algorithm, they give the part of program which is needed parallel computing to GPU. The function which runs on GPU for parallel computing is called kernel. Kernel is not a complete program; it is part of CUDA programs which runs on GPU and is used to compute what you need. The kernel function and the serial processing in host-side compose a complete Parallel algorithm program (as shown in Figure 2).These programs will be executed with the order of the sentences in the program. Host-side code is mainly used to prepare data and run the kernel on GPU.

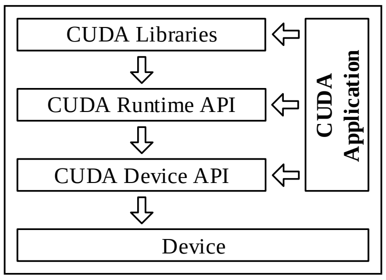


Figure 1. The Architecture of CUDA

In CUDA application, kernel is organized as grid, every kernel has one grid, grid is made up of blocks; and every block has many threads. Generally, the number of threads in every block has the relation with GPU. There are two levels of parallelism in a kernel, parallelism between blocks in the grid and parallelism between threads in the block. Each thread executes the kernel one time according to the serial order of the instruction in the program [12]. Threads in the same block can communicate through shared memory, so developers would have more room for their program.

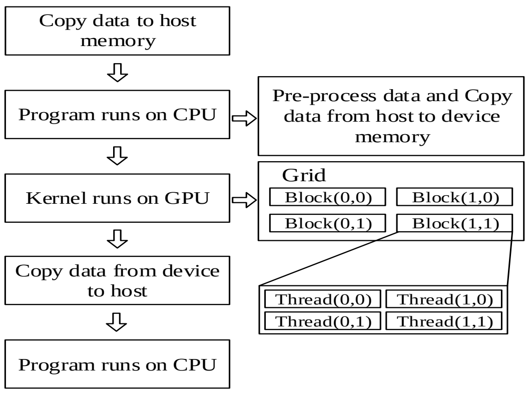


Figure 2. Parallel Algorithm Program Model

3. Parallel Algorithm of JPEG System

The system's figure of basic JPEG based on DCT shows in Figure 3. It mainly has four parts: DCT and IDCT, quantification and de-quantification, Zig-zag ordering and inverse Zig-zag ordering, Huffman coding and decoding. In JPEG system, the image is processed by image block. The image is divided into 8×8 sub-images. The sub-image is a processing unit, it will be processed by DCT, after DCT, every sub-image gets an 8×8 matrix. The 64 DCT coefficients will be quantized by quantization tables, then the coefficients should be coded by Huffman coding. After the above processing, the source image is compression as JPEG format.

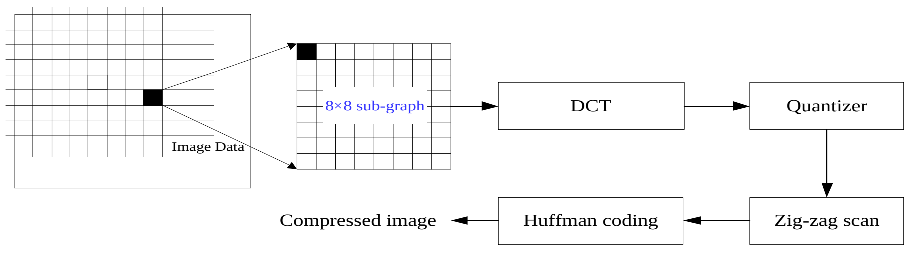


Figure 3. DCT-JPEG System

Accordingly, the decoder also has four parts: Huffman decoder, inverse Zig-zag ordering, inverse quantification, and inverse DCT. It has the reverse order with encoder.

3.1. The Architecture of Parallel DCT-JPEG

The important part of designing parallel program is breaking down task. These sub tasks are processed parallel for improving the computational efficiency. At beginning of JPEG coding, the source image is divided into 8×8 sub-images, and every sub-image has 64 pixel data. So it can be mapped into CUDA architecture perfectly. Because there are two levels of parallelism in a kernel, parallelism between blocks in the grid and parallelism between threads in the block. In the parallel DCT-JPEG model, the source image is mapped into the grid, the sub-image is mapped into block, and every block have 64 threads, every thread process one pixel in sub-image, like Figure 4.

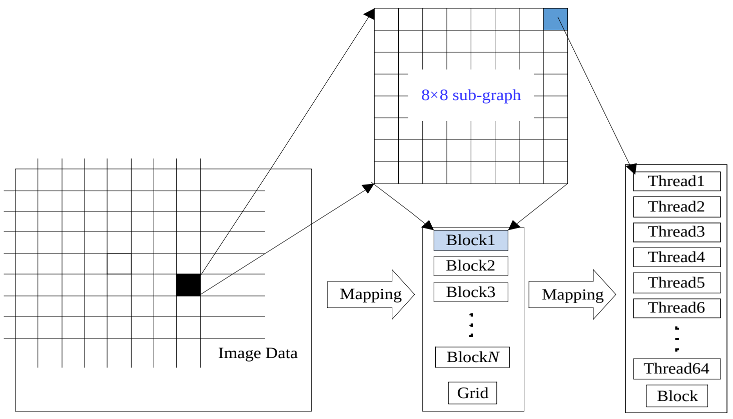


Figure 4. Mapping Relation Model

The whole task of DCT-JPEG need to be divided into N parts, every part is processed in one block, and every block has 64 threads. If the size of the source image is not integral multiple of 64, it can fill zeros at the end of source data.

$$N = \frac{W \times H}{64} \tag{1}$$

where N is the number of blocks in the grid, W is the width of the source image, and H is the height of source image.

3.2. Parallel DCT and Quantizer

DCT removes the correlation between pixels in each sub-image; it provides necessary conditions for image compression. In JPEG system, DCT is a 2D transform. Every block in GPU is executed parallel, and the DCT is executed in every block. After DCT, every block gets 64 DCT coefficients from 64 threads. The first coefficient, which locates in the upper left corner, is DC coefficient; the other 63 coefficients are AC coefficient. The substance of DCT is matrix multiplication, expressed as

$$\boldsymbol{Y} = \boldsymbol{C}\boldsymbol{X}\boldsymbol{C}^{\mathrm{T}}, \tag{2}$$

where $\boldsymbol{X}$ is the two-dimensional image matrix, $\boldsymbol{Y}$ is the transform coefficients block, $\boldsymbol{C}$ is the DCT transform matrix which is shown in (3),

$$\boldsymbol{C}(i,j)=\begin{cases}\sqrt{\dfrac{1}{N}}, & i=0,\ j=0,1,\cdots,N-1,\\[2ex] \sqrt{\dfrac{2}{N}}\cos\dfrac{i(2j+1)\pi}{2N}, & i=1,2,\cdots,N-1,\ j=0,1,\cdots,N-1.\end{cases} \tag{3}$$

In parallel algorithm of DCT, blocks and threads are executed parallel in the macro. Every thread gets a DCT coefficient and then processes the data of DCT coefficient parallel. After DCT, every DCT coefficient is quantified by quantization table. In JPEG system, it contains two kinds of quantization tables (as shown in Figure 5): chrominance quantization table and luminance quantization table.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

(a)

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

(b)

Figure 5. Quantization Tables: (a) Luminance Quantization Table, (b) Chrominance Quantization Table

3.3. The Design of Parallel Huffman Coding

After threads executing the code of DCT and quantization completely, the block will get an 8×8 coefficient matrix. Every thread puts its data in a one-dimensional array with the scanning order of Zig-zag in Figure 6.

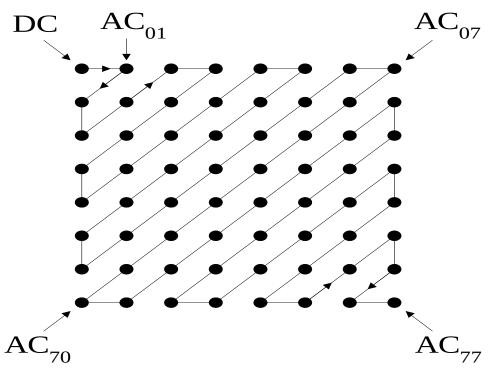


Figure 6. Zig-zag Ordering

In JPEG system, entropy coding contains Huffman coding and arithmetic coding, and in the baseline JPEG, only Huffman coding is used. DCT and quantization are prepared for image compression, by entropy coding; the image data is further compressed.

3.3.1. The Parallel Huffman Coding of DC Coefficient

In parallel JPEG system, there are two levels for parallelism in Huffman entropy. The first level is block-level parallelism, which is insufficient to achieve good performance as we witnessed during the development process. The second level is intra block parallelism; it utilizes the fact that both run-length encoding (RLE), which compresses long sequences of zeros after quantization of DCT-transformed data, and lookup of Huffman codes can be done in parallel. At first DC coefficient is coded by DPCM as shown in Figure 7.

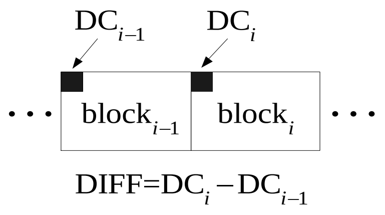


Figure 7. The DPCM of DC Coefficient

The first thread is called thread zero, when the difference between two adjacent blocks is calculated by thread zero. The next task of thread zero is coding the difference. The DC coefficient can be described by two symbols in Figure 8. A is the size of the difference and B is the amplitude of the difference. While thread zero gets A, it will query the DC Huffman table for Huffman code, and B is coded by VLI. The thread zero puts the Huffman code of B in the end of the Huffman code of A, which constitutes the Huffman code of DC coefficient.

Symbol A	Symbol B
(SIZE)	(AMPLITUDE)

Figure 8. The Huffman Code of DC Coefficient

3.3.2. The Parallel Huffman Coding of AC Coefficient

Huffman coding of AC coefficient can also be described as two symbols like DC coefficients: symbol A contains run-length and the size of non-zero AC coefficient, which is coded by run-length encoding (RLE). Symbol B is the amplitude of AC coefficient. In parallel algorithm, how to get run-length is a technical problem, because threads which running on GPU is concurrent, we cannot count it like in serial algorithm. In parallel algorithm, we get run-length by sorting algorithm.

We use a vector which size is 64 in the share memory. When Zig-zag ordering, every thread compares its coefficient with zero, if the coefficient is not equal to zero, we put the number of the thread into the vector. If the coefficient is equal to zero, we put the number which is above 64 in the vector like in Figure 9, in order to facilitate the sorting, because the biggest number of thread is 63. When the arrow is sorted, the difference of two adjacent numbers in the arrow is run-length. The maximum number of the arrow is at the end of the arrow which is described as end of block (EOB).

0	65	65	65	4	65	6	65	···	65

0	4	6	15	65	65	65	65	···	65

Figure 9. The Mean to Get Run-Length

When the run-length of the AC coefficient and the size of AC coefficient are got, the Huffman code can be obtained by querying the luminance Huffman table of AC coefficient. The amplitude of AC coefficient is also coded by VLI. The highest bit of VLI is the sign bit. If the amplitude is above zero, the sign bit is “1” and the binary of the amplitude is the symbol B. If the amplitude is below zero, the sign bit is “0” and the ones-complement code of amplitude is the symbol B.

4. Experimental Results

In the experiment, the gray image Lena (8bits/pixel) is used and the CPU used for the experiment is Intel i3 3.10GHz with 6GB DDR3 memory. The GPU used for the experiment is GTX480 with 384 CUDA cores. The result shows that the efficiency of parallel DCT-JPEG is much higher than serial DCT-JPEG. The core of DCT-JPEG is DCT JPEG, the runtime of DCT on different platforms is compared in Figure 10, from Figure 10, we can know that the speed of DCT algorithm runs on GPU is much high than CPU. The running time of serial program in Figure 10 is ten times smaller than real data, because the efficiency of parallel algorithm is too high to use the same ordinate with the serial algorithm. Peak signal to noise ratio (PSNR) is chosen to measure the performance of reconstructed images in the experiment.

$$\mathrm{PSNR} = 10\lg\left[\frac{255^2 MN}{\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}[x(i,j)-y(i,j)]^2}\right](\mathrm{dB}) \tag{4}$$

where x and y are the original image and reconstructed image respectively, M and N are the dimensions of image array, i and j are the locations of pixels.

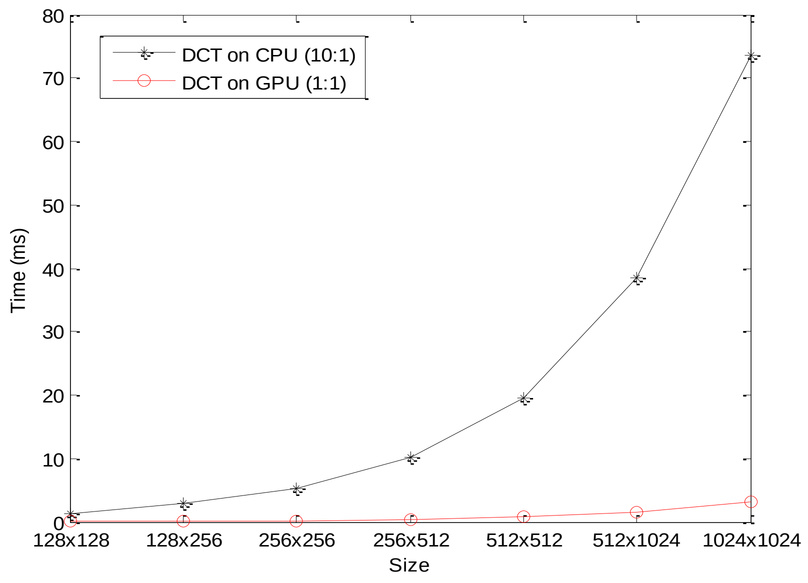


Figure 10. The Runtime of DCT on Different Platforms

Figure 11 shows the reconstructed images of Lena which size is 512×512 and the PNSR is 0.20bpp. From Figure 11, it can be seen that the reconstructed image using parallel algorithm has the same subjective effect with the serial algorithm The PSNR of reconstructed images which obtained from the parallel algorithm which runs on GPU is almost the same as the serial algorithm runs on CPU. So the reconstructed image using the parallel algorithm has the same performance with the serial algorithm in terms of objective quality and subjective effect.

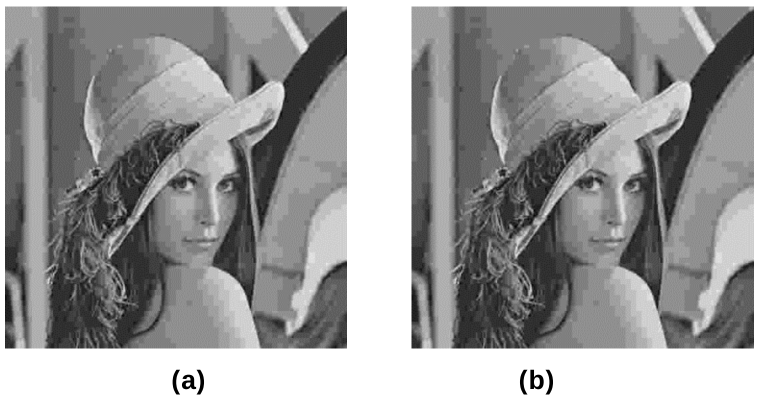


Figure 11. The Reconstructed Images of Lena (0.20bpp): (a) DCT-JPEG on CPU, (b) DCT-JPEG on GPU

The different sizes from 128×128 to 1024×1024 of Lena are used to test the efficiency of parallel DCT-JPEG algorithm. We compare the runtime on GPU with the runtime on CPU as shown in Figure 12, and the runtime of GPU is smaller than the real time too. From the experimental results in Figure 12, we can know the efficiency of parallel DCT-JPEG algorithm is much higher than serial algorithm, so the efficiency of parallel computing is very impressive in general.

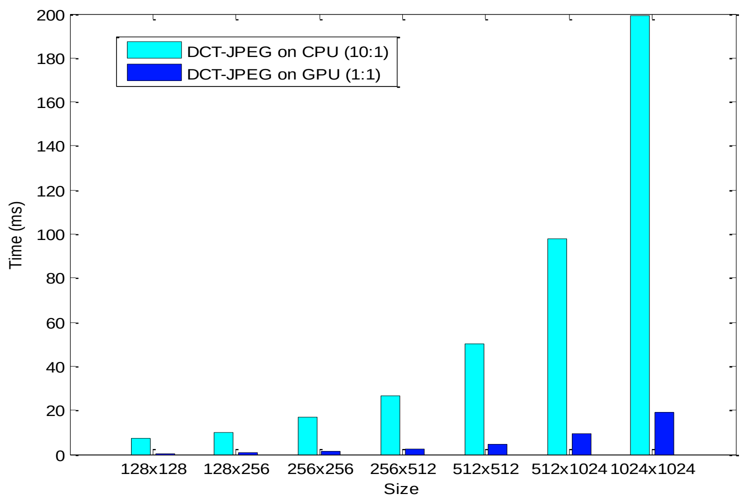


Figure 12. The Runtime of DCT-JPEG on Different Platforms

5. Conclusion

In this paper, we use parallel computing base on GPU in DCT-JPEG. Compared with the conventional algorithm, the parallel DCT-JPEG that runs on GPU could gain at least 100 times acceleration, and the maximum speedup ratio can reach more than 130 times. The parallel algorithm runs on GPU is not only can get high efficiency but also can get the same reconstructed image with serial algorithm. Therefore, parallel algorithm is very helpful to improve the real-time of WMSN. Through comparing the runtime of parallel algorithm, serial algorithm, although the parallel computing gains higher efficiency than serial computing, the DCT-JPEG has serious blocking artifacts when the image is highly compressed at low bit rate. These issues will be further researched in the future work.

Acknowledgements

This work was supported by the promotive research fund for excellent young and middle-aged scientists of Shandong Province, China (Grant No. BS2013DX022) and the National Natural Science Foundation of China (Grant No. 61201371). The authors would like to thank Qiming Fu and Fanfan Yang for their help and valuable suggestions. The authors also thank the anonymous reviewers and the editor for their valuable comments to improve the presentation of the paper.

References

[1] ISO/IEC, “Information Technology -- Digital Compression and Coding of Continuous-tone Still Images, Part 1: Requirements and Guidelines”, ISO/IEC 10918-1 | ITU-T Rec. T.81, **(1994)** February 17.

[2] N. Ahmed, T. Natarajan, and K. R. Rao, “Discrete cosine transform”, IEEE Transactions on Computers, vol. 23, no. 1, **(1974)** January, pp. 90-93.

[3] ISO/IEC, “Information Technology -- Generic Coding of Moving Pictures and Associated Audio Information, Part 2: Video”, ISO/IEC 13818-2: 2013, **(2013)** September 27.

[4] T. Ebrahimi and C. Horne, “MPEG-4 natural video coding - An overview”, Signal Processing: Image Communication, vol. 15, no. 4-5, **(2000)** January, pp. 365-385.

[5] Joint Video Team of ITU-T and ISO/IEC, “Information Technology -- Coding of Audio-Visual Objects -- Part 10: Advanced Video Coding”, ITU-T Rec. H.264 | ISO/IEC 14496-10: **(2014)** August 27.

[6] ISO/IEC, “Information Technology -- High Efficiency Coding and Media Delivery in Heterogeneous Environments, Part 2: High Efficiency Video Coding”, ISO/IEC 23008-2, (**2013**) November 25.

[7] X. H. Zhao, Z. L. Wang, and K. K. Zhao, “Research on distributed image compression algorithm in coal mine WMSN”, International Journal of Digital Content Technology and its Applications, vol. 5, no. 18, **(2011)** January, pp. 283-291.

[8] M. Ciznicki, M. Kierzynka, P. Kopta, K. Kurowski, and P. Gepner, “Benchmarking JPEG 2000 implementations on modern CPU and GPU architectures”, Journal of Computer Science, vol. 5, no. 2, **(2014)** March, pp. 90-98.

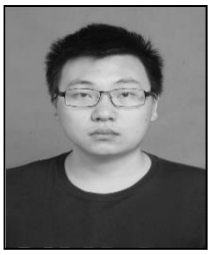
[9] C. G. Yan, Y. D. Zhang, J. Z. Xu, F. Dai, L. Li, Q. H. Dai, and F. Wu, “A highly parallel framework for HEVC coding unit partitioning tree decision on many-core processors”, IEEE Signal Processing Letters, vol. 21, no. 5, **(2014)** May, pp. 573-576.

[10] S. Tokdemir and S. Belkasim, “Parallel processing of DCT on GPU”, Proceedings of the Data Compression Conference, Snowbird, UT, USA, **(2011)** March 29-31, pp. 479.

[11] D. Liu and X. Y. Fan, “Parallel program design for JPEG compression encoding”, Proceedings of the 9th International Conference on Fuzzy Systems and Knowledge Discovery, Chongqing, China, **(2012)** May 29-31, pp. 2502-2506.

[12] NVIDIA Corporation: NVIDIA CUDA programming guide, available at http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf, **(2014)** April 15.

Authors



Rongyang Shan, was born in Anhui province, China in 1992. He received his B.E. degree in communication engineering from Shandong University, Weihai, China in 2014. Now he is pursuing his M.E. degree in signal and information processing in Shandong University, Weihai, China. His current research interests include digital image processing and analysis.

Chengyou Wang, was born in Shandong province, China in 1979. He received his B.E. degree in electronic information science and technology from Yantai University, China in 2004, and his M.E. and Ph.D. degree in signal and information processing from Tianjin University, China in 2007 and 2010 respectively. Now he is an associate professor and supervisor of postgraduate in the School of Mechanical, Electrical and Information Engineering, Shandong University, Weihai, China. His current research interests include digital image/video processing and analysis, multidimensional signal and information processing.

Wei Huang, was born in Inner Mongolia autonomous region, China in 1990. He received his B.E. degree in electronic information science and technology from Wuhan University, China in 2014. He is currently pursuing his M.E. degree in communication and information system at Wuhan University, China. His current research interests include communication technology and digital image processing.

Xiao Zhou was born in Shandong province, China in 1982. She received her B.E. degree in automation from Nanjing University of Posts and Telecommunications, China in 2003, her M.E. degree in information and communication engineering from Inha University, Korea in 2005, and her Ph.D. degree in information and communication engineering from Tsinghua University, China in 2013. Now she is a lecturer in the School of Mechanical, Electrical and Information Engineering, Shandong University, Weihai, China. Her current research interests include wireless communication technology, digital image processing and analysis.