International Journal of Multimedia and Ubiquitous Engineering
Vol.10, No.7 (2015), pp.295-304
http://dx.doi.org/10.14257/ijmue.2015.10.7.31

All Phase Biorthogonal Transform Based on GPU

Rongyang Shan, Chengyou Wang*, Xiao Zhou and Liping Wang

School of Mechanical, Electrical and Information Engineering, Shandong University, Weihai 264209, China
sdusry@163.com, wangchengyou@sdu.edu.cn, zhouxiao@sdu.edu.cn, sduwlp@163.com

Abstract

In this paper, all phase biorthogonal transform (APBT) based on parallel algorithm is proposed. It can solve two problems. First, block-based DCT transform coding has serious blocking artifacts when the image is highly compressed at low bit rate. Second, APBT can solve the problem about blocking artifacts, but it does not have a fast algorithm, it has a low efficiency when APBT applies to image processing. So APBT based on parallel algorithm can solve the above problems, and it provides more space for improving the processing speed of APBT. We use the CUDA toolkit based on GPU which is released by NVIDIA to design the parallel algorithm of APBT. Experimental results show that the maximum speedup ratio of parallel algorithm of APBT can reach more than 40 times with a very low version GPU, compared with conventional serial APBT. And the reconstructed image using the proposed algorithm has the same performance with the serial one in terms of objective quality and subjective effect. The proposed parallel algorithm based on GPU of APBT also can be used in image compression, video compression, the edge detection, and some other fields of image processing.

Keywords: *GPU; Parallel Computing; All Phase Biorthogonal Transform (APBT); Discrete Cosine Transform (DCT)*

1. Introduction

At present, the research for the discrete cosine transform (DCT) has been developed in-depth. Discrete cosine transform (DCT) [1] is widely used in the field of image processing. DCT is applied to the standards of the international image compression and video compression, like JPEG [2], MPEG-2 [3], MPEG-4 [4], H.264/AVC [5] and H.265/HEVC [6]. Although discrete wavelet transform (DWT) took the place of discrete cosine transform in JPEG 2000, DCT still takes an important place in image processing. With the development of orthogonal transform, DCT has become quite mature, and two-dimensional DCT is the core of JPEG coding. However, DCT is not the best choice in image coding, because block DCT transform coding has serious blocking artifacts when the image is highly compressed at low bit rate. The all phase biorthogonal transform (APBT) [7] which is based on Walsh-Hadamard transform (WHT), DCT and inverse discrete cosine transform (IDCT) proposed by Hou *et al.* is a new transform for image compression instead of DCT, which solves the problem of blocking artifacts in DCT, and APBT uses the uniform quantization step instead of the complex quantization table in DCT which makes APBT save the storage space of quantization Table.

Currently, parallel computation based on GPU is more and more popular in scientific computation, because GPU has more cores than CPU, and the parallel computation based on GPU can bring higher efficiency in complex computation than CPU. CUDA toolkit which is released by NVIDIA makes parallel computation based on GPU easier than before. Parallel algorithm can get 10 times acceleration easily than serial algorithm by

ISSN: 1975-0080 IJMUE

CUDA. CUDA and massively parallel GPU hardware are changing how we think about computation. No longer limited to performing one or a few operations at a time, CUDA programmers write programs that perform tens of thousands of operations simultaneously. In 2011, S. Tokdemir and S. Belkasim [8] used parallel DCT algorithm in data compression, and they got a satisfying result in efficiency and their study indicates a clear superiority of the GPU as parallel processor for image compression using DCT over the CPU. D. Liu and X. Y. Fan [9] designed parallel program for JPEG compression encoding 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. P. Holub *et al.* [10] proposed an approach to parallelization of JPEG compression and the use of auxiliary indexes for efficient decompression in 2013. Their approach to parallelization also provides sufficient performance even for the future generation of 8K video processing systems.

APBT solves the problem of blocking artifacts in DCT, and APBT uses uniform quantization step. So APBT has more advantages than DCT, but APBT does not have fast algorithm. Because the processing of APBT requires a large amount of calculation, and the pixels' data of image are independent, so APBT is very suitable for parallel computation. In this paper, parallel algorithm based on GPU is used in APBT for improving the efficiency.

The rest of this paper is organized as follows. Section 2 introduces CUDA. Section 3 starts with a brief review of DCT and APBT. Section 4 describes the design of APBT's parallel algorithm. Experimental results of the proposed method are presented in Section 5. Conclusions and remarks on possible further work are given finally in Section 6.

2. CUDA

At present, the operating frequency of processor has hit a clock rate limit at around 4 GHz, but for current technology, if we continue to increase the clock rate, we will get more heat and electric bill than efficiency. People are not able to improve the efficiency of computation by improving the frequency of processor, so parallel computation becomes more and more popular. With the increasing compute capability of GPU, GPU is not only a graphic card, but also it is being used in the field of computing. Although GPU's operation frequency is lower than CPU, GPU's Flops (floating-point operations per second) is much higher than CPU, because GPU has more ALU. GPU 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. In 2007, NVIDIA adds an easy-to-use programming interface to GPU which is called CUDA or Compute Unified Device Architecture. CUDA opened up the possibility to program GPUs without having to learn complex shader languages or to think only in terms of graphics primitives.

Figure 1 shows that the architecture of CUDA has three main components. It contains CUDA libraries, CUDA runtime API and CUDA device API. CUDA application is programmed by C-like language, developers do not need know what GPU is, and they can rewrite serial code what they code before in C language to parallel code easily. So they can get higher computational efficiency than serial code. But in order to get higher efficiency CUDA program, you would better know some basic knowledge about the GPU.

In the programming model of CUDA (as shown in Figure 2), CPU could be seen as host and GPU should be seen as device or the co-processor of CPU which has a memory. In this system, it can have one host and multiple devices. Under this model, host and devices work together and fulfill their proper function. The CPU is responsible for the strongly logical transactions and serial computing; while the GPU is focused on the implementation of highly threaded parallel processing tasks [11]. CPU and GPU have

their own memory and they can't visit each memory directly. 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.

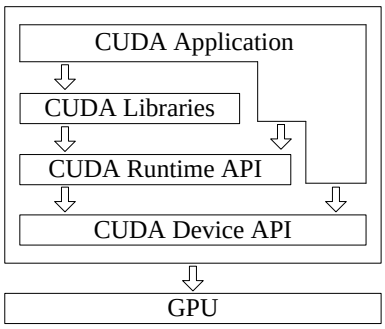


Figure 1. The Architecture of CUDA

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. The kernel function and the serial processing in host-side compose a complete CUDA 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, and the kernel function is used to compute what you need.

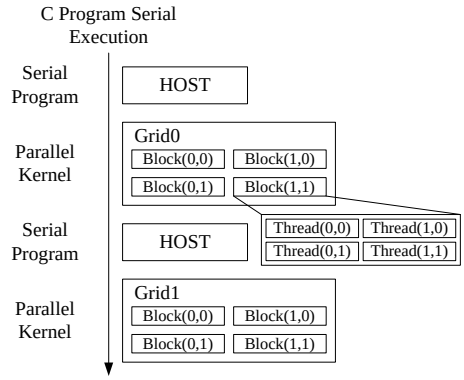


Figure 2. CUDA Program Model

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 [11]. Threads in the same block can communicate through shared memory, so developers would have more room for their program.

3. DCT and APBT

3.1 Discrete Cosine Transform (DCT)

The conventional two-dimensional DCT transform is made usually by two one-dimensional DCT transform on row and column directions separately [1]. $\boldsymbol{X}$ is $N \times N$ image block, and $\boldsymbol{C}$ represents DCT matrix with size of $N \times N$ respectively.

After two-dimensional DCT transform, transform coefficients block $\boldsymbol{Y}$ can be denoted by

$$\boldsymbol{Y} = \boldsymbol{C}\boldsymbol{X}\boldsymbol{C}^{\mathrm{T}}, \tag{1}$$

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

where $\boldsymbol{C}^{\mathrm{T}}$ is the transpose matrix of $\boldsymbol{C}$.

Since DCT is an orthogonal transform, i.e. $\boldsymbol{C}^{\mathrm{T}} = \boldsymbol{C}^{-1}$, we use

$$\boldsymbol{X} = \boldsymbol{C}^{-1}\boldsymbol{Y}(\boldsymbol{C}^{\mathrm{T}})^{-1} = \boldsymbol{C}^{-1}\boldsymbol{Y}(\boldsymbol{C}^{-1})^{\mathrm{T}} = \boldsymbol{C}^{-1}\boldsymbol{Y}C, \tag{3}$$

to reconstruct the image, where $\boldsymbol{C}^{-1}$ is the inverse matrix of $\boldsymbol{C}$.

3.2 All Phase Biorthogonal Transform (APBT)

On the basis of all phase digital filtering, three kinds of all phase biorthogonal transforms based on the WHT, DCT and IDCT were proposed and the matrices of APBT were deduced in [7]. Similar to DCT matrix, it can be used in image compression transforming the image from spatial domain to frequency domain too.

Taking all phase discrete cosine biorthogonal transform (APDCBT) for example, the process of two-dimensional APBT is introduced as follows. $\boldsymbol{X}$ is $N \times N$ image block, and $\boldsymbol{V}$ represents APDCBT matrix with size of $N \times N$ respectively. After two-dimensional APDCBT transform, transform coefficients block $\boldsymbol{Y}$ can be denoted by

$$\boldsymbol{Y} = \boldsymbol{V}\boldsymbol{X}\boldsymbol{V}^{\mathrm{T}}, \tag{4}$$

$$\boldsymbol{V}(m,n) = \begin{cases} \dfrac{N-m}{N^2}, & m = 0,1,\cdots,N-1,\ n = 0, \\ \dfrac{1}{N^2}\left[(N-m)\cos\dfrac{mn\pi}{N} - \csc\dfrac{n\pi}{N}\sin\dfrac{mn\pi}{N}\right], & m = 0,1,\cdots,N-1,\ n = 1,2,\cdots,N-1. \end{cases} \tag{5}$$

where $\boldsymbol{V}^{\mathrm{T}}$ is the transpose matrix of $\boldsymbol{V}$. We use

$$\boldsymbol{X} = \boldsymbol{V}^{-1}\boldsymbol{Y}(\boldsymbol{V}^{-1})^{\mathrm{T}}, \tag{6}$$

to reconstruct the image, where $\boldsymbol{V}^{-1}$ is the inverse matrix of $\boldsymbol{V}$.

4. The Design of Parallel APBT

The system of APBT is similar to the conventional DCT system, as Figure 3 shows, it contains: blocking, APBT, quantization, inverse quantization and inverse APBT. Through DCT or APBT, the energy of image is concentrated in the top left corner, DC coefficient is in the upper left corner, and others are AC coefficients. After quantization, the image can be compressed easily.

The essence of DCT, APBT and their inverse transform is the matrix multiplication, as shown in Eqs. (7) and (8).

$$\boldsymbol{Y} = \boldsymbol{T}\boldsymbol{X}\boldsymbol{T}^{\mathrm{T}}, \tag{7}$$

$$\boldsymbol{X} = \boldsymbol{T}^{-1}\boldsymbol{Y}(\boldsymbol{T}^{-1})^{\mathrm{T}}, \tag{8}$$

Where $\boldsymbol{X}$ is the two-dimensional image matrix, $\boldsymbol{Y}$ is the transform coefficients block, $\boldsymbol{T}$ is the transform matrix.

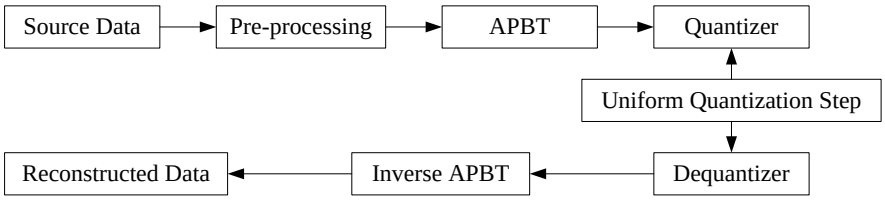


Figure 3. APBT System

In DCT system, transform coefficients block is quantified by quantization table. But APBT use uniform quantization step. They are quantified by

$$F_q = \left\lfloor \frac{F}{Q} \right\rceil, \tag{9}$$

Where F is the data before quantization, F_q is the data after quantization, Q is quantization table or uniform quantization step, $\lfloor \cdot \rceil$ is round of data.

When designing parallel program, the task needs to be divided into many subtasks. These subtasks are processed parallel for improving the computational efficiency. In the algorithm of APBT or DCT, the first step is the preprocessing of source image, which should be divided into some 8×8 sub-images, every sub-image has 64 pixels' data, then to do APBT or DCT. In the conventional algorithm of APBT or DCT, every sub-image is processed serially in transform, quantization, inverse quantization and inverse transform. This indicates that, these sub-images have no correlate with each other, so they can be processed independently. Therefore parallel algorithm is suitable for APBT and DCT. In CUDA program model, it has parallelism between blocks in grid and parallelism between threads in block, the image has similar correlations, sub-images in the image are independent, and pixels in the sub-image are independent too. Naturally, as shown in Figure 4, the image is mapped to the grid, every 8×8 sub-image is mapped to block. The source image will be divided into N blocks,

$$N = \frac{W \times H}{64}, \tag{10}$$

where N is the number of blocks in grid, W is the width of the source image, and H is the height of source image. Every block has 64 threads, and one thread is going to process one pixel's data. When the program is executed, the kernel starts. Then blocks are going to be distributed to each streaming multiprocessors (SM), SM is the complete core of GPU, the more SMs which GPU has, the higher compute capability GPU will get. Threads are going to be distributed to each SP, SP is a part of SM, and one SM has 8 SPs. Every block and thread can run independently. So in this model, every sub-image is transformed, quantized, inverse quantized and inverse transformed independently and in parallel, the efficiency of APBT is improved and the runtime of APBT is shorter.

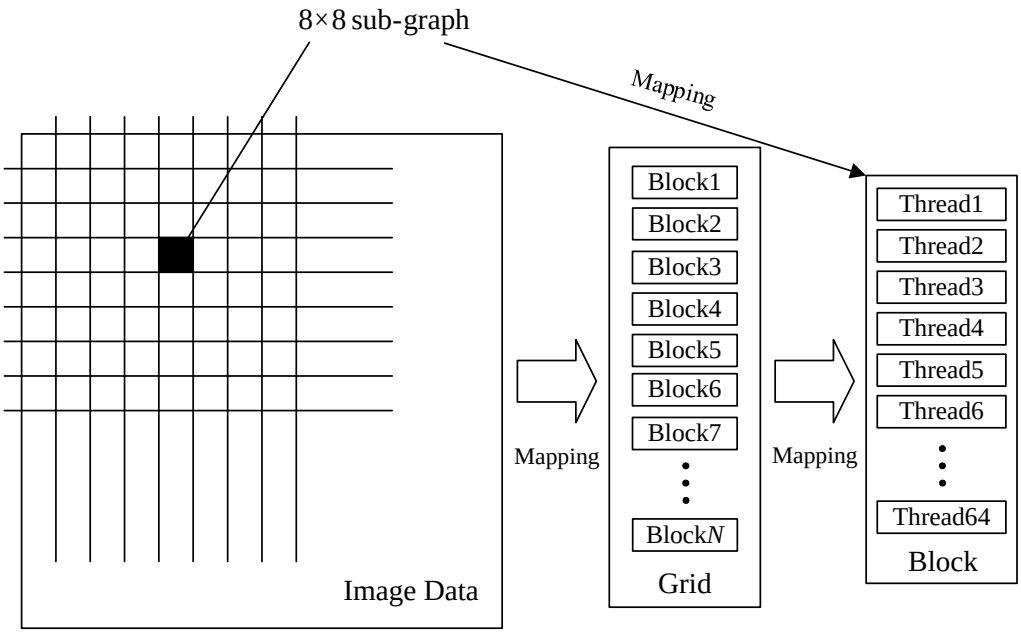


Figure 4. Correspondence Between Image Component and Grid

At the beginning of the program, CPU loads APBT coefficients matrix to the share memory in the GPU and copies image data from global memory to GPU memory. Every thread reads a pixel's data from GPU memory to the share memory which is good for the computational efficiency in parallel program, because share memory has higher read-write speed than global memory. After all phase biorthogonal transforms, every block gets an 8×8 transform coefficients block, and every thread has one transform coefficient, then every thread will quantize the data what they have. When the part of all phase biorthogonal transforms is running, the program needs to use the synchronous function to make sure that every thread in the block has completed their work, and then it will be allowed to do next part of tasks in the APBT system.

In this paper, we make the APBT algorithm execute on the GPU, when the block completes the APBT and quantization, it will do the inverse quantization and inverse APBT. After that we will get the reconstructed image.

5. Experimental Results and Analysis

In the experiments, Lena.bmp is selected as the source image, and it has different sizes from 128×128 to 1024×1024. We compare the runtime of APBT and DCT on CPU, the runtime of all phase biorthogonal transform and discrete cosine transform on GPU and the efficiency of serial APBT algorithm and parallel APBT algorithm. We also compare the reconstructed image which comes from serial APBT algorithm and parallel APBT algorithm; through the image's PSNR and subjective evaluation to ensure APBT algorithm and parallel APBT algorithm have the same result. The test platform is shown in Table 1.

Table 1. C Test Platform

Hardware		Software	
CPU	Intel Core2 Duo E7500	Operating System	Windows 8.1 (64bit)
	Frequency: 2.93GHz		
RAM	2G		

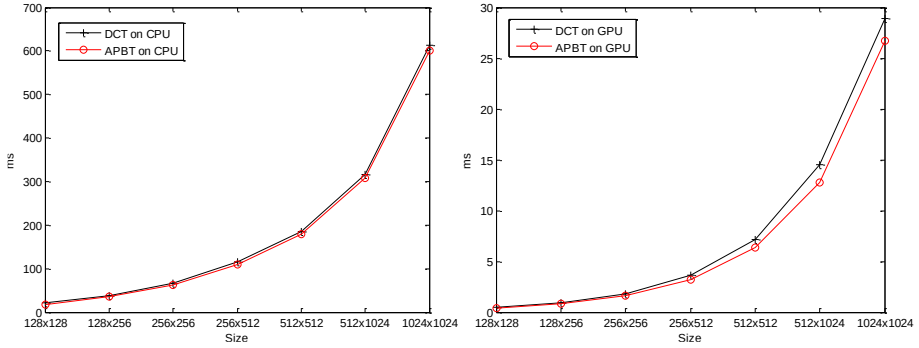
GPU	GeForce 425M	Software Environment	CUDA SDK 5.5
	Clock Rate:1.12GHz		CUDA Toolkit 5.5
	Compute Capability: 2.1		Microsoft Visual Studio 2012
	SP: 96		MATLAB R2012b

In this paper, we integrate qualitative analysis and quantitative analysis, making the result more scientific, objective and fair. Taking the image Lena the size of which is 512×512 as an example, after the serial APBT algorithm and the parallel APBT algorithm, we can get their reconstructed image. When the serial APBT algorithm and the parallel APBT algorithm have the same uniform quantization step, they have the same PSNR, and the reconstructed images have the same result in subjective effects (as shown in Figure 5).

(a) (b)

Figure 5. The Reconstructed Image: (a) On CPU, (b) On GPU

In Figure 6, we compare the runtime of APBT and DCT which use serial algorithm, we conclude that the efficiency of serial APBT algorithm is higher than DCT (as shown in Figure 6 (a)), because the APBT algorithm is quantified by uniform quantization step instead of complex quantization table in DCT. When the size of the image increases, the runtime of APBT and DCT increase too. We also compare the runtime of APBT and DCT which are run on GPU, the efficiency of parallel APBT algorithm is higher than parallel DCT with the same reason, when the size of the image increases, the runtime increases with the almost same proportion (as shown in Figure 6 (b)).



(a) (b)

Figure 6. The Running Time of DCT and APBT in Different Platforms: (a) DCT and APBT on CPU, (b) DCT and APBT on GPU

From the experimental results in Figure 7, we can know that the efficiency of parallel APBT algorithm is much higher than serial APBT algorithm, the parallel APBT that runs on GPU could gain at least 25 times acceleration, and the maximum speedup ratio can reach more than 40 times, so the computational efficiency of parallel computing is very impressive in general. Parallel computing based on GPU can process dozens of images at the same time, the efficiency of all phase biorthogonal transforms is greatly improved.

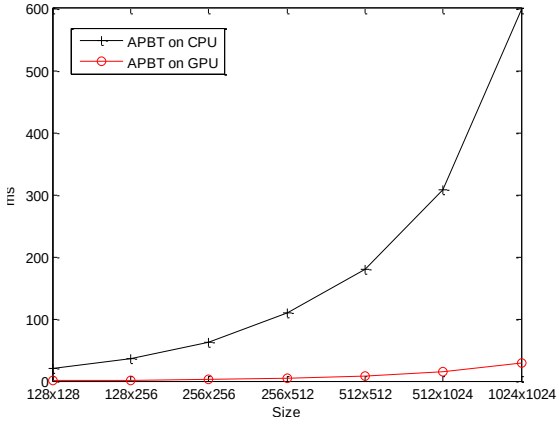


Figure 7. The Running Time of Parallel APBT and Serial APBT

6. Conclusion

In this paper we use parallel computing base on GPU in APBT algorithm, through comparing the runtime of parallel APBT algorithm, serial APBT algorithm, parallel DCT algorithm and serial DCT algorithm, the parallel computing gains higher efficiency than serial computing. The time of APBT algorithm is shorter than before, the efficiency of APBT is greatly improved and the quality of reconstructed image is same as serial APBT. We gained more than 40 times acceleration than conventional APBT, which makes up the shortcoming of APBT which has no fast algorithm, compared with DCT algorithm, and better objective quality and subjective effects are achieved at low bit rate. So the parallel APBT algorithm solves the efficiency of conventional APBT algorithm and the blocking artifacts in DCT.

Acknowledgements

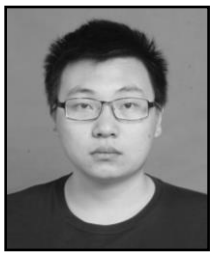
This work was supported by the promotive research fund for excellent young and middle-aged scientists of Shandong Province, China (Grant No. BS2013DX022) and the National Natural Science Foundation of China (Grant No. 61201371). The authors would like to thank Xiaoyan Wang and Fanfan Yang for their kind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

References

[1] N. Ahmed, T. Natarajan, and K. R. Rao, “Discrete cosine transform,” IEEE Transactions on Computers, vol. 23 no. 1, January **(1974)**, pp. 90-93.
[2] ISO/IEC, “Information Technology -- Digital Compression and Coding of Continuous-tone Still Images -- Part 1: Requirements and Guidelines”, ISO/IEC 10918-1:1994 | ITU-T Rec. T.81, September 22, **(2011)**.
[3] ISO/IEC, “Information Technology -- Generic Coding of Moving Pictures and Associated Audio Information -- Part 2: Video,” ISO/IEC 13818-2:2013, September 27, **(2013)**.
[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 14496-10:2012, **(2014),** January 28.
[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, **(2013)**.
[7] Z. X. Hou, C. Y. Wang and A. P. Yang, "All phase biorthogonal transform and its application in JPEG-like image compression", Signal Processing : Image Communication, vol. 24 no. 10, **(2009)** November, pp. 791-802.
[8] S. Tokdemir and S. Belkasim, "Parallel processing of DCT on GPU", Proceedings of the Data Compression Conference, Snowbird, UT, USA, March 29-31, **(2011)**, pp. 479.
[9] 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.
[10] P. Holub, M. Srom, M. Pulec, J. Matela and M. Jirman, "GPU-accelerated DXT and JPEG compression schemes for low-latency network transmissions of HD, 2K, and 4K video," Future Generation Computer Systems, vol. 29 no. 8, **(2013)** October, pp. 1991-2006.
[11] "NVIDIA Corporation: NVIDIA CUDA programming guide," available at http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf, **(2014),** March 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 research interests concentrate on image processing and transmission techniques.

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 in the School of Mechanical, Electrical and Information Engineering, Shandong University, Weihai, China. His current research interests include image processing and transmission techniques, multidimensional signal and information processing, and smart grid technology.

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, image processing and transmission technology.

Liping Wang, was born in Shandong province, China in 1990. She received her B.E. degree in communication engineering from Shandong University, Weihai, China, in 2014. Now she is pursuing her M.E. degree in signal and information processing in Shandong University, Weihai, China. Her research interests concentrate on image processing and transmission technniques.