All Phase Discrete Sine Biorthogonal Transform and Its Application in JPEG-like Image Coding Using GPU

Rongyang Shan[#], Xiao Zhou[#*], Chengyou Wang, and Baochen Jiang
School of Mechanical, Electrical and Information Engineering, Shandong University, Weihai 264209, China
[e-mail: sdusry@163.com, zhouxiao@sdu.edu.cn, wangchengyou@sdu.edu.cn, jbc@sdu.edu.cn]
[#]These authors contributed equally to this work and should be considered co-first authors.
*Corresponding author: Xiao Zhou

Received February 1, 2016; revised June 6, 2016; accepted July, 2016; published September 30, 2016

Abstract

Discrete cosine transform (DCT) based JPEG standard significantly improves the coding efficiency of image compression, but it is unacceptable event in serious blocking artifacts at low bit rate and low efficiency of high-definition image. In the light of all phase digital filtering theory, this paper proposes a novel transform based on discrete sine transform (DST), which is called all phase discrete sine biorthogonal transform (APDSBT). Applying APDSBT to JPEG scheme, the blocking artifacts are reduced significantly. The reconstructed image of APDSBT-JPEG is better than that of DCT-JPEG in terms of objective quality and subjective effect. For improving the efficiency of JPEG coding, the structure of JPEG is analyzed. We analyze key factors in design and evaluation of JPEG compression on the massive parallel graphics processing units (GPUs) using the compute unified device architecture (CUDA) programming model. Experimental results show that the maximum speedup ratio of parallel algorithm of APDSBT-JPEG can reach more than 100 times with a very low version GPU. Some new parallel strategies are illustrated in this paper for improving the performance of parallel algorithm. With the optimal strategy, the efficiency can be improved over 10%.

Keywords: Image coding, parallel computing, GPU, all phase discrete sine biorthogonal transform (APDSBT), discrete sine transform (DST)

This work was supported by the Natural Science Foundation of Shandong Province, China (Grant No. ZR2015PF004), the National Natural Science Foundation of China (Grant No. 61201371), the promotive research fund for excellent young and middle-aged scientists of Shandong Province, China (Grant No. BS2013DX022), and the Fundamental Research Funds of Shandong University (Grant No. 2014ZQXM008).

http://dx.doi.org/10.3837/tiis.2016.09.024 **ISSN : 1976-7277**

1. Introduction

In image and video compression standards, like JPEG [1], MPEG-2, MPEG-4, H.264/AVC [2] and H.265/HEVC [3], discrete cosine transform (DCT) [4] is the core transform of these compression standards, which requires intensive and complex computations. In JPEG standard, the coding efficiency of image compression is improved significantly. However, the serious blocking artifacts caused by block-based DCT and low efficiency of high-definition image are unacceptable. To solve the above problems, some transforms have been proposed in recent years. Based on all phase digital filter (APDF) theory [5, 6], Hou *et al*. proposed all phase biorthogonal transform (APBT) [7] which can be used in JPEG scheme instead of the conventional DCT. Three matrices of APBT were deduced based on Walsh-Hadamard transform (WHT), DCT, and inverse DCT (IDCT) called APWBT, APDCBT, and APIDCBT, respectively. Experimental results show that the coding performance is improved. Another advantage is that the transform coefficients can be quantized uniformly. It could save the storage space of quantization table and reduce the computational complexity. In 2015, Fu *et al*. proposed windowed all phase biorthogonal transform (WAPBT) [8] based on windowed all phase digital filter (WAPDF) [9]. It attempts to find the best window function for most images, although it improves the quality of reconstructed image, the time and computational complexity of algorithm increase dramatically. Besides, there are other solutions focused on the content of image. Shape-adaptive DCT (SA-DCT) uses different compression ratios to different regions, the region of interest (ROI) is coded with low compression ratio, while background area is coded with high compression ratio [10]. But the ROI is hard to choose, and how to split the ROI effectively is still a problem.

Apart from DCT, there are many transforms, such as DFT, WHT, and IDCT which are also used in the field of image/video processing. Discrete sine transform (DST) has received considerable interest recently. Especially in the latest video compression standard H.265 [3], DST is adopted to replace DCT in the 4×4 luma residual blocks for intra picture prediction modes [11]. In terms of complexity, the 4×4 DST-style transform is not much more computationally demanding than the 4×4 DCT-style transform, and it provides approximately 1% bit-rate reduction in intra picture predictive coding [12].

Inspired by the DST and APBT, applying APDF theory [5, 6] to DST, we propose a new transform named all phase discrete sine biorthogonal transform (APDSBT) in this paper. With the new transform, the energy of the coefficients is more concentrated than DCT transform, so in Huffman coding part of image compression, it can bring a better performance in removing redundancy than DCT transform. Similar to APWBT, APDCBT, and APIDCBT, APDSBT can also be used in JPEG-like image coding scheme. Experimental results in the following show that APDSBT-based JPEG can achieve better performance than DCT-based JPEG (DCT-JPEG). On the subjective evaluation, the reconstructed image using the proposed algorithm performs better than that with DCT-JPEG. The blocking artifacts which appear in reconstructed image at low bit rate are reduced significantly. On the objective quality evaluation, the PSNR of the proposed APDSBT-JPEG is higher than that of DCT-JPEG. Besides, the quantization step is simple in the proposed algorithm in which the uniform quantization step is used instead of the complex quantization table used in DCT-JPEG.

In image processing, the real-time performance is a standing challenge. For improving the accuracy and performance of algorithm, sometimes the intensive computational power is necessary. How to improve the efficiency of algorithm is one of the main problems in image

processing. Methods for implementation involve the using of graphic processing units (GPU) or digital signal processors (DSP) [13]. In multimedia transmission system, for improving bandwidth utilization [14], the image and video transform system should be compressed by some compression standard. However, these algorithms are too complex to process the image data in real-time. Considering the massive computational cost of high-definition image or video, the serial algorithm based on central processing units (CPU) has been unable to satisfy the real-time requirement. However, GPU can handle this easily by adopting highly parallel architecture and operating at high frequency.

Recently, GPU has evolved into an extremely powerful computation resource [15]. Prior to 2003, the better computation efficiency could be obtained by improving the performance of processor. Researchers could get their algorithm performance improved easily by increasing CPU's frequency. At present, the operating frequency of processor has hit a clock rate limit around 4 GHz. From architectural point of view, the technology faced bottle-neck, and it is very difficult to improve performance still by improving CPU's frequency. Furthermore, if the clock rate is increased continuously, the benefit from computing efficiency can no longer cover the required electricity expenses. In order to obtain higher efficiency, multi-core technology has been introduced to CPU, and which can be easily found nowadays. Even the smallest modern computers, such as phones, are supported by the multi-core technology. For example, Apple A9 is a 64-bit system on chip (SoC) with four cores. Qualcomm Snapdragon 810 has eight cores and MediaTek helio x20 has ten cores. Some researches of parallel computing are based on many-core processors [16, 17]. Yan and Zhang proposed a parallel framework to decouple motion estimation (ME) for different partitions on many-core processors, and compared with serial execution. Their work achieves more than 30 and 40 times speedup for 1920×1080 and 2560×1600 video sequences. Parallel algorithm based on GPU also performs well. Parallel computation can improve the efficiency of the algorithm, and improve the real-time property of the system.

Although CPU can continue to increase computational capacity by using multi-core technology, GPU has more computational power than CPU. Therefore, hybrid architecture of CPU and GPU is becoming more and more popular [18]. In today's computer systems, heterogeneous architecture exists in servers and desktop computers, and also extends to portable and hand held devices. Even the technology can be found in some supercomputer. Titan supercomputer is the second-world's fastest computer, located at the Oak Ridge national laboratory in Tennessee, which employs AMD Opteron CPUs in conjunction with Nvidia Tesla GPUs to improve computational power. Due to the development of multi-core CPU and GPU, the image processing based on parallel computation is becoming a hot area. Compute unified device architecture (CUDA) toolkit is released in 2007 by Nvidia, which makes parallel computation based on GPU easier than before [19]. GPU has more cores than CPU, and it can launch ten thousand threads at the same time, so the algorithm based on GPU can further improve the parallel degree of program. In the field of image processing, the parallel algorithm based on GPU is used widely [20-22]. Liu and Fan [23] 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. However, they did not make all parts of JPEG algorithm run on GPU. Holub *et al*. [24] proposed that GPU-accelerated DXT for low-latency network transmissions of HD, 2K, and 4K video in 2013. In 2014, Alqudami and Kim developed an optimized parallel implementation of the forward DCT algorithm using OpenCL [25]. Shen *et al*. [26] implemented color space conversion for MPEG video encoding on GPU using DirectX to achieve 2~3 times acceleration.

We also focus on how JPEG image compression schemes can be efficiently implemented by using parallel computation based on GPU and used for low-latency high-resolution multimedia transmission over commodity networks. We leverage the computing capabilities of many-core CPU and GPUs. In this paper, the formula of APDSBT is deduced and the parallel computation based on GPU is used to accelerate the APDSBT based JPEG.

The rest of this paper is organized as follows. Section 2 introduces the background, a brief introduction to DCT-JPEG and CUDA. Section 3 starts with a brief review of APDF and then the APDSBT is deduced in detail. Section 4 is the design of parallel algorithm of APDSBT-JPEG system and some accelerating optimization strategies are discussed. 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. Background

2.1 JPEG: DCT-Based Image Compression Standard

The conventional 2-D DCT is always implemented separately by two 1-D DCTs. Let us use $\boldsymbol{X}$ and $\boldsymbol{C}$ to denote an image block and the DCT matrix with size of $N \times N$, respectively. After the conventional 2-D DCT, the transform coefficient block $\boldsymbol{Y}$ can be expressed as $\boldsymbol{Y} = \boldsymbol{C}\boldsymbol{X}\boldsymbol{C}^{\mathrm{T}}$, where $\boldsymbol{C}^{\mathrm{T}}$ is the transpose matrix of $\boldsymbol{C}$,

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

Since DCT is an orthogonal transform, i.e. $\boldsymbol{C}^{\mathrm{T}} = \boldsymbol{C}^{-1}$, we use $\boldsymbol{X} = (\boldsymbol{C}^{-1})\boldsymbol{Y}(\boldsymbol{C}^{-1})^{\mathrm{T}} = \boldsymbol{C}^{\mathrm{T}}\boldsymbol{Y}\boldsymbol{C}$ to reconstruct the image.

The baseline JPEG system based on DCT is shown in **Fig. 1**. The encoder mainly contains four parts: DCT, quantization, Zig-zag ordering, and Huffman coder. At the beginning of JPEG coding, the source image is divided into 8×8 sub-images. Then every sub-image is processed by DCT. After DCT, every sub-image gets an 8×8 matrix which includes 64 DCT coefficients. The upper left corner coefficient is called DC coefficient, and the other 63 coefficients are AC coefficients. According to the visual characteristics of human eyes which is sensitive to the low frequency component, but not sensitive to the high one, some AC coefficients can be dropped by quantization table for removing redundancy. As for coefficient matrix of every sub-image, DC coefficient is coded by differential pulse code modulation (DPCM), and AC coefficients are put in a one-dimensional array with Zig-zag ordering. After that, Huffman coding is used for image compression.

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

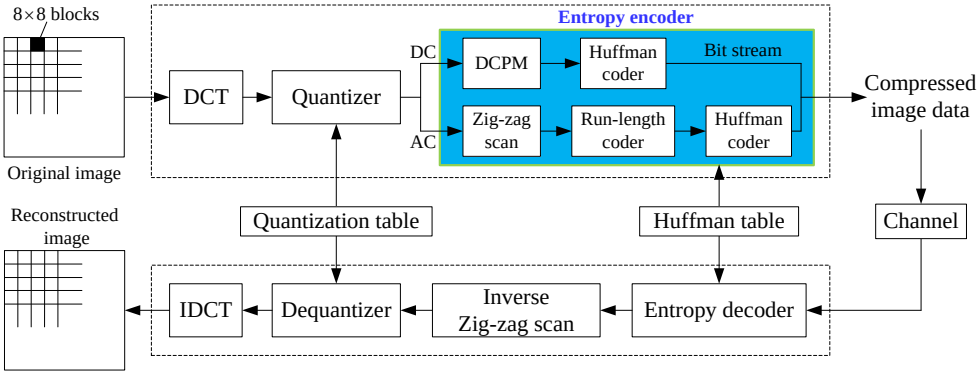


Fig. 1. DCT-JPEG system

2.2 CUDA

The data of image and video is too much to be processed in real-time. Therefore, parallel computation is necessary in image and video processing. In parallel computation, there are mainly two methods to parallel a program: task based parallelism and data based parallelism. For task based on parallelism, the task is divided into some independent tasks. Every task is processed by a thread. For data based parallelism, the data is mapped into threads, such as image processing, every pixel can be mapped into a thread. Because threads are parallel, the algorithm is executed parallel.

CUDA is an easy-to-use programming interface which is added to graphics card by NVIDIA in 2007. In CUDA, it has three main components: CUDA libraries, CUDA runtime API, and CUDA device API. Due to the interface of CUDA, researchers could parallel their algorithm easier on GPU. In the programming model of CUDA displayed in **Fig. 2**, it has two parts: the host and the devices. CPU can be regard as host and GPUs should be seen as device. They both have their own memory, and the host memory could not be visited by device directly [25]. The data which will be processed in parallel should be copied from host memory to device memory, and copied back in the end of the program. Under this model, CPU and GPUs work together and fulfill their proper function. The CPU's responsibility is to prepare data and driving kernel, while the GPU is focused on the implementation of highly threaded parallel processing tasks.

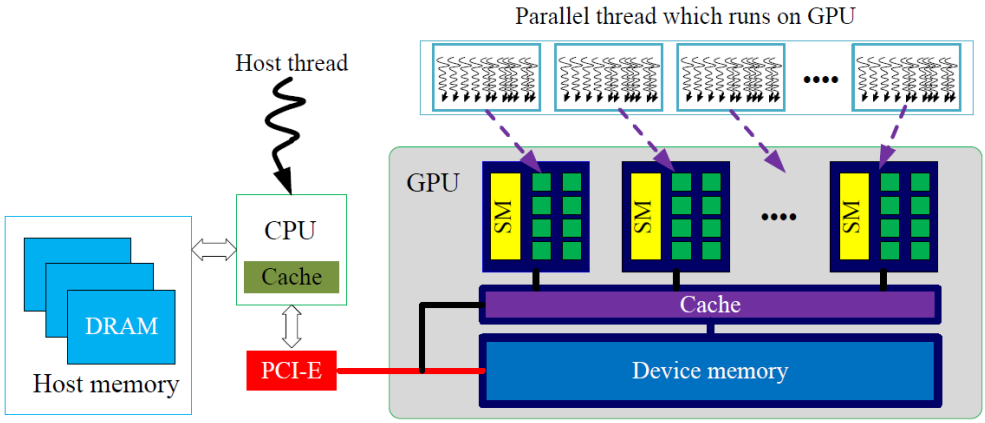


Fig. 2. CUDA program model

After developers analyzing the algorithm, they give the part of program which requires parallel computing to GPU. The function which runs on GPU for parallel computing is called

kernel. Kernel is not a complete and independent 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 CUDA program. 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.

As shown in **Fig. 3**, 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 [27]. Threads in the same block can communicate through shared memory. So developers would have more room for their program.

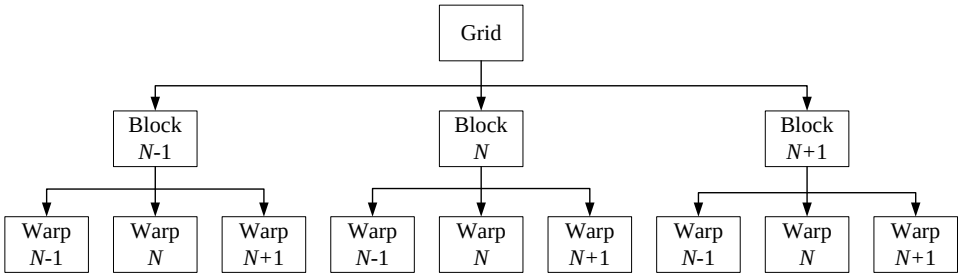


Fig. 3. The form of threads which was organized in GPU

3. All Phase Digital Filtering and APDSBT

3.1 All Phase Digital Filtering

All phase digital filtering can be considered to an application of overlap filtering. For a digital sequence $\{x(n)\}$, there are N vectors which are N-dimensional. Each vector contains $x(n)$ and has different intercept phases:

$$\begin{aligned} \boldsymbol{X}_0 &= [x(n), x(n+1), \cdots, x(n+N-1)]^{\mathrm{T}}, \\ \boldsymbol{X}_1 &= z^{-1}\boldsymbol{X}_0 = [x(n-1), x(n), \cdots, x(n+N-2)]^{\mathrm{T}}, \\ &\vdots \\ \boldsymbol{X}_{N-1} &= z^{-(N-1)}\boldsymbol{X}_0 = [x(n-N+1), x(n-N+2), \cdots, x(n)]^{\mathrm{T}}, \end{aligned} \tag{2}$$

where z^{-j} ($j = 0,1,\cdots,N-1$) is the delay operator. Obviously, $x(n)$ is the intersection of $\boldsymbol{X}_i (i = 0,1,\cdots,N-1)$, that is $x(n) = \boldsymbol{X}_0 \cap \boldsymbol{X}_1 \cap \cdots \cap \boldsymbol{X}_{N-1}$. According to the conventional representation of data matrices, the all phase data matrix of $x(n)$ is defined as $\boldsymbol{A}_N(n) = [\boldsymbol{X}_0, \boldsymbol{X}_1, \cdots, \boldsymbol{X}_{N-1}]$, and it is spanned by the column vectors $\boldsymbol{X}_i (i = 0,1,\cdots,N-1)$.

3.2 Proposed APDSBT

The all phase digital filter based on DFT proposed in 2003 is a new scheme for 1-D digital FIR filter [5]. The performance of APDF is superior to that of conventional filter. With the development of APDF theory, some further works were proposed recently, like APBT [7] and WAPBT [8]. In this paper, we propose a new transform called APDSBT based on DST.

Similar to APDF based on DCT [28], the process of 1-D signal passing through the APDF based on DST is shown in **Fig. 4**.

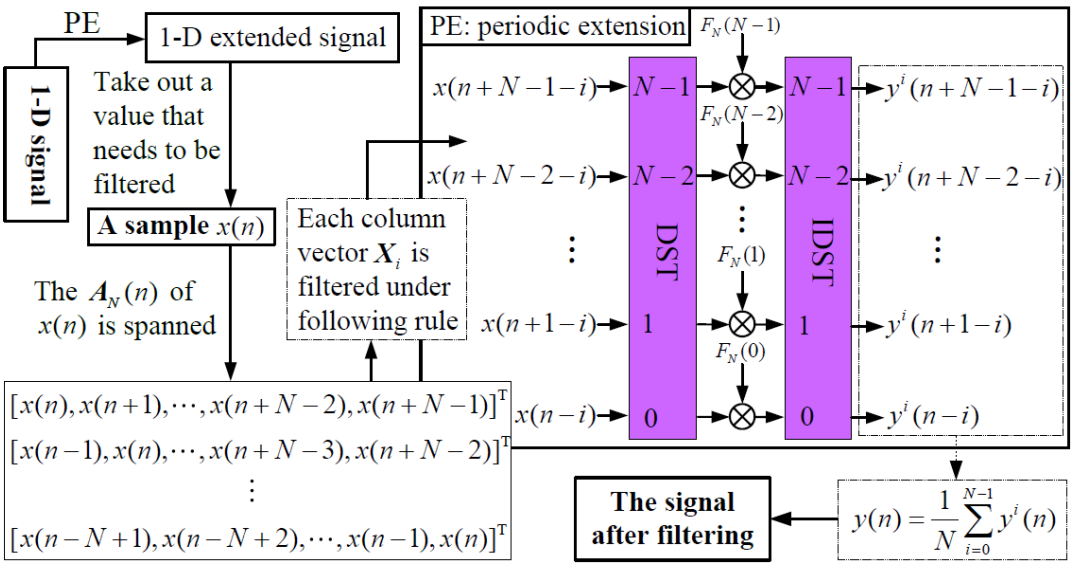


Fig. 4. 1-D signal flowing through the APDF based on DST

In APDF based on DST, $\boldsymbol{F}$ is an N-D expected sequency response vector, where sequency can be regarded as "extensive frequency" in a broad sense. DST matrix is represented as $\boldsymbol{S}$ and it is defined in Eq. (4). Since DST is an orthogonal transform, the IDST matrix is $\boldsymbol{S}^{-1}=\boldsymbol{S}^{\mathrm{T}}$.

$$\boldsymbol{F}=[F_N(0),F_N(1),\cdots,F_N(N-1)]^{\mathrm{T}}. \tag{3}$$

$$S(i,j)=\frac{2}{\sqrt{2N+1}}\sin\frac{(2i+1)(j+1)\pi}{2N+1},\quad i,j=0,1,\cdots,N-1. \tag{4}$$

The relation between input signal $x(n)$ and output signal $y(n)$ in DST-based APDF is shown in **Fig. 5**.

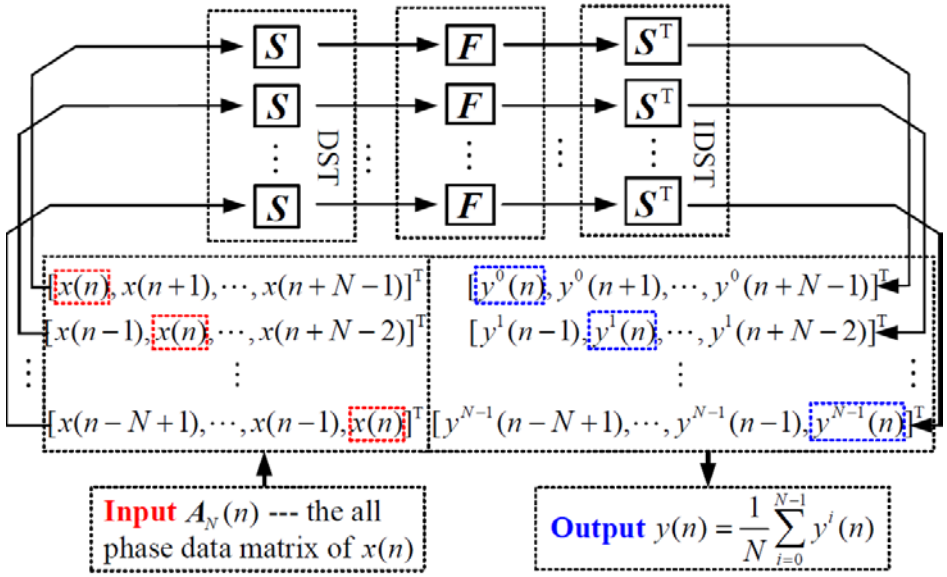


Fig. 5. Relation between input signal $x(n)$ and output signal $y(n)$ in APDF based on DST

In order to describe the APDF based on DST more clearly, in the following, the way to design APDF based on DST will be introduced mathematically. Denote $\boldsymbol{X}_i$ $(i=0,1,\cdots,N-1)$ be the i-th column vector of the all phase data matrix of $x(n)$. After $\boldsymbol{X}_i$ is filtered, the value of $y^i(n)$ is obtained:

$$y^i(n)=\boldsymbol{e}_i^{\mathrm{T}}\{\boldsymbol{S}^{\mathrm{T}}[\boldsymbol{F}\cdot(\boldsymbol{S}\boldsymbol{X}_i)]\}, \tag{5}$$

where the mark "$\cdot$" represents dot product operation. $\boldsymbol{e}_i(i=0,1,\cdots,N-1)$ is the i-th N-D column vector. The i-th element in vector $\boldsymbol{e}_i$ is the value 1 and the rest elements are the value 0.

$$\boldsymbol{E}=[\boldsymbol{e}_0,\boldsymbol{e}_1,\cdots,\boldsymbol{e}_{N-1}], \tag{6}$$

According to Eqs. (2)~(6), the output $y(n)$ can be expressed as:

$$\begin{aligned} y(n)&=\frac{1}{N}\sum_{i=0}^{N-1}y^i(n)=\frac{1}{N}\sum_{i=0}^{N-1}\{\boldsymbol{e}_i^{\mathrm{T}}\{\boldsymbol{S}^{\mathrm{T}}[\boldsymbol{F}\cdot(\boldsymbol{S}\boldsymbol{X}_i)]\}\}\\ &=\frac{1}{N}\sum_{i=0}^{N-1}[\sum_{j=0}^{N-1}(\sum_{k=0}^{N-1}F_N(k)S^{\mathrm{T}}(i,k)S(k,j))]x(n-i+j)\\ &=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}[H(i,j)x(n-i+j)], \end{aligned} \tag{7}$$

where

$$H(i,j)=\frac{1}{N}\sum_{m=0}^{N-1}F_N(m)S^{\mathrm{T}}(i,m)S(m,j)=\frac{1}{N}\sum_{m=0}^{N-1}F_N(m)S(m,i)S(m,j). \tag{8}$$

Then Eq. (7) can be rewritten as the convolution form of signal filtering:

$$\begin{aligned} y(n)&=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}[H(i,j)x(n-i+j)]=\sum_{i=\tau}^{\tau+N-1}\sum_{\tau=-(N-1)}^{N-1}[H(i,i-\tau)x(n-\tau)]\\ &=\sum_{\tau=-(N-1)}^{N-1}[\sum_{i=\tau}^{\tau+N-1}H(i,i-\tau)]x(n-\tau)=\sum_{\tau=-(N-1)}^{N-1}h(\tau)x(n-\tau)=h(n)*x(n). \end{aligned} \tag{9}$$

Therefore, the unit impulse response $h(\tau)$ is obtained and can be expressed by:

$$h(\tau)=\begin{cases}\sum_{i=\tau}^{N-1}H(i,i-\tau), & \tau=0,1,\cdots,N-1,\\ \sum_{i=0}^{\tau+N-1}H(i,i-\tau), & \tau=-1,-2,\cdots,-N+1.\end{cases} \tag{10}$$

Since $\boldsymbol{H}$ is a symmetric matrix: $H(i,j)=H(j,i)$, according to Eqs. (8) and (10), we have:

$$\begin{aligned} h(\tau)&=\sum_{i=\tau}^{N-1}H(i,i-\tau)=\frac{1}{N}\sum_{i=\tau}^{N-1}\sum_{m=0}^{N-1}[F_N(m)S(m,i)S(m,i-\tau)]\\ &=\sum_{m=0}^{N-1}V(\tau,m)F_N(m). \end{aligned} \tag{11}$$

Eq. (11) can be expressed in matrix format: $\boldsymbol{h}=\boldsymbol{V}\boldsymbol{F}$. The transition matrix $\boldsymbol{V}$ establishes the connection between unit pulse response in time domain and sequency response in orthogonal transform domain. We call it APDSBT matrix. Similar to DST matrix $\boldsymbol{S}$, it can be used in image compression transforming the image from spatial domain to frequency domain too. The elements of $\boldsymbol{V}$ can be calculated by:

$$V(\tau,m)=\frac{1}{N}\sum_{i=\tau}^{N-1}S(m,i)S(m,i-\tau)=\frac{1}{N}\sum_{i=0}^{N-1-\tau}S(m,i)S(m,i+\tau)\,. \tag{12}$$

Making variable substitution: $i\to l,\ \tau\to i,\ m\to j$, Eq. (12) can be rewritten as:

$$V(i,j)=\frac{1}{N}\sum_{l=0}^{N-1-i}S(j,l)S(j,l+i)\,. \tag{13}$$

Substituting Eq. (4) into Eq. (13), the APDSBT matrix with size of $N\times N$ is obtained:

$$V(i,j)=\begin{cases}\dfrac{1}{N}, & i=0,\ j=0,1,\cdots,N-1,\\ \dfrac{4}{N(2N+1)}\displaystyle\sum_{l=0}^{N-1-i}\left[\sin\frac{(2j+1)(l+1)\pi}{2N+1}\sin\frac{(2j+1)(l+i+1)\pi}{2N+1}\right], & \begin{array}{l}i=1,2,\cdots,N-1,\\ j=0,1,\cdots,N-1.\end{array}\end{cases} \tag{14}$$

For example, when $N=8$,

$$\boldsymbol{V}=\begin{bmatrix}0.1250 & 0.1250 & 0.1250 & 0.1250 & 0.1250 & 0.1250 & 0.1250 & 0.1250\\ 0.1083 & 0.0927 & 0.0635 & 0.0248 & -0.0182 & -0.0598 & -0.0943 & -0.1171\\ 0.0889 & 0.0407 & -0.0297 & -0.0842 & -0.0925 & -0.0481 & 0.0277 & 0.0971\\ 0.0684 & -0.0095 & -0.0767 & -0.0560 & 0.0299 & 0.0817 & 0.0336 & -0.0713\\ 0.0484 & -0.0415 & -0.0543 & 0.0345 & 0.0588 & -0.0295 & -0.0616 & 0.0453\\ 0.0305 & -0.0488 & -0.0011 & 0.0496 & -0.0302 & -0.0300 & 0.0538 & -0.0237\\ 0.0158 & -0.0358 & 0.0296 & -0.0021 & -0.0270 & 0.0386 & -0.0283 & 0.0092\\ 0.0054 & -0.0149 & 0.0210 & -0.0226 & 0.0197 & -0.0139 & 0.0072 & -0.0020\end{bmatrix}. \tag{15}$$

3.3 APDSBT-JPEG Image Coding Scheme

Similar to DCT-JPEG, the APDSBT-JPEG mainly has four parts: transform and inverse transform, encoding and decoding. The proposed scheme of APDSBT-JPEG image codec is shown in **Fig. 6**. The peculiarity of APBT-JPEG, the uniform quantization step, is inherited by the proposed scheme. With the uniform quantization step, the computational complexity is reduced.

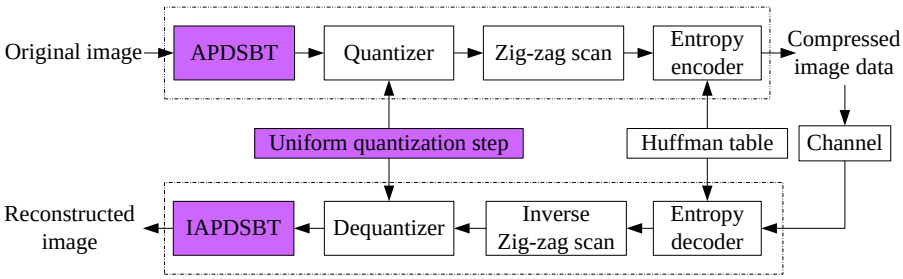


Fig. 6. The proposed scheme of APDSBT-JPEG image codec

4. Parallel JPEG-like Image Coding Based on APDSBT

4.1 Design of Parallel APDSBT-JPEG

For JPEG compression based on GPU, due to the independence of image data, the JPEG compression and decompression are quite suitable for a highly parallel GPU architecture. But

some operations could not be processed in parallel. Therefore, the task should be split between CPU and GPU. The APDSBT-JPEG encoder and decoder mainly comprise eight steps: APDSBT and inverse APDSBT (IAPDSBT), quantization and dequantization, Zig-zag ordering and inverse Zig-zag scanning, Huffman encoding and decoding. All these operations can be processed in parallel by GPU, as shown in **Fig. 7**. But some operations should be processed serially by CPU, like image data transmission. In heterogeneous architecture, the memories of CPU and GPU are independent. We call them the CPU's memory, host memory, and GPU's memory, graphics memory, respectively. They could not access each other. For parallel computing, CPU needs to load the data from hard disk to host memory. Then the data will be transferred to graphics memory through PCI-E bus by CUDA interface.

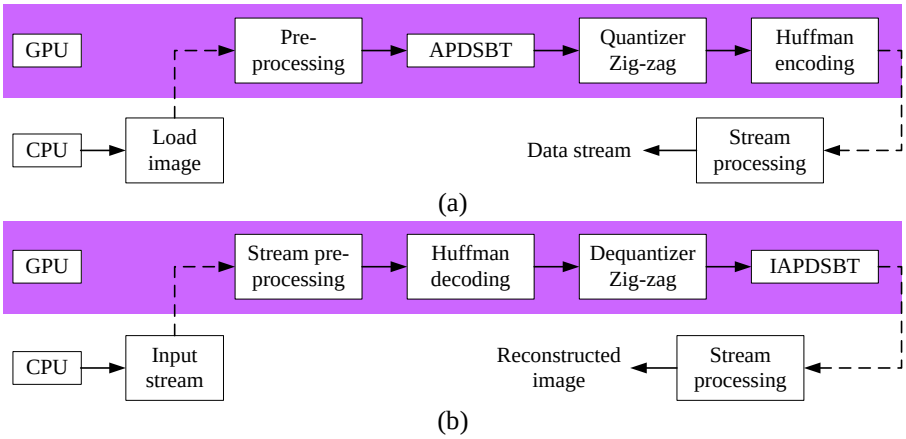


Fig. 7. Parallel model of APDSBT-JPEG: (a) Encoding and (b) Decoding

In the pre-processing of parallel JPEG compression, the data has been transformed to graphics memory. After that, the data can be processed in parallel by GPU [19]. Although parallel computation can improve the efficiency of algorithm, there is latency existing in data transmission. If we want to get higher efficiency, we should reduce the volume of data transfer between host memory and Graphics' memory, and improve the parallelism between the GPU and CPU. In the future, this phenomenon will disappear with some new technology, like zero-copy memory and APU (accelerated processing unit). And for large-scale image processing, the latency of data transmission could be hidden by massive parallel computation and the coordination processing with CPU.

There are two parallel levels in CUDA program model: the first one is parallelism between blocks in grid, and the second one is parallelism between threads in block. In JPEG compression, the image is compressed by 8×8 block, and each block has 64 pixels. The independent nature of the pixels and blocks automatically leads to data level parallelism in CUDA. Because of these two factors, JPEG compression can take advantage of CUDA's massive parallelism. With the help of GPU, the JPEG compression time can be greatly reduced to a reasonable level. Therefore, the strategy of threads mapping is obviously shown in **Fig. 8**. Different types of thread-mapping are assumed because the CPU and GPU have a different number of hardware computing units that will run work-items. In our parallel implementation, thread-mapping is the relationship between the number of image blocks and the number of CUDA block, then each CUDA block has 64 threads which corresponds to 64 pixels in image block.

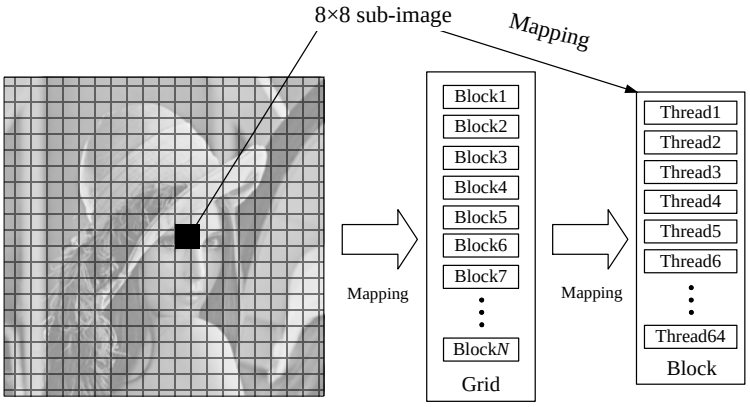


Fig. 8. Threads mapping between CUDA and image data

The forward and inverse APDSBT implementations can utilize the data parallelism per 8×8 block of samples, where each data block could be consecutively scanned by 64 concurrent threads. Thus the transform allows for effective implementations on highly parallel architectures. The quantization also could be processed in which every APDSBT coefficient from an 8×8 block is transformed according to quantization table. After forward transform, quantization, and Zig-zag scan, we will get 64 coefficients. The first coefficient, which locates in the upper left corner, is DC coefficient; the other 63 coefficients are AC coefficient which will be coded by Huffman algorithm. In parallel entropy coding, the most important part of Huffman coding is run-length coding. We can not process it like CPU, just count it. If we process it on CPU, the latency of data transmission is huge. So we use a new strategy to obtain the run-length, sorting in **Fig. 9**. AC coefficients are compared with zero parallel. If the coefficient is not equal to zero, we put the number of the thread in the array. If the coefficient is equal to zero, we put the number which is above 64 in the array in order to facilitate the sorting. After sorting, the run-length is got. In this paper, odd-even sort is used, because it is quite suitable to parallel.

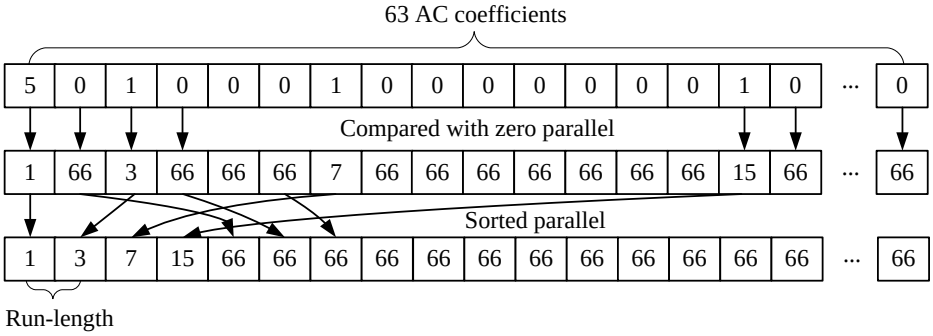


Fig. 9. Method for getting run-length

4.2 Optimal Strategy of Parallel APDSBT-JPEG

GPU is more suitable to parallel computation, because it has much more cores than CPU, and it can lunch thousands of threads at the same time. But the source of GPU is limited. In CUDA program, the threads are grouped into warps, blocks, and grid. A warp consists of 32 threads. Warps and blocks run on SM (stream multi-processor). The number of SM which can be thought of the core of CPU is an important indicator to measure the computation capacity of GPU. In GTX 480, each SM can launch up to 1536 threads. Thus, with the increase of image

data, the cost of time on GPU will increase accordingly. Applying large-scale JPEG encoding to parallel computing may result in overload problem. In parallel APDSBT-JPEG compression, we use two parallel levels to process the image data, and all the compressed operation is implemented on GPU. But the CPU is idled, when the GPU is running. So we optimize the previous parallel strategy. Most of image data is transferred to GPU, while some image data is processed on CPU, as shown in **Fig. 10**.

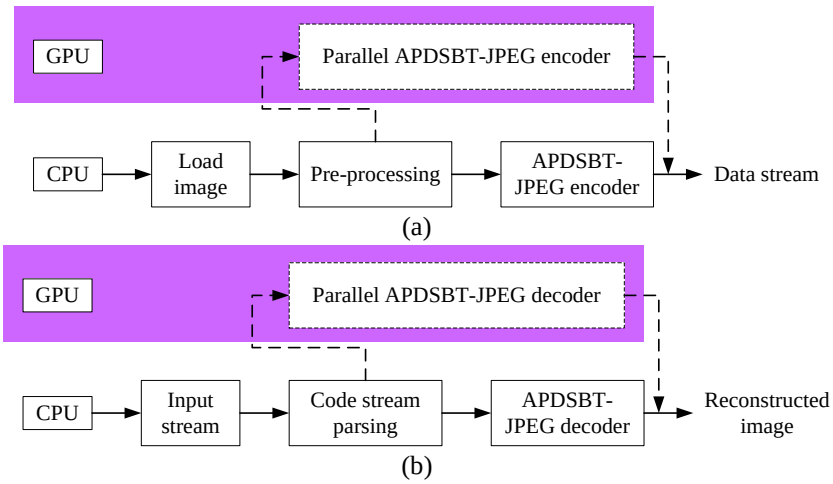


Fig. 10. The optimized parallel strategy for APDSBT-JPEG: (a) Encoding and (b) Decoding

On the GPU, not only the SM is limited, but also the register is limited. There are a number of levels of areas where the data can be placed. **Table 1** shows each defined-storage by its potential bandwidth and latency. Although register is the fastest type of storage, a kernel that requests too many registers per thread can limit the number of blocks. The GPU can schedule on an SM, and thus the total number of threads will run. So we should pay attention to balance the usage of different types of storage.

Table 1. Time cost of different memory type

Storage of Type	Registers	Shared memory	Texture memory	Constant memory	Global memory
Bandwidth	8 TB/s	1.5 TB/s	200 MB/s	200 MB/s	200 MB/s
Latency/cycle	1	1~32	400~600	400~600	400~600

5. Experimental Results and Analysis

In the experiment, we evaluate the performance of the proposed algorithm, parallel APDSBT-JPEG based on GPU, from three aspects: PSNR comparison of DCT-JPEG, DST-JPEG, APDCBT-JPEG, and APDSBT-JPEG; visual quality of the reconstructed images of DCT-JPEG, DST-JPEG, APDCBT-JPEG, and APDSBT-JPEG; and the efficiency comparison of serial algorithm and parallel algorithm. Throughout this paper, all experiments are conducted with CUDA 7.0 on the desktop computer (3.10 GHz Intel Core i3-2100 CPU, 6GB DDR3 RAM) installed with 64-bit Windows 10 operating system, the GPU used for the experiment is GTX480 with 1.5 GB graphics memory.

5.1 PSNR Comparison of DCT-JPEG, DST-JPEG, APDCBT-JPEG, and APDSBT-JPEG

In order to estimate the objective performance of propose methods, in this paper, we apply the proposed transform APDSBT to JPEG image coding. In the experiment, the novel JPEG-like image compression algorithm is tested by typical test images (Lena, Baboon, Barbara, and Bridge, all of them are 8 bit/pixel, monochrome images with size of 512×512. The original format is BMP.). Peak signal to noise ratio (PSNR) is chosen to measure the performance of reconstructed images in this experiment.

$$\mathrm{PSNR} = 10\log_{10}\left(\frac{255^2 MN}{\sum_{i=1}^{M}\sum_{j=1}^{N}\left[I_{\mathrm{in}}(i,j) - I_{\mathrm{out}}(i,j)\right]^2}\right)(\mathrm{dB}), \tag{16}$$

where I_{in} and I_{out} stand for the original image and the reconstructed image respectively. M and N represent the height and width of the test image.

Table 2 shows the experimental results of image Lena with DCT-JPEG, DST-JPEG, APDCBT-JPEG, and APDSBT-JPEG in terms of PSNR at different bit rates. The experimental results of image Baboon are shown in **Table 3**. From **Table 2** and **Table 3**, we conclude that compared with DCT-JPEG algorithm at the same bit rates in terms of PSNR, the performance of APDSBT-JPEG algorithm is superior to conventional DCT-JPEG in image compression. And the performance of proposed transform is the best among the four transforms. **Fig. 11** shows the ratio distortion curves of different test images (Lena, Barbara, Baboon, and Bridge). With the curves shown in **Fig. 11**, we get the more intuitive results.

Table 2. Experimental results of image Lena

Bit rate/bpp	PSNR/dB			
	DCT-JPEG	APDCBT-JPEG	DST-JPEG	APDSBT-JPEG
0.15	25.82	26.76	20.38	27.21
0.20	28.91	29.16	22.41	29.40
0.25	30.69	30.67	23.52	30.95
0.30	31.92	31.72	24.60	32.11
0.40	33.62	33.30	26.49	33.71
0.50	34.74	34.40	27.76	34.88
0.60	35.61	35.27	28.88	35.82
0.75	36.62	36.33	30.56	36.90
1.00	37.93	37.63	32.59	38.20
1.25	39.00	38.65	33.72	39.24

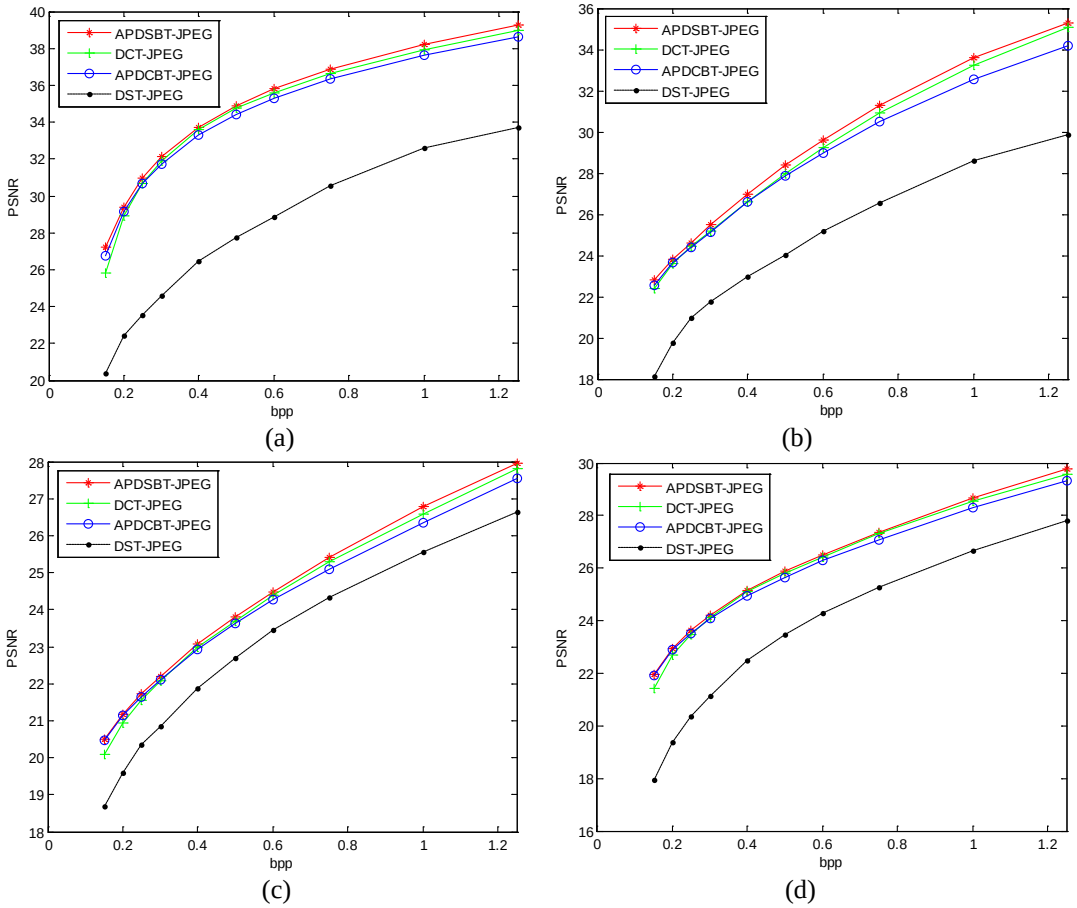


Fig. 11. The ratio distortion curves: (a) Lena, (b) Barbara, (c) Baboon, and (d) Bridge

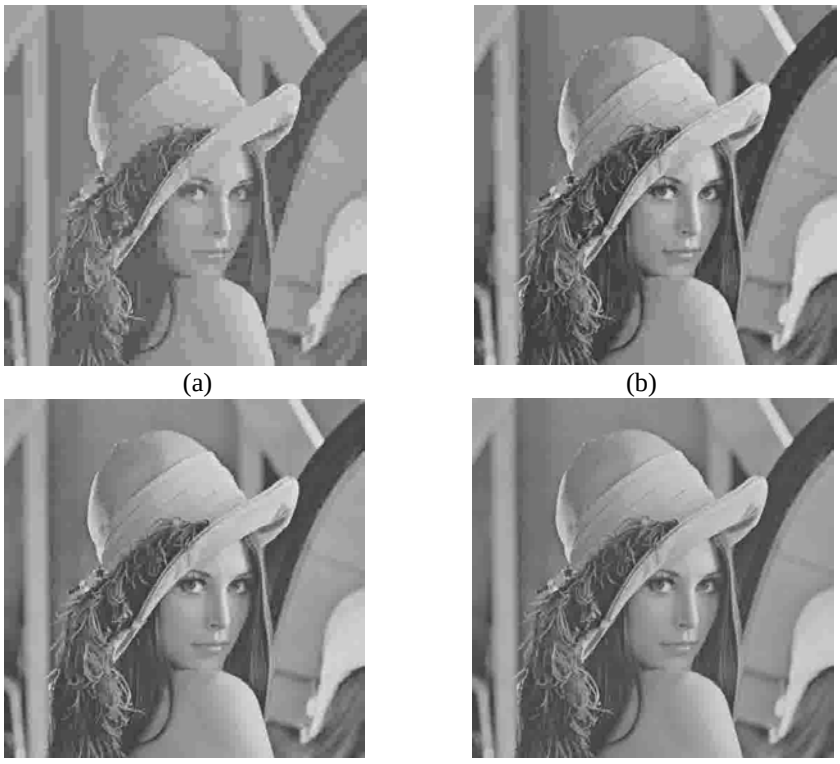
Table 3. Experimental results of image Baboon

Bit rate/bpp	PSNR/dB			
	DCT-JPEG	APDCBT-JPEG	DST-JPEG	APDSBT-JPEG
0.15	20.09	20.47	18.68	20.50
0.20	20.94	21.13	19.59	21.18
0.25	21.56	21.65	20.35	21.72
0.30	21.09	22.12	20.84	22.20
0.40	22.98	22.93	21.86	23.06
0.50	23.69	23.62	22.70	23.80
0.60	24.38	24.27	23.44	24.48
0.75	25.30	25.10	24.32	25.40
1.00	26.57	26.36	25.57	26.79
1.25	27.81	27.56	26.64	27.96

5.2 Visual Quality of the Reconstructed Images of DCT-JPEG, DST-JPEG, APDCBT-JPEG, and APDSBT-JPEG

In order to compare the compression performance subjectively, **Fig. 12** shows the reconstructed image Lena which is obtained by using DST-JPEG, DCT-JPEG,

APDCBT-JPEG, and APDSBT-JPEG at a certain bit rate 0.20 bpp. From the experiment results, it is clear that, compared with the DCT-JPEG and other like-JPEG algorithm, the visual quality of the proposed algorithm APDSBT-JPEG has a better subjective quality. And we can see that compared with conventional DCT-JPEG, the proposed algorithm improves the performance at various bit rates, both in terms of PSNR and visual quality.



(a) (b) (c) (d)

Fig. 12. Lena images obtained at 0.20 bpp: (a) DST-JPEG, with PSNR=22.41 dB, (b) DCT-JPEG, with PSNR=28.91 dB, (c) APDCBT-JPEG, with PSNR=29.16 dB, and (d) APDSBT-JPEG, with PSNR=29.40 dB

5.3 The Efficiency Comparison of Serial Algorithm and Parallel Algorithm

From the experimental results in **Fig. 13**, we know the efficiency of parallel APDSBT algorithm is much higher than that of serial APBT algorithm. In order to facilitate the comparison, the time of APDSBT-JPEG on CPU is reduced 10 times. The parallel APBT that runs on GPU can gain at least 100 times acceleration, and the maximum speedup ratio can reach more than 140 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, and the efficiency of APDSBT is greatly improved.

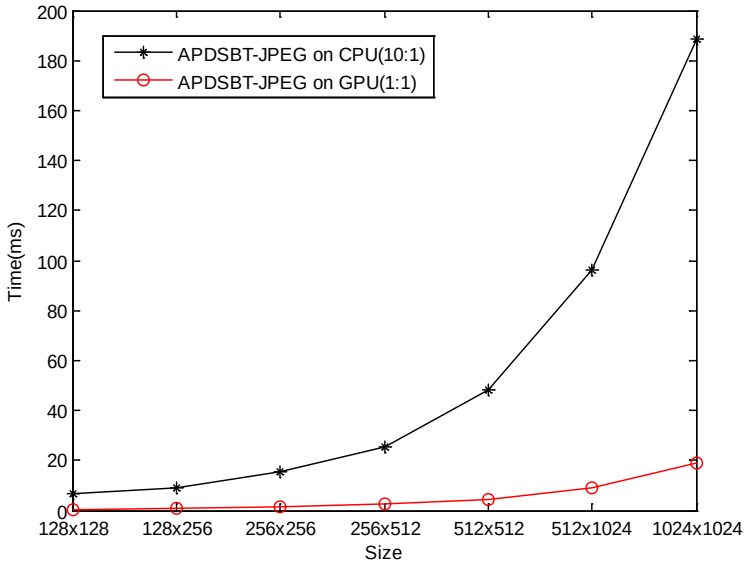


Fig. 13. The running time of APDSBT-JPEG in different platforms

In APDSBT-JPEG, the part of quatization and dequatization is different from DCT-JPEG. The uniform quantization step is adopted in APDSBT-JPEG. Since the uniform quantization step is simpler than luminance quantization table, the complexity of APDSBT-based JPEG is lower than that of DCT-based JPEG, and APDSBT-JPEG could reduce memory usage rate. From **Fig. 14**, we know that the efficiency of proposed method is higher than that of DCT-JPEG.

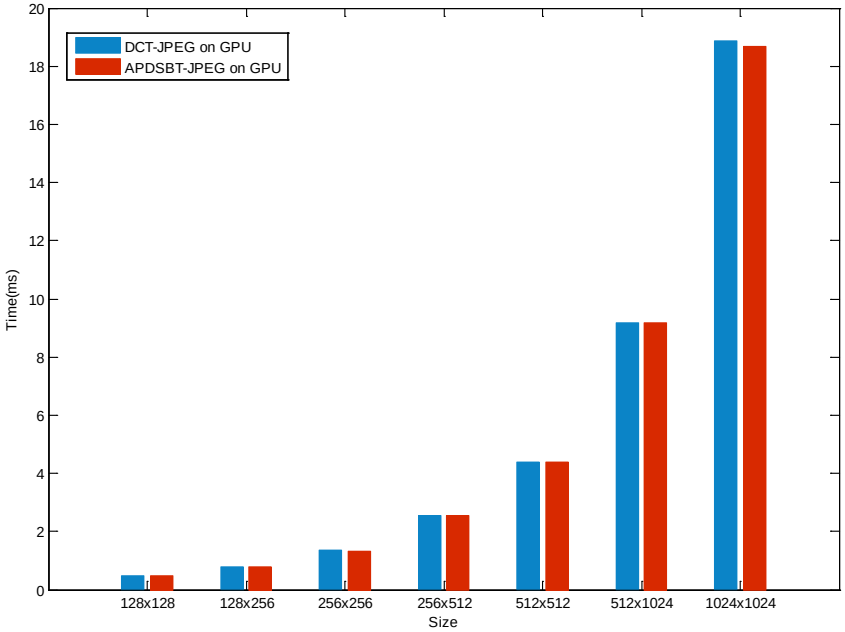


Fig. 14. Comparison of DCT-JPEG on GPU and APDSBT-JPEG on GPU

From **Fig. 14**, we can see that with the increase of image data, the cost of time on GPU will increase correspondingly. Because the source of GPU is limited, in this paper, we use the optimal strategy to parallel the proposed APDSBT-JPEG. The image is divided into two slices

with different size. The small slice will be compressed by CPU, and the big one will be transferred to graphics memory. The image data on GPU will be processed in parallel by thousands threads. The comparison between the parallel algorithm which is proposed previously (GPU, in red) and the optimal parallel strategy (CPU+GPU, in blue) is shown in **Fig. 15**. From the bar graph, we know that the efficiency can be improved over 10% by the optimal parallel strategy.

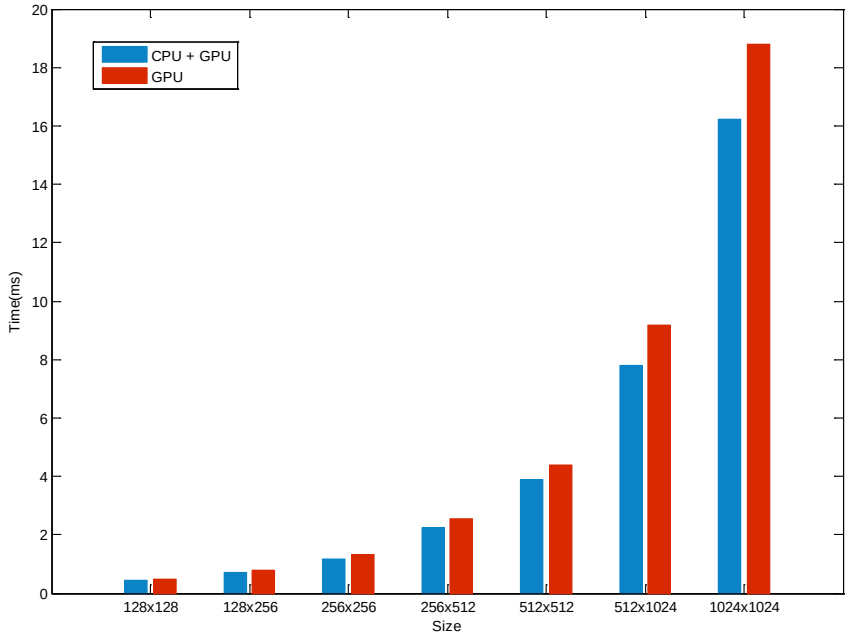


Fig. 15. Comparison between the parallel algorithm and the optimal parallel strategy

6. Conclusion

The paper proposes a new transform which is called APDSBT based on the theory of all phase digital filter and discrete sine transform. The compression and reconstruction of image is successfully achieved with APDSBT by replacing DCT which is commonly used in image compression. Compared with DCT-JPEG and APDCBT-JPEG, better objective and subjective performance is obtained. By inheriting good properties from APBT-JPEG, the advantage of the proposed algorithm is the simple quantization, taking uniform quantization for transform coefficients, especially saving many multiplication operations when adjusting the bit rates. A simpler and more effective algorithm is therefore developed, and can be easily implemented in both software and hardware. An efficient parallel implementation of APDSBT-JPEG is presented with parallel computation based on GPU. With the parallel computing algorithm, the computing efficiency is improved more than one hundred times speed ratio, compared with conventional serial algorithms based on CPU. We also propose some parallel strategies and methods to improve the efficiency of parallel algorithm, which can improve the efficiency over 10%.

In the near future, we will apply the APDSBT to video compression, like H.264 and HEVC, and use parallel algorithm to accelerate the speed of video compression.

Acknowledgments

This work was supported by the Natural Science Foundation of Shandong Province, China (Grant No. ZR2015PF004), the National Natural Science Foundation of China (Grant No. 61201371), the promotive research fund for excellent young and middle-aged scientists of Shandong Province, China (Grant No. BS2013DX022), and the Fundamental Research Funds of Shandong University (Grant No. 2014ZQXM008). The authors thank Qiming Fu, Heng Zhang, and Yunpeng Zhang for their kind help and valuable suggestions in revising this paper. The authors also thank the anonymous reviewers and the editors 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 10918-1:1994 | ITU-T Rec. T.81, Sept. 22, 2011. [Online] Available ww.iso.org/iso/catalogue_detail.htm?csnumber=18902.

[2] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103-1120, Sept. 2007. Article (CrossRef Link).

[3] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no. 12, pp. 1649-1668, Dec. 2012. Article (CrossRef Link).

[4] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Transactions on Computers*, vol. 23, no. 1, pp. 90-93, Jan. 1974. Article (CrossRef Link).

[5] Z. X. Hou and X. Yang, "The all phase DFT filter," in *Proc. of the 10th IEEE Digital Signal Processing (DSP) Workshop and the 2nd IEEE Signal Processing Education (SPE) Workshop*, Pine Mountain, Georgia, USA, Oct. 13-16, 2002, pp. 221-226. Article (CrossRef Link).

[6] Z. X. Hou, Z. H. Wang, and X. Yang, "Design and implementation of all phase DFT digital filter," *Acta Electronica Sinica*, vol. 31, no. 4, pp. 539-543, Apr. 2003. Article (CrossRef Link).

[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, pp. 791-802, Nov. 2009. Article (CrossRef Link).

[8] Q. M. Fu, X. Zhou, C. Y. Wang, and B. C. Jiang, "Windowed all phase biorthogonal transform and its application in JPEG-like image compression," *Journal of Communications*, vol. 10, no. 4, pp. 284-293, Apr. 2015. Article (CrossRef Link).

[9] Z. X. Hou and N. N. Xu, "Windowed all phase DFT digital filter," *Journal of Tianjin University (Science and Technology)*, vol. 38, no. 5, pp. 448-454, May 2005. Article (CrossRef Link).

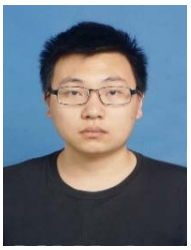
[10] A. Foi, K. Dabov, V. Katkovnik, and K. Egiazarian, "Shape-adaptive DCT for denoising and image reconstruction," in *Proc. of the SPIE - IS & T Electronic Imaging -- Image Processing: Algorithms and Systems, Neural Networks, and Machine Learning*, San Jose, USA, Jan. 16-18, vol. 6064, Article number: 60640N, 12 pages, 2006. Article (CrossRef Link).

[11] M. Budagavi, A. Fuldseth, G. Bjontegaard, V. Sze, and M. Sadafale, "Core transform design in the high efficiency video coding (HEVC) standard," *IEEE Journal of Selected Topics in Signal Processing*, vol. 7, no. 6, pp. 1029-1041, Dec. 2013. Article (CrossRef Link).

[12] Y. A. Reznik, "Relationship between DCT-II, DCT-VI, and DST-VII transforms," in *Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, Canada, May 26-31, pp. 5642-5646, 2013. Article (CrossRef Link).

[13] S. Zoican, R. Zoican, and D. Galatchi, "Methods for real time implementation of image processing algorithms," *UPB Scientific Bulletin, Series C: Electrical Engineering*, vol. 77, no. 2, pp. 127-148, Mar. 2015. Article (CrossRef Link).

[14] Z. Xue, K. K. Loo, J. Cosmas, and P. Y. Yip, "Distributed video coding in wireless multimedia sensor network for multimedia broadcasting," *WSEAS Transactions on Communications*, vol. 7, no. 5, pp. 418-427, Mar. 2008. Article (CrossRef Link).
[15] I. K. Park, N. Singhal, M. H. Lee, S. Cho, and C. Kim, "Design and performance evaluation of image processing algorithms on GPUs," *IEEE Transactions on Parallel and Distributed Systems*, vol. 23, no. 1, pp. 91-104, Jan. 2011. Article (CrossRef Link).
[16] C. G. Yan, Y. D. Zhang, J. Z. Xu, F. Dai, J. Zhang, Q. H. Dai, and F. Wu, "Efficient parallel framework for HEVC motion estimation on many-core processors," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 12, pp. 2077-2089, Dec. 2014. Article (CrossRef Link).
[17] 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, pp. 573-576, May 2014. Article (CrossRef Link).
[18] W. Xiao, B. Li, J. Z. Xu, G. M. Shi, and F. Wu, "HEVC encoding optimization using multicore CPUs and GPUs," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 25, no. 11, pp. 1830-1843, Nov. 2015. Article (CrossRef Link).
[19] S. Che, M. Boyer, J. Y. Meng, D. Tarjan, J. W. Sheaffer, and K. Skadron, "A performance study of general-purpose applications on graphics processors using CUDA," *Journal of Parallel and Distributed Computing*, vol. 68, no. 10, pp. 1370-1380, Oct. 2008. Article (CrossRef Link).
[20] P. Enfedaque, F. Auli-Llinas, and J. C. Moure, "Implementation of the DWT in a GPU through a register-based strategy," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 12, pp. 3394-3406, Dec. 2015. Article (CrossRef Link).
[21] Y. K. Wang and W. B. Huang, "A CUDA-enabled parallel algorithm for accelerating retinex," *Journal of Real-Time Image Processing*, vol. 9, no. 3, pp. 407-425, Sept. 2014. Article (CrossRef Link).
[22] H. Fassold and J. Rosner, "A real-time GPU implementation of the SIFT algorithm for large-scale video analysis tasks," in *Proc. of the SPIE - IS & T Electronic Imaging -- Real-Time Image and Video Processing*, San Francisco, USA, Feb. 10, vol. 9400, Article number: 940007, 8 pages, 2015. Article (CrossRef Link).
[23] D. Liu and X. Y. Fan, "Parallel program design for JPEG compression encoding," in *Proc. of the 9th International Conference on Fuzzy Systems and Knowledge Discovery*, Chongqing, China, May 29-31, vol. 9, pp. 2502-2506, 2012. Article (CrossRef Link).
[24] 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, pp. 1991-2006, Oct. 2013. Article (CrossRef Link).
[25] N. Alqudami and S. D. Kim, "Adaptive discrete cosine transform-based image compression method on a heterogeneous system platform using open computing language," *Journal of Electronic Image*, vol. 23, no. 6, 16 pages, Nov.-Dec. 2014. Article (CrossRef Link).
[26] G. B. Shen, G. P. Gao, S. P. Li, H. Y. Shum, and Y. Q. Zhang, "Accelerate video decoding with generic GPU, accelerate video decoding with generic GPU," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 5, pp. 685-693, May 2005. Article (CrossRef Link).
[27] C. Y. Wang, R. Y. Shan, and X. Zhou, "APBT-JPEG image coding based on GPU," *KSII Transactions on Internet and Information Systems*, vol. 9, no. 4, pp. 1457-1470, Apr. 2015. Article (CrossRef Link).
[28] X. Zhou, Q. M. Fu, F. F. Yang, and C. Y. Wang, "Implementation of biorthogonal wavelet transform using windowed APDF based on DCT," *International Journal of Signal Processing, Image Processing and Pattern Recognition*, vol. 7, no. 6, pp. 1-16, Dec. 2014. Article (CrossRef Link).



Rongyang Shan received his B.E. degree in communication engineering from Shandong University, Weihai, China, in 2014. He is currently pursuing his M.E. degree in signal and information processing at Shandong University, China. His current research interests include transform coding, parallel computing, and video watermarking.

Xiao Zhou 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. She is currently a lecturer and supervisor of postgraduate students with the School of Mechanical, Electrical and Information Engineering, Shandong University, Weihai, China. Her current research interests include wireless communication technology, and digital image processing and analysis.

Chengyou Wang received his B.E. degree in electronic information science and technology from Yantai University, China, in 2004, and his M.E. and Ph.D. degrees in signal and information processing from Tianjin University, China, in 2007 and 2010, respectively. He is currently an associate professor and supervisor of postgraduate students with the School of Mechanical, Electrical and Information Engineering, Shandong University, Weihai, China. His current research interests include digital image/video processing and analysis (transform coding, digital watermarking, image forensics, image dehazing, image quality assessment, etc.), computer vision, pattern recognition and machine learning, and multidimensional signal and information processing.

Baochen Jiang received his B.S. degree in radio electronics from Shandong University, China, in 1983 and his M.E. degree in communication and electronic systems from Tsinghua University, China, in 1990. He is currently a professor and supervisor of postgraduate students with the School of Mechanical, Electrical and Information Engineering, Shandong University, Weihai, China. His current research interests include signal and information processing, digital image/video processing and analysis, and smart grid technology.