TELKOMNIKA, Vol.13, No.3, September 2015, pp. 1089~1096
ISSN: 1693-6930, accredited A by DIKTI, Decree No: 58/DIKTI/Kep/2013
DOI: 10.12928/TELKOMNIKA.v13i3.1776

A Fast Fractal Image Compression Algorithm Combined with Graphic Processor Unit

Hui Guo*, Jie He
School of Information and Electronic Engineering, Wuzhou University, Wuzhou543002, Guangxi, China
*Corresponding author, e-mail: guohui928@qq.com

Abstract

Directed against the characteristics of computational intensity of fractal image compression encoding, a serial-parallel transfer mechanism is built for encoding procedures. By utilizing the properties of single instruction and multithreading execution of compute unified device architecture (CUDA), the parallel computational model of fractal encoding is built on the graphic processor unit (GPU) in order to parallelize the considerably time-consuming serial execution process of searching for the block of best match. The experimental result indicates, the algorithm in this paper shortens the encoding time to the millisecond scale and significantly boosts the execution efficiency of fractal image encoding algorithm while keeping the decoded image in good quality.

Keywords: *fractal image compression, graphic processor unit, compute unified device architecture, parallel computing*

1. Introduction

Fractal image encoding is a compression method within the spatial domain with high compression ratio and high decoding quality. However, the unnecessarily long encoding time has limited its popularization and application. In order to reduce the encoding time, a lot of scholars have raised the feature classification method or method of clustering to reduce the search time for matching blocks [1-4]. Jiang Zheng *et al.* [5] proposed a K-mean clustering optimization based fractal encoding. Against the problem that the K-mean clustering fractal encoding algorithm depends on data distribution, Wu Yiquan and Sun Ziyi [6] proposed an immune particle swarm optimization (PSO) and kernel fuzzy clustering based method, which can achieve an acceleration of 6 times as compared with the basic fractal encoding algorithm. Hui Guo *et al.* [7] have modified the quadtree fractal encoding method in combination with the human visual system, in order to control the distortion within the range beyond human eye recognition when decoding, but the acceleration effect is no more than 27 times. Since the fractal encoding renders the typical characteristic of serial execution, the matching procedure is to implement global or classified local search of D-block pool for each R block one by one, which can be viewed as serial repetitive execution over the same procedures, so the encoding is rather time-consuming. Parallelized execution of these procedures would be a feasible optimization method. Especially, given the availability of a lot of hardware with parallel computation structure nowadays, the encoding speed will promote significantly if the fractal encoding can be integrated with a certain parallel hardware with high popularity and low cost to establish a corresponding implementation mechanism.

The research of the above scholars is to shorten the encoding time in the dimension of optimization of encoding algorithm. The fractal encoding renders a typical characteristic of serial execution. The matching procedure is to implement global or classified local search of D-block pool for each R block one by one, which can be viewed as serial repetitive execution over the same procedures. Thus, to parallelize these procedures would be a feasible optimization method. Especially, given the availability of a lot of hardware with parallel computation structure nowadays, the encoding speed will promote significantly if the fractal encoding can be integrated with a certain parallel hardware with high popularity and low cost to establish a corresponding implementation mechanism. The image processor graphic processor unit (GPU) has large quantities of parallel hardware arithmetic units which are applicable to parallel

Received March 23, 2015; Revised July 8, 2015; Accepted July 25, 2015

computation of multiple data objects. The compute unified device architecture *compute* unified device architecture (CUDA) is a new type of software architecture and programming model for handling and managing GPU computation. With single instruction and multidata execution modes, it can utilize CPU to process the sequential portion of applications, and at the same time perform parallel execution of the compute-intensive portion on GPU via API with thread as the basic unit [8].

This paper offers a fast fractal image compression algorithm built on the base of GPU and utilizing CUDA for parallel encoding. This parallel encoding method comprises of three components: the 4-neighborhood average method adopted for space compression of the domain block, preprocessing of range and domain blocks, and computation of minimum mean square error. The process of space compression of the domain block begins with the use of a parallel execution scheme, namely each thread of GPU performs the average sampling job of one domain block. On the preprocessing stage, each thread of GPU will figure out the sum of pixels and sum of squares of pixels, respectively, for each range block and the searched domain block. In the computation of minimum mean square error, each range block will have a corresponding standalone thread for affine transformation and solution to minimum mean square error. The experimental result indicates the algorithm in this paper can speed up 120 and more times as compared with the traditional fractal compression method while keeping the decoded image in good quality.

2. Traditional Fractal Encoding Method

Mandelbrot first raised the fractal image was an iterated function system [9]. He believed that many matters in the natural world had similar parts, and pointed out a fractal cloud could be described by a simple mathematical function. In 1988, Barnsly and Sloan raised the fractal image compression method, utilizing the image's local self-similarity for compression [10]. The practical fractal blocked encoding put forward by Barnsley's doctoral student Jacquin was exactly developed on this base [11]. The fractal encoding method shall first partition an image into non-overlapping $R\times R$ blocks and possibly overlapping $D\times D$ blocks, which are called range blocks (R blocks) and domain blocks (D blocks). The size of domain blocks must be greater than that of range blocks. The following step is to perform average sampling of the domain blocks so as to accord the size of domain blocks with that of range blocks. All the domain blocks can be saved in the domain pool S_D.

A $N\times N$ sized image can be partitioned into i domain blocks and j range blocks, where $i=0,1,2,\ldots(N-2R+1)^2$, $j=0,1,2,\ldots,(N/R)^2$. Search for the domain block of best match from the domain pool S_D by the norm of minimum square error. The affine transformation formula is shown as Formula 1.

$$\begin{bmatrix} x \\ y \\ R_{xy} \end{bmatrix} = \begin{bmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & s_i \end{bmatrix} \begin{bmatrix} x \\ y \\ D_{xy} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ o_i \end{bmatrix} \quad (1)$$

Where $x = 0, 1, 2, \ldots, R-1$, $y = 0, 1, 2, \ldots, R-1$, R_{xy} and D_{xy} are the values of pixels within range block R_j and domain block D_i. Parameters a, b, c and d are used for 8 isometric transformations of pixel: 4 rotations and 4 reversals, as shown in Figure 1. S_i and O_i are the contrast and brightness adjustment coefficients, respectively, in the process where domain block D_i is matched with range block R_j. The computational formulas for S_i and O_i are shown as Formula 2 and Formula 3, respectively.

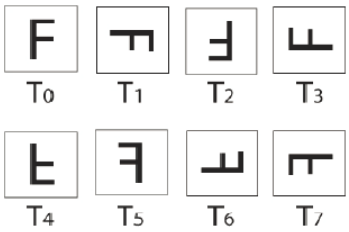

Figure 1. 8 Isometric Transformations

$$S_i = \frac{R^2\left[\sum_x^R\sum_y^R D'_{xy}\, R_{xy}\right] - \left[\sum_x^R\sum_y^R D'_{xy}\right]\left[\sum_x^R\sum_y^R R_{xy}\right]}{R^2\left[\sum_x^R\sum_y^R (D'_{xy})^2\right] - \left[\sum_x^R\sum_y^R D'_{xy}\right]} \tag{2}$$

$$O_i = \frac{\sum_x^R\sum_y^R R_{xy} - S_{ij}\sum_x^R\sum_y^R D'_{xy}}{R^2} \tag{3}$$

D'_{xy} is the pixel value in correspondence to the domain blocks via the 8 isometric transformations. The root-mean-square, RMS_{i-j}, in the process where domain block D_i is matched with range block R_j can be figured out as Formula 4. The minimum root-mean-square $RMS_{\min}$ is shown as Formula 5.

$$RMS_{i-j} = \left(\frac{1}{R^2}\sum_x^R\sum_y^R \left(s_j D_{xy} + o_j - R_{xy}\right)^2\right)^{1/2} \tag{4}$$

$$RMS_{\min} = \min(RMS \quad, i = 0,1,2,...(N-2R+1)^2, \; j = 0,1,2,..., (N/R)^2) \tag{5}$$

Suppose *I* is the original image, *R* is the size of range block, *D* is the size of domain block. The concrete algorithmic steps are shown as the following:

Step 1: Partition the original image *I* into non-overlapping range blocks R_j with the size of *R*×*R.*

Step 2: Partition the original image *I* into possibly overlapping domain blocks D_i with the size of *D*×*D*.

Step 3: Perform average sampling of the domain blocks so as to accord their size with that of the range blocks.

Step 4: For each range block R_j, find the corresponding domain block D_i from the domain pool S_D. Make sure if the difference of mean square errors between R_j and D_i is the minimum after the affine transformation over D_i, then this D_i is the block of best match for R_j.

Step 5: For each range block R_j, record the fractal code (IFS code) constituted by transformation library $w_i(i, n, s_i, o_i)$:

(1) The number *i* of block D_i of best match;

(2) Turn R_j and D_i into a number *n* (*n* ranges from 0 to 7) of isometric transformations;

(3) Contrast adjustment coefficient S_i and brightness adjustment coefficient O_i.

3. GPU Combined Parallel Fractal Encoding Algorithm

CUDA is a new hardware and software architecture for handling and managing GPU computation. Applications can handle the section of sequential execution via CPU and perform parallel execution of the compute-intensive section on GPU via relevant API to CUDA, thereby give more full play to the large-scale concurrent computing power of display card. While the program is running, the concurrent processing section in CUDA program is performed by the kernel function. The basic unit of the kernel function running on GPU is thread. CUDA may produce a lot of concurrent threads at different addresses. These threads execute the kernel function and implement parallel processing of data. Figure 2 demonstrates the basic execution principle of CUDA. The program code developed by CUDA programming comprises of two sections: Host code and the device code. The host code is the serial processing running on CPU, whereas the device code is the parallel processing running on the display chip GPU. The host code takes the typical and principal charge of scheduling the overall and strongly logical serial operations, such as initialization of GPU and data exchange, etc., while the device code is mainly responsible for parallel data processing with high degree of parallelization in the program.

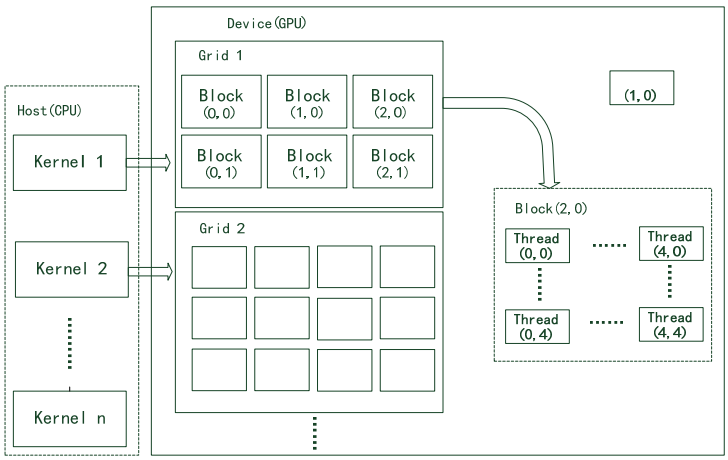

Figure 2. Execution Principle of CUDA

The traditional fractal image compression method is considerably time-consuming because of the great amount of calculation in the process of searching the block of match for the range blocks. In order to speed up the searching, this paper raises a GPU based fast fractal image compression algorithm using CUDA for parallel encoding. In the following such parallel execution mechanism is demonstrated via Figure 3, in which T_1, T_2, ... are threads on GPU, D_{i_sum} and $D_{i\_powersum}$ are the sum of pixels and sum of squares of pixels of domain blocks D_i, R_{j_sum} and $R_{j\_powersum}$ are the sum of pixels and sum of squares of pixels of range blocks R_j. The fractal image compression method of such parallel processing falls into three steps: Average sampling of domain blocks, preprocessing of range blocks and domain blocks, and computation of minimum mean square error.

The Average-Sample in Figure 3 is average sampling. Prior to searching for the matched, a 4-neighborhood pixel means processing shall be performed over the *D* blocks within the *D* block pool. The computation work of this part is equally distributed over all threads. The processed data are saved into the texture memory. The concrete process for 4-neighborhood average is shown as Figure 4.

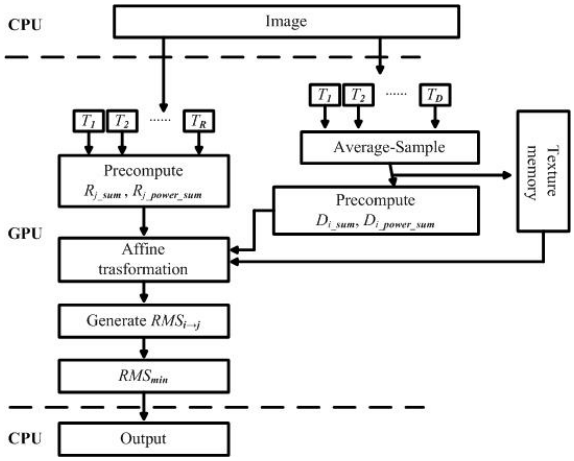

Figure 3. Parallel Processing on GPU

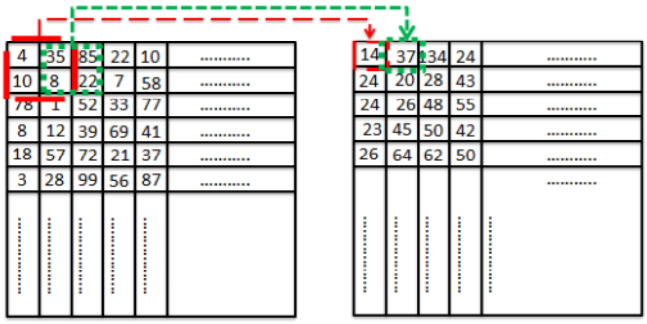

Figure 4. 4-Neighborhood Average Method

Precompute is the preprocessing. After conducting sub-sampling over the D blocks, each thread on GPU is allocated to precompute the R_{j_sum} value and $R_{j\_powersum}$ value of each range block, as well as the D_{i_sum} value and $D_{i\_powersum}$ value of each domain block, with the respective computational formula shown as follows:

$$R_{j\_sum} = \sum_{x}^{R}\sum_{y}^{R} R_{xy} \tag{6}$$

$$R_{j\_powersum} = \sum_{x}^{R}\sum_{y}^{R} \left(R_{xy}\right)^2 \tag{7}$$

$$D_{i\_sum} = \sum_{x}^{R}\sum_{y}^{R} D_{xy} \tag{8}$$

$$D_{i\_powersum} = \sum_{x}^{R}\sum_{y}^{R} \left(D_{xy}\right)^2 \tag{9}$$

Computation of the minimum root-mean-square error. Each *R* block shall be equally distributed to each thread for search and match of *D* blocks in the *D* block pool, for computation of affine transformation, and for generation of *RMS*. Finally, RMS_{min} is found out by comparing

the generated *RMS*. The *D* block in correspondence to this RMS_{min} is just the searched block of match. Record its corresponding IFS code $w_i(i, n, s_i, o_i)$ and perform quantization coding to get the fractal code of each range block. The concrete matching process is shown in Fig 5, where T_0, T_1, T_2, ... are threads on GPU.

The following is the concrete steps of fractal image encoding through which the CUDA structure is utilized for parallel encoding:

Input: For an *N*×*N* sized grayscale image, the grayscale of pixels is quantized by 8 bits, *N* is generally a power of *2*.

Output: Fractal code, namely $w_i(i, n, s_i, o_i)$.

Step 1: Read in the image data at the CPU side. Partition the original image *I* into non-overlapping range blocks R_j with the size of *R*×*R*; partition the original image *I* into possibly overlapping domain blocks D_i with the size of *D*×*D*. And transmit these data into the device's memory.

Step 2: Perform average sampling to constitute a codebook. Allocate the average sampling work of each domain block to each thread, namely assign one thread to perform the sub-sampling work of one domain block.

Step 3: Processing of the domain blocks and range blocks, namely compute R_{j_sum}, $R_{j\_powersum}$, D_{i_sum} and $D_{i\_powersum}$. Allocate the preprocessing of each domain block and range block to each thread to perform the computation work.

Step 4: Computation of mean square error *RMS*. In the kernel function, distribute the range blocks equally into each thread for the match with all the domain blocks in the domain pool, and for affine transformation and *RMS* computation. Finally, find out RMS_{min} and record the fractal code $w_i(i, n, s_i, o_i)$ in correspondence to this RMS_{min}.

Step 5: Transmit the fractal code from the device end to the CPU end.

Step 6: Output the fractal code $w_i(i, n, s_i, o_i)$.

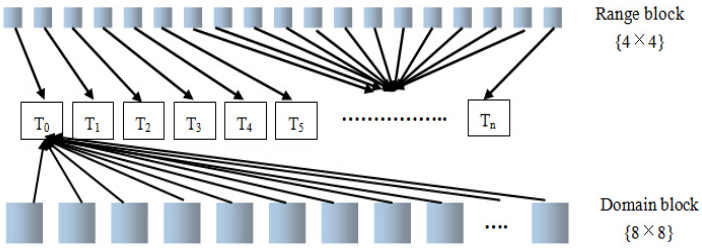

Figure 5. Matching Process between Range Blocks and Domain Blocks

4. Experimental Result Analysis

To verify the validity of the algorithm, this paper adopts four standard test images - 256×256×8 Lena, Pepper, Cat and Cell - for a test. These 4 images are of representative significance in terms of equilibrium and change of texture and marginal details, and are well capable of testing various image processing algorithms. For all the images to be tested, the sizes of range blocks are set as 4×4, the sizes of domain blocks as 8×8. The development environment of program is Microsoft Visual Studio 2010+ CUDA6.0+Opencv2.3, the system is 64-bit Windows 7, the memory is 4GB, the display card is Nvidia Geforce GT 630M, and the CPU is Core I3. In the following the traditional fractal encoding algorithm and the GPU combined parallel fractal encoding algorithm are adopted separately to encode these four images. While decoding, a blank matrix is created. Via 9 iterations, the original images can be approximated. The experimental result is shown as follows:

(a) Decoded Images by Traditional Algorithm

(b) Decoded Images by Parallel Algorithm

Figure 6. Effects of Decoded Images by Two Algorithms

Figure 6 shows, from naked eyes, the decoded images by the traditional fractal image encoding algorithm and by the GPU combined parallel fractal image encoding algorithm have achieved the coincident effects, which demonstrates the feasibility of the algorithm in this paper. The following Table 1 provides the time (unit: s) of encoding, the PSNR value (unit: dB) of decoded images and the speed-up ratio of both algorithms.

Table 1. Experimental Data by Using Both Algorithms

Images	Traditional Algorithm		Parallel Algorithm		Speed-up Ratio
	T	PSNR	T	PSNR	
Lena	29.13	31.46	0.243	31.46	120
Pepper	29.11	32.00	0.237	32.00	123
Cat	29.00	36.98	0.252	36.96	115
Cell	29.06	34.92	0.245	34.94	119

From the data in Table 1, the encoding time by using the GPU combined parallel fractal encoding method is significantly shorter than that by the traditional fractal encoding algorithm; the maximum speed-up ratio can achieve 123 times, and the decoded images can be retained in good quality. Therefore, it is feasible to utilize GPU for parallelization execution of the encoding process of fractal image compression in combination with CUDA, which attaches vital significance to popularization and application of fractal image compression encoding.

5. Conclusion

This paper has raised a fast fractal image compression algorithm utilizing CUDA on GPU. Such parallel fractal image compression method falls into three steps: Average sampling of domain blocks, preprocessing of range blocks and domain blocks, computation of minimum mean square error. The experiment evinces the algorithm in this paper can achieve an acceleration of 123 times as compared with the traditional fractal image compression method, and keep the decoded images in good quality. Further development is anticipated in the future

work to use GPU to optimize CPU codes so as to make fractal image compression real-time. Furthermore, these methods will be used in other fields like dynamic image encoding, etc.

Acknowledgements

This work was supported by Guangxi Natural Science Foundation Program (Grant No. 2013GXNSFBA019275, Grant No. 2013GXNSFBA019276) and Guangxi University of Science and Technology Research Program (Grant No.2013YB227, Grant No.2013YB228).

References

[1] Bo Wang, Yubin Gao. An Image Compression Scheme Based on Fuzzy Neural Network. *TELKOMNIKA Telecommunication Computing Electronics and Control*. 2015; 13(1): 137-145
[2] Mohsen Nasri, Abdelhamid Helali, Halim Sghaier, Hassen Maaref. Efficient JPEG2000 Image Compression Scheme for Multihop Wireless Networks. *TELKOMNIKA Telecommunication Computing Electronics and Control*. 2011; 9(2): 311-318.
[3] Cui Xin-Xia, Luo Chen-Xu. Fractal and Chaos Characteristics in Rock Milled Process. *TELKOMNIKA Indonesian Journal of Electrical Engineering.* 2014; 12(1): 530-538.
[4] Yi Li, Hongchan Zheng, Guohua Peng, Min Zhou. Normal Vector Based Subdivision Scheme to Generate Fractal Curves. *TELKOMNIKA Indonesian Journal of Electrical Engineering*. 2013; 11(8): 4273-4281.
[5] Jiang Zheng, Jiang Mingyan. A Fast Fractal Image Compression Algorithm Based on K-mean Clustering Optimization. *Journal of Electrical& Electronic Education.* 2006; 36(03): 22-25.
[6] Wu Yiquan, Sun Ziyi. Fast Fractal Image Coding Based Immunity Partice Swarm Optimization and Fuzzy Kernel Clustering. *Journal of Beijing University of Posts and Telecommunications.* 2011; 34(01): 69-74.
[7] Hui Guo, Yunping Zheng, Jie He. A New HVS-Based Fractal Image Compression Algorithm. *Lecture Notes in Electrical Engineering*. 2012; 138(2): 753-759.
[8] Ma Wei-wei, SUN Dong, WU Xian-liang. Research on high-order SFDTD parallel computing based on GPU. *Journal of Hefei University of Technology(Natural Scicence).* 2012; 35(7): 926-929.
[9] BB Mandelbrot. The Fractal Geometry of Nature. Second edition. WH Freedman, New York. 1982.
[10] M. Barnsley and A. Sloan, A better way to compress images, BYTE, 1988, no.1,215-223
[11] AE Jacquin. Image coding based on a fractal theory of iterated contractive image transformations. *IEEE Trans. Image Processing*. 1992; 1(1): 18-30.