Journal of the Korean Astronomical Society
51: 65 ~ 71, 2018 June

https://doi.org/10.5303/JKAS.2018.51.3.65
pISSN: 1225-4614 · eISSN: 2288-890X
http://jkas.kas.org

GPU-Accelerated Speckle Masking Reconstruction Algorithm for High-Resolution Solar Images

Yanfang Zheng, Xuebao Li, Huifeng Tian, Qiliang Zhang, Chong Su, Lingyi Shi, and Ta Zhou
College of Electrical and Information Engineering, Jiangsu University of Science and Technology, Zhangjiagang 215600, China; zyf062856@163.com

Received April 5, 2018; accepted June 11, 2018

Abstract: The near real-time speckle masking reconstruction technique has been developed to accelerate the processing of solar images to achieve high resolutions for ground-based solar telescopes. However, the reconstruction of solar subimages in such a speckle reconstruction is very time-consuming. We design and implement a new parallel speckle masking reconstruction algorithm based on the Compute Unified Device Architecture (CUDA) on General Purpose Graphics Processing Units (GPGPU). Tests are performed to validate the correctness of our program on NVIDIA GPGPU. Details of several parallel reconstruction steps are presented, and the parallel implementation between various modules shows a significant speed increase compared to the previous serial implementations. In addition, we present a comparison of runtimes across serial programs, the OpenMP-based method, and the new parallel method. The new parallel method shows a clear advantage for large scale data processing, and a speedup of around 9 to 10 is achieved in reconstructing one solar subimage of 256×256 pixels. The speedup performance of the new parallel method exceeds that of OpenMP-based method overall. We conclude that the new parallel method would be of value, and contribute to real-time reconstruction of an entire solar image.

Key words: Sun: general — instrumentation: high angular resolution — methods: numerical

1. Introduction

In solar ground-based high resolution imaging, speckle image reconstruction is used to yield diffraction-limited resolutions for partially corrected images with an adaptive optics system, such as speckle masking (Lohmann et al. 1983; von der Lühe 1993), phase diversity methods (Gonsalves 1982; Paxman et al. 1992), and shift-and-add (SAA) (Li et al. 2014). Due to its high accuracy of image reconstruction, the speckle masking algorithm is commonly used in post-processing reconstruction. Because of the large amount of data and calculations involved in this algorithm, the long reconstruction time has limited universal application of the algorithm. Near real-time image reconstruction performance was obtained for the New Vacuum Solar Telescope (NVST) on a high performance cluster using the Message Passing Interface (MPI). The program we developed requires 48 seconds to reconstruct one 2368×1920 pixels image from one burst with at least 100 original images on the multi-core CPU cluster (Li et al. 2015). However, much time (one-third of the computation time) was used to reconstruct subimages from the many burst subimages in such a speckle reconstruction. Real-time image reconstruction can not only shorten the time between the observations and data analysis, but also reduce the data volume at least by a factor of 100. Thus, it is essential to parallelize and accelerate subimage reconstruction to further enhance the speedup performance.

Some aspects of using speckle masking algorithms and parallel computing techniques to reconstruct solar observation images have been already studied by Li et al. (2015), Wöger & von der Lühe (2008), and Li & Zheng (2016). They used traditional MPI and Open Multiple Processing (OpenMP) to accelerate solar images reconstruction in near real-time based on multi-core CPU. General Purpose Graphics Processing Units (GPGPU) is a novel parallel computing technique and has been widely applied to real-time astronomical data processing. The Daniel K. Inouye Solar Telescope (DKIST) is about to apply GPUs to accelerate data processing of visible broadband imager (Beard et al. 2014). NVST applied GPUs to accelerate frames-selection method of the Level1 data processing (Shi et al. 2015). However, only Wöger & Ferayorni (2012) have attempted to accelerate speckle masking in one solar subimage reconstruction on GPUs. However, they considered harnessing parallelization within the average bispectrum calculation algorithm itself, and no significant speedups were gained. By analyzing the speckle masking algorithm in subimage reconstruction, we find that the average bispectrum calculation is most time-consuming.

In this paper, we propose a new parallel method for speckle masking algorithm for solar subimage reconstruction using GPGPU. Our method differs from Wöger & Ferayorni (2012) in that the method accelerates the average bispectrum computation algorithm. Furthermore, we compare the GPU-based solutions to the OpenMP multi-core CPU execution described in Li & Zheng (2016), the computation speed of solar subimage reconstruction is increased significantly. The rest

Corresponding author: X. Li

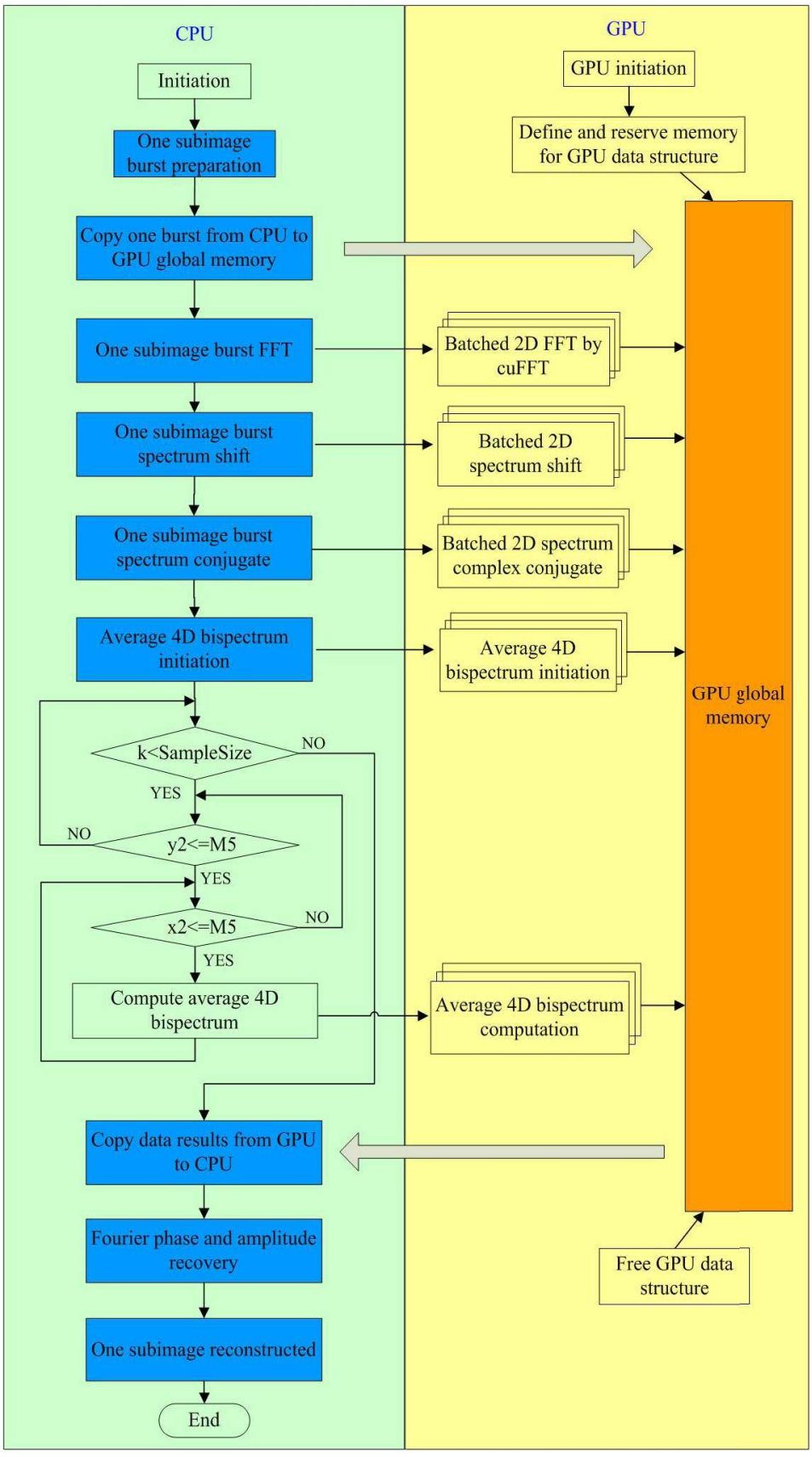

Figure 1. Flow chart of the new parallel speckle masking algorithm based on CUDA on GPGPU.

```
1：cuDoubleComplex *subimgsfft_dev, *subimgsfft_conj_dev, ****bp1, ****bp2;
2：double *subimgs_dev;
3：int SampleSize=100, row=col=256, Lb=row/2, Total=row*col, M5=5;
4：// subimgsfft_dev is batched 2D spectrum of subimgs_dev on global memery
5：// subimgsfft_conj_dev is the conjugate of subimgsfft_dev on global memery
6：// bp1,bp2 is the average 4D bispectrum on global memery
7：Batched 2D subimage preprocessing on a single core CPU
8：Transfer the preprocessed subimage data from CPU host memeory to GPU device memory
9：…
10：//Batched 2D subimage FFT
11：CUDA_FFT_FORWARD_Batch(SampleSize, subimgs_dev, subimgsfft_dev);
12：…
13：//Average 4D bispectrum computation，Eq.(1)
14：for(int k=0;k<SampleSize;k++)
15：  for(int y2=0;y2<=M5;y2++)
16：   for(int x2=0;x2<=M5;x2++)
17：   {
18：      kk=Lb-y2+1;
19：      bispectrum_kernel<<<Lb-x2+1,Lb-y2+1>>>(bp1,bp2,subimgsfft_dev,subimgsfft_conj_dev,k,kk,y2,x2,
20：      SampleSize);
21：   }
22：…
23：Transfer the average 4D bispectrum data from GPU device memory to CPU host memeory
24：Fourier phase and modulus recovery on a single core CPU
25：One subimage is reconstructed on a single core CPU

26：function CUDA_FFT_FORWARD_Batch( int SampleSize, double *indata_dev, cuDoubleComplex *outdata_dev)
27：{
28：  cufftHandle plan;
29：  cuDoubleComplex *indata_c_dev;
30：  cudaMalloc((void **)&indata_c_dev, SampleSize*Total*sizeof(cuDoubleComplex));
31：  double2complex_kernel<<<SampleSize*row*col/1024,1024>>>(indata_dev,indata_c_dev);
32：  cufftPlanMany(&plan,2,n,NULL,1,0,NULL,1,0,CUFFT_Z2Z,SampleSize);
33：  cufftExecZ2Z(plan,indata_c_dev,outdata_dev,CUFFT_FORWARD);
34：  cufftDestroy(plan);
35：  cudaFree(indata_c_dev);
36：}
37：__global__ void double2complex_kernel(double * data_dev, cuDoubleComplex *data_c_dev)
38：{
39：  int index;
40：  index=blockIdx.x*blockDim.x+threadIdx.x;
41：  data_c_dev[index].x=data_dev[index];
42：  data_c_dev[index].y=0;
43：}
44：__global__ void bispectrum_kernel(cuDoubleComplex ****bp1,cuDoubleComplex ****bp2,cuDoubleComplex
45：*datafft,cuDoubleComplex* datafft_conj, int k,int kk,int y2,int x2,int SampleSize)
46：{
47：   int index_x3,index_y3;
48：   cuDoubleComplex temp;
49：   index_y3=(blockIdx.x*blockDim.x+threadIdx.x)%kk;
50：   index_x3=(blockIdx.x*blockDim.x+threadIdx.x)/kk;
51：   temp=datafft[k][index_x3+128][index_y3+128]*datafft_conj[k][index_x3+128+x2][index_y3+128+y2];
52：   bp1[y2][x2][index_y3][index_y2]+=temp*datafft[k][128+x2][128+y2]/SampleSize;
53：   temp=datafft[k][index_x3+128][index_y3+1+y2]*datafft_conj[k][index_x3+128+x2][index_y3+1];
54：   bp2[y2][x2][index_y3][index_y2]+=temp*datafft[k][128+x2][128-y2]/SampleSize;
55：}
```

Figure 2. The significant part of the pseudo code for subimage reconstruction using speckle masking based on CUDA.

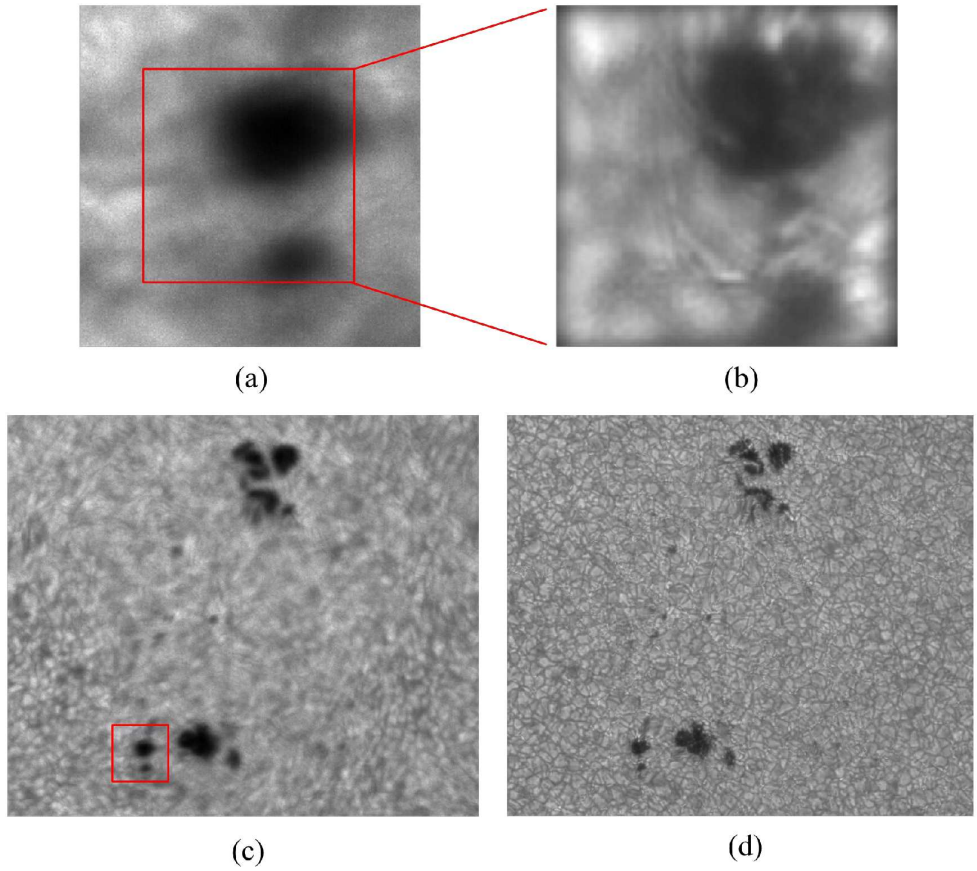

Figure 3. (a) One subimage of one subimage burst of 100×256^2 pixels before reconstruction. (b) The parallel reconstruction subimage using speckle masking based on CUDA on GPGPU. (c) and (d) are the fields from which (a) and (b) are taken.

of the paper is organized as follows. We describe the new parallel method for speckle masking algorithm in Section 2. We present results and analysis in Section 3. Finally, we present our conclusions and future work in Section 4.

2. CUDA METHOD FOR THE SPECKLE MASKING ALGORITHM

In general, one burst with at least 100 short exposure images is converted into a high resolution image. One burst with at least 100 preprocessed images is segmented into a mosaic of partially overlapping subimage bursts with 100×256^2 pixels (Li et al. 2015). The processing of each subimage burst is relatively independent of other subimage regions. Each subimage burst provides the modulus and phases of the object's Fourier transform, using the method of Labeyrie (1970) and speckle masking respectively (Lohmann et al. 1983). All the reconstructed subimages are merged to form a full high resolution solar image (Li et al. 2015). The speckle masking algorithm uses the bispectrum of the Fourier transforms which is defined by

$$\begin{aligned} B(u,v) &= \langle I_i(u) I_i(v) I_i^*(u+v) \rangle_i \\ &= O_i(u) O_i(v) O_i^*(u+v) \\ &\quad \cdot \langle H_i(u) H_i(v) H_i^*(u+v) \rangle_i, \end{aligned} \tag{1}$$

where u and v are the two-dimensional spatial frequencies. $I_i(u)I_i(v)I_i^*(u+v)$ represents the ith speckle masking bispectrum; $\langle H_i(u)H_i(v)H_i^*(u+v)\rangle_i$ denotes the average speckle masking transfer function. After obtaining the average bispectrum, the object's phase can be recovered from it. A detailed description of the technical parts of the phase reconstruction method is given in Pehlemann & von der Lühe (1989).

The Compute Unified Device Architecture (CUDA) is a general purpose parallel computing architecture with a new parallel programming model and instruction set architecture developed by the NVIDIA Corporation (CUDA Toolkit 2018). CUDA leverages the parallel engine in GPUs to solve many complex computational problems in a more efficient way than on a single CPU. It allows developers to use C or C++ as programming languages and to

Table 1
Specifications of the host PC

Host hardware specification	
Processor	Intel Xeon E5-2620
Clock speed	2.00 GHz
Memory	32 GB DDR3
Operating system	Red Hat 6.2

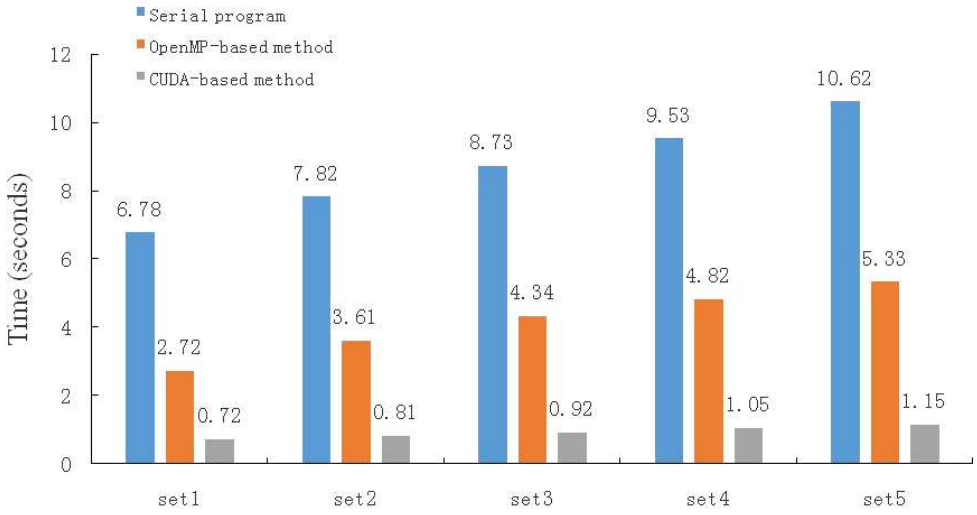

Figure 4. Subimage reconstruction time for a single core CPU serial program, the OpenMP-based method, and the CUDA-based method for different data sets: set1=100×256^2 pixels, set2=125×256^2 pixels, set3=150×256^2 pixels, set4=175×256^2 pixels, set5=200×256^2 pixels.

create high-level CPU+GPU applications without the need to explicitly manage the parallel architecture configuration and its communication with the CPU host (Flores et al. 2014). In our work, we propose a novel parallel method for speckle masking algorithm for solar subimage reconstruction based on CUDA. In the CUDA program model, CPU and GPU work together and execute their respective tasks. The GPU accelerates the computationally intense and time-consuming code, and the CPU takes advantage of its optimized ability to execute serialization code. The code to be executed on the GPU is grouped into data parallel functions called kernels. By analyzing the speckle masking algorithm on the CPU, we find that the most time-consuming process is the average four-dimensional (4D) bispectrum computation. Therefore, in our algorithm, main functions such as the batched two-dimensional (2D) Fast Fourier Transform (FFT), batched 2D spectrum shift, batched 2D spectrum conjugate, average 4D bispectrum initiation and computation are implemented in C language for the GPU. Other processes that do not take much time, such as Fourier phase and amplitude recovery, are implemented on the CPU, and the work flow is controlled by CPU. The flow chart of the new parallel method for speckle masking algorithm is shown in Figure 1.

Table 2
Specifications of GPU device

GPU specification	
Model	NVIDIA Tesla C2050
Number of processor cores	448
Processor core clock	1.15 GHz
Device memory	3 GB
Memory clock	1.50 GHz
CUDA compute capability	7.0

One subimage burst was made up of a batch of 100 2D 256×256 pixels subimages. The implementation of the CUDA-based parallel method for one subimage reconstruction covers the following steps, and the significant part of the pseudo code is shown in Figure 2. Firstly, one subimage burst is preprocessed on the CPU, and then the data is transferred to the global memory on the GPU from the CPU host memory. Secondly, the batched 2D FFT operation is executed using the cuFFT library distributed with CUDA and multi-threads kernel on GPU (cuFFT Library 2018), and then a batch of 100 2D spectra is acquired. Thirdly, three different kernels are implemented to execute parallel computing tasks with a large of number of threads on the GPU. These kernels correspond to 2D spectrum shift and conjugation, and initiation of the averaging of the 4D bispectra, respectively, for the batch of 100 images. Fourthly, the computation of the average 4D bispectrum in serial mode requires five loops, which is most time-consuming. Therefore, for the module in parallel mode, the major part of computing is reduced to three loops by unrolling two inner loops and calling the kernel `bispectrum_kernel` multiple times on CPU+GPU. Fifthly, the data of the average 4D bispectrum is transferred to CPU host memory from GPU device memory, and then is used to reconstruct the object's phase on the CPU. Finally, the object's amplitude is recovered using the method of Labeyrie (1970), and then one final subimage is reconstructed through inverse Fourier transformation of the phase and amplitude on the CPU.

The new parallel method avoids high data transfer rates between host memory and device memory. In the subimage parallel reconstruction, only one copy of subimage burst and average 4D bispectrum needs to be transferred between host memory and GPU device memory, and the data volume is so small that the time of data transfer can be neglected. The time cost of sev-

Table 3
Runtime and speedup performance measurements for one subimage burst of 100×256^2 pixels between various modules.

Module	Serial program (s)	OpenMP (s)	CUDA (s)	Speedup	
				OpenMP	CUDA
Batched 2D FFT	0.6	0.6	0.02	1	30
Batched 2D spectrum shift	0.35	0.04	0.1	8.75	3.5
Batched 2D spectrum conjugate	0.1	–	0.02	–	5
Average 4D bispectrum calculation	5.1	1.1	0.3	4.63	17
Subimage reconstruction	6.78	2.72	0.72	2.49	9.42

eral individual parallel reconstruction steps is presented in the next section.

3. Results and Analysis

For experimental purposes, we used actual solar image data acquired from the NVST. One solar subimage was reconstructed from a batch of 100 256×256 pixels subimages in the photosphere TiO channel. Figure 3(a) shows one observational original subimage of one subimage burst before reconstruction. Figure 3(b) shows the parallel reconstruction subimage using speckle masking based on CUDA on GPGPU, and its field of view is slightly smaller than that of original subimage. The spatial resolution of the reconstructed subimage is significantly better than that of original subimage. Similar images constructed using other methods are presented elsewhere (Li & Zheng 2016). There is no difference among the quality of the reconstruction, because the different methods differ mostly in the speed. All experiments were performed on a Linux operating system on a computer equipped with an Intel Xeon E5-2620 CPU at 2.00 GHz (total 16 CPU cores), 32 GB host memory, and an NVIDIA Tesla C2050 GPU with 448 cores and 3 GB device memory. The hardware specification is illustrated in Table 1 and Table 2.

In order to demonstrate the performance advantages of our proposed method, we present a runtime and speedup performance comparison based on reconstructing one subimage from one subimage burst of 100×256^2 pixels with various modules with a CPU (single core) serial program, an OpenMP-based (16 CPU cores) method, and the CUDA-based method, in Table 3. The CPU parallel processing algorithm is implemented using OpenMP. From Table 3, it is seen that the runtime of the CUDA-based method is much smaller than that of the single core CPU serial program and the OpenMP-based method. The runtime of reconstruction of one entire subimage is reduced to around 0.7 seconds based on CUDA, outperforming the OpenMP-based methodnby a factor of around 10 in speed. The most time-consuming module, the average 4D bispectrum calculation, shows a substantial speed increase with a speedup factor of around 17. In the module for the batched 2D FFT, there is a speedup by a factor of about 30 when cuFFT library and multi-thread kernel are used to perform the FFT operation for one subimage burst on GPU. The processing speed of the other two modules is about 3-5 times faster than that of the serial program. From the above comparison results, we can see that the processing speed of several reconstruction steps is increased significantly, and the accelerating performance of the CUDA-based method is superior to that of OpenMP-based method overall.

Figure 4 shows a computing time comparison of one subimage reconstruction between serial program, OpenMP-based method, and CUDA-based method using different sizes of data sets. It can be seen that the computation time of the new CUDA-based parallel method invariably decreases compared to that of the serial program and OpenMP-based method, when the number of data sets becomes larger. The speedup ratio of CUDA-based method maintains around 9-10, while that of OpenMP-based method maintains around 2-3, as the size of data sets increases from set1 to set5. Therefore, the new parallel method based on CUDA in our work becomes more efficient than serial implementation and OpenMP-based parallel method for large scale data processing.

4. Conclusions

In this study, we design and implement a new parallel method for speckle masking reconstruction of solar subimages based on CUDA on GPGPU. Tests are performed to verify the correctness of our program in C language on NVIDIA Tesla C2050. The details of several parallel reconstruction steps are presented, and the parallel implementation among various modules shows a great speed increase as compared to the previous serial implementation. We also compare the runtime of these modules between the new parallel method and OpenMP-based method, and the speedup performance of the new parallel method exceeds that of OpenMP-based method overall. Furthermore, we present a comparison in the consumed time among serial program, OpenMP-based method, and the new parallel method. The new parallel method shows a clear advantage for large scale data processing, and a speedup of around 9-10 is achieved in reconstructing one solar subimage of 256×256 pixels.

The current parallel method is applied to accelerating the reconstruction of one subimage based on CUDA on one GPGPU. In our future work, we will port the code of the CUDA-based implementation to a high performance cluster with more GPUs, and accelerate the reconstruction of all subimages. In summary, our new parallel method would be valuable, and contribute to

real-time reconstruction of a full high resolution solar image.

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 11703009), and the Natural Science Foundation of Jiangsu Province, China (No. BK20170566). The authors gratefully acknowledge the helpful comments and suggestions of the reviewers.

References

Beard, A., Cowan, B., & Ferayorni, A. 2014, DKIST Visible Broadband Imager Data Processing Pipeline, SPIE, 9152, 91521J

CUDA Toolkit, 2018, https://developer.nvidia.com/cuda-toolkit

cuFFT Library, 2018, http://docs.nvidia.com/cuda/cufft

Flores, L. A., Vidal, V., Mayo, P., Rodenas, F., & Verdu, G. 2014, Parallel CT Image Reconstruction Based on GPUs, Radiation Physics and Chemistry, 247, 250

Gonsalves, R. A. 1982, Phase Retrieval and Diversity in Adaptive Optics, Optical Engineering, 21, 829

Labeyrie, A. 1970, Attainment of Diffraction Limited Resolution in Large Telescopes by Fourier Analysing Speckle Patterns in Star Images, A&A, 85, 87

Li, X. B., Wang, F., Xiang, Y. Y., et al. 2014, Parallel Image Reconstruction for New Vacuum Solar Telescope, JKAS, 47, 43

Li, X. B., Liu, Z., Wang, F., et al. 2015, High-Performance Parallel Image Reconstruction for the New Vacuum Solar Telescope, PASJ, 67, 47

Li, X. B., & Zheng, Y. F. 2016, A Novel Parallel Method for Speckle Masking Reconstruction Using the OpenMP, JKAS, 157, 162

Lohmann, A. W., Weigelt, G., & Wirnitzer, B. 1983, Speckle Masking in Astronomy: Triple Correlation Theory and Applications, Applied Optics, 4028, 4037

Paxman, R. G., Schulz, T. J., & Fienup, J. R. 1992, Joint Estimation of Object and Aberrations by Using Phase Diversity, JOSAA, 9, 1072

Pehlemann, E., & von der Lühe, O. 1989, Technical Aspects of the Speckle Masking Phase Reconstruction Algorithm, A&A, 216, 337

Shi, Z., Xiang, Y. Y., Deng, H., Ji, K. F., & Wei, S. L. 2015, A Method of Level 1 Frames-Selection Based on GPU for New Vacuum Solar Telescope, Chinese Science Bulletin, 1408, 1413

von der Lühe, O. 1993, Speckle Imaging of Solar Small Scale Structure. I – Methods, A&A, 268, 374

Wöger, F., & von der Lühe, O. 2008, KISIP: A Software Package for Speckle Interferometry of Adaptive Optics Corrected Solar Data, SPIE, 7019, 70191E

Wöger, F., & Ferayorni, A. 2012, Accelerated Speckle Imaging with the ATST Visible Broadband Imager, SPIE, 8451, 84511C-1