The CUBLAS and CULA based GPU acceleration of adaptive finite element framework for bioluminescence tomography

Bo Zhang,[1] **Xiang Yang,**[2,3] **Fei Yang,**[2] **Xin Yang,**[2] **Chenghu Qin,**[2] **Dong Han,**[2] **Xibo Ma,**[2] **Kai Liu,**[2] **and Jie Tian**[1,2,3,4,5]

[1]*Sino-Dutch Biomedical and Information Engineering School of Northeastern University, Shenyang, 110004, China*
[2]*Medical Image Processing Group, Institute of Automation, Chinese Academy of Sciences, P. O. Box 2728, Beijing, 100190, China*
[3]*Life Sciences Research Center, School of Life Sciences and Technology, Xidian University, Xian 710071, China*
[4]*tian@ieee.org*
[5]*jie.tian@ia.ac.cn*

http://www.mitk.net; http://www.3dmed.net

Abstract: In molecular imaging (MI), especially the optical molecular imaging, bioluminescence tomography (BLT) emerges as an effective imaging modality for small animal imaging. The finite element methods (FEMs), especially the adaptive finite element (AFE) framework, play an important role in BLT. The processing speed of the FEMs and the AFE framework still needs to be improved, although the multi-thread CPU technology and the multi CPU technology have already been applied. In this paper, we for the first time introduce a new kind of acceleration technology to accelerate the AFE framework for BLT, using the graphics processing unit (GPU). Besides the processing speed, the GPU technology can get a balance between the cost and performance. The CUBLAS and CULA are two main important and powerful libraries for programming on NVIDIA GPUs. With the help of CUBLAS and CULA, it is easy to code on NVIDIA GPU and there is no need to worry about the details about the hardware environment of a specific GPU. The numerical experiments are designed to show the necessity, effect and application of the proposed CUBLAS and CULA based GPU acceleration. From the results of the experiments, we can reach the conclusion that the proposed CUBLAS and CULA based GPU acceleration method can improve the processing speed of the AFE framework very much while getting a balance between cost and performance.

OCIS codes: (100.3190) Inverse problems; (170.6960) Tomography; (100.3010) Image reconstruction techniques; (170.6280) Spectroscopy, fluorescence and luminescence.

References and links

1. R. Weissleder and V. Ntziachristos, "Shedding light onto live molecular targets," Nat. Med. **9,** 123–128 (2003).

#128085 - $15.00 USD Received 6 May 2010; revised 22 Jul 2010; accepted 1 Sep 2010; published 8 Sep 2010

2. V. Ntziachristos, J. Ripoll, L. H. V. Wang, and R. Weissleder, "Looking and listening to light: the evolution of whole-body photonic imaging," Nat. Biotechnol. **23,** 313–320 (2005).
3. M. K. So, C. J. Xu, A. M. Loening, S. S. Gambhir, and J. H. Rao, "Self-illuminating quantum dot conjugates for in vivo imaging," Nat. Biotechnol. **24,** 339–343 (2006).
4. R. Weissleder and M. J. Pittet, "Imaging in the era of molecular oncology," Nature **452,** 580–589 (2008).
5. J. K. Willmann, N. van Bruggen, L. M. Dinkelborg, and S. S. Gambhir, "Molecular imaging in drug development," Nat. Rev. Drug Discov. **7,** 591–607 (2008).
6. C. Contag and M. H. Bachmann, "Advances in Bioluminescence imaging of gene expression," Annu. Rev. Biomed. Eng. **4,** 235–260 (2002).
7. W. Cong and G. Wang, "Boundary integral method for bioluminescence tomography," J. Biomed. Opt. **11,** 020503 (2006).
8. C. Qin, J. Tian, X. Yang, K. Liu, G. Yan, J. Feng, Y. Lv, and M. Xu, "Galerkin-based meshless methods for photon transport in the biological tissue," Opt. Express **16,** 20317–20333 (2008), http://www.opticsinfobase.org/oe/abstract.cfm?URI=oe-16-25-20317.
9. B. Zhang, J. Tian, D. Liu, L. Sun, X. Yang, and D. Han, "A multithread based new sparse matrix method in bioluminescence tomography," presented at Conference 7626 of SPIE on Medical Imaging, San Diego, USA, 13–18 February 2010.
10. Y. Lu and A. F. Chatziioannou, "A parallel adaptive finite element method for the simulation of photon migration with the radiative-transfer-based model," Commun. Numer. Methods Eng. **25,** 751–770 (2009).
11. http://developer.nvidia.com/page/home.html
12. http://www.culatools.com/
13. X. Gu, Q. Zhang, L. Larcom, and H. Jiang, "Three-dimensional bioluminescence tomography with model-based reconstruction," Opt. Express **12,** 3996–4000 (2004), http://www.opticsinfobase.org/oe/abstract.cfm?URI=oe-12-17-3996.
14. M. Schweiger, S. R. Arridge, M. Hiraoka, and D. T. Delpy, "The finite element method for the propagation of light in scattering media: Boundary and source conditions," Med. Phys. **22,** 1779–1792 (1995).
15. J. J. Duderstadt and L. J. Hamilton, *Nuclear Reactor analysis*, (Wiley, New York, 1976).
16. S. S. Rao, *The finite element method in engineering*, (Butterworth-Heinemann, Boston, 1999).
17. Y. Lv, J. Tian, W. Cong, G. Wang, J. Luo, W. Yang, and H. Li, "A multilevel adaptive finite element algorithm for bioluminescence tomography," Opt. Express **14,** 8211–8223 (2006), http://www.opticsinfobase.org/abstract.cfm?URI=oe-14-18-8211.
18. W. Cong, D. Kumar, Y. Liu, A. Cong, and G. Wang, "A practical method to determine the light source distribution in bioluminescent imaging," Proc. SPIE **5535,** 679–686 (2004).
19. B. Zhang, X. Yang, C. Qin, D. Liu, S. Zhu, J. Feng, L. Sun, K. Liu, D. Han, X. Ma, X. Zhang, J. Zhong, X. Li, X. Yang, and J. Tian, "A trust region method in adaptive finite element framework for bioluminescence tomography," Opt. Express **18,** 6477–6491 (2010), http://www.opticsinfobase.org/abstract.cfm?uri=oe-18-7-6477.
20. G. Alexandrakis, F. R. Rannou, and A. F. Chatziioannou, "Tomographic bioluminescence imaging by use of a combined optical-PET (OPET) system: a computer simulation feasibility study," Phys. Med. Biol. **50,** 4225–4241 (2005).
21. L. H. Wang, S. L. Jacques, and L. Q. Zheng, "MCML-Monte Carlo modeling of photon transport in multi-layered tissues," Comput. Meth. Prog. Biomed. **47,** 131–146 (1995).
22. D. Boas, J. Culver, J. Stott, and A. Dunn, "Three dimensional Monte Carlo code for photon migration through complex heterogeneous media including the adult human head," Opt. Express **10,** 159–169 (2002), http://www.opticsinfobase.org/abstract.cfm?URI=OPEX-10-3-159.
23. H. Li, J. Tian, F. Zhu, W. Cong, L. V. Wang, E. A. Hoffman, and G. Wang, "A mouse optical simulation enviroment (MOSE) to investigate bioluminescent phenomena in the living mouse with the Monte Carlo Method," Acad. Radiol. **11,** 1029–1038 (2004).

1. Introduction

1.1. Bioluminescence tomography

In small animal imaging, optical MI, especially bioluminescence imaging (BLI), has many advantages in probing capabilities, sensitivity, specificity, and cost-effectiveness [1–3]. These advantages can prompt BLI's use in cancer research [4] and drug development [5]. In BLI, biological entities, such as tumor cells, genes and compounds of drug, are tagged with luciferase enzymes. When the luciferase is combined together with the substrate luciferin, oxygen and ATP, a biochemical reaction that transforms part of the chemical energy into the bioluminescent photons with a wavelength of about 600*nm* [6] occurs. But BLI can only give the 2D in-

#128085 - $15.00 USD Received 6 May 2010; revised 22 Jul 2010; accepted 1 Sep 2010; published 8 Sep 2010

formation of the object being imaged. Bioluminescence tomography (BLT) is the 3D imaging modality of BLI, it can reconstruct the inner bioluminescent light source distribution, according to the surface light distribution that is acquired the same way in BLI. Besides the boundary integral method [7] and the element free method [8], the finite element method (FEM), especially the adaptive finite element (AFE) framework, is usually adopted in BLT.

1.2. Graphics processing unit (GPU)

The graphics processing unit (GPU, also occasionally called visual processing unit or VPU) is a processor attached to a graphics card dedicated to calculating floating point operations. It is usually used in embedded systems, mobile phones, personal computers, workstations and game consoles. Modern GPU is very efficient at manipulating computer graphics, and their highly parallel structure makes them more effective than general-purpose central processing unit (CPU) for a range of complex algorithms on parallelizable floating point operations. In other words, GPU is a CPU that is especially powerful when dealing with the images. When dealing with the images, the efficiency of GPU is much higher than that of CPU. But GPU can not take the place of CPU, as CPU is a general purpose processor and is powerful when handling numerical computations.

To our best knowledge, there are 3 kinds of hardware acceleration strategies. The first one is the multi-thread CPU technology using many CPU cores based on one shared memory computer. The second one is the multi CPU technology using parallel computation technology based on high-performance computer clusters. The last one is the GPU technology based on the graphic cards. The multi-thread CPU technology [9] and the multi CPU technology [10] has already been applied to improve the AFE framework for BLT. As the operations in the AFE framework are mainly link list related, it is easy to be accelerated by the multi-thread CPU technology. Although the high-performance computer clusters used in literature [10] can improve the processing speed very much, the high cost of owning and maintaining makes them difficult to access for most researchers and clinical users. The multi-thread workstation used in literature [9] is not that powerful. However, GPU can make a balance between the cost and performance. GPU can get better performance than the multi-thread workstation and lower cost than the high-performance computer clusters when handling the parallelizable floating point operation.

We've discovered that in the relation forming part of the AFE framework there are some time costing floating point operations and those operations are parallel. So we propose GPU technology to accelerate the AFE framework for the parallelizable floating point operations.

1.3. CUBLAS and CULA libraries

To the best of our knowledge, there are 2 libraries for programming on NVIDIA GPU. One is the CUBLAS that is provided by NVIDIA [11], for some basic linear algebra operations. The other is the CULA that is provided by EM Photonics [12] for some advanced numeric linear algebra operations. With the help of the CUBLAS and CULA, developers on NVIDIA GPU can focus on the mathematical procedure of their algorithms rather than the hardware environment of NVIDIA GPU.

The paper is organized in the following sequence. In Section 2, we firstly introduce the diffuse approximation (DA) for the BLT forward problem. Then a brief introduction of the AFE framework will be presented for the BLT inverse problem in Subsection 2.2. Then our main work on the CUBLAS and CULA based GPU acceleration will be detailedly presented in Subsection 2.3. In Section 3, we will firstly describe the experimental setup. Then a set of experiments on the need and feasibility of the GPU acceleration will be presented in Subsection 3.2. Then a set of experiments on the effect of the GPU acceleration will be shown in

#128085 - $15.00 USD Received 6 May 2010; revised 22 Jul 2010; accepted 1 Sep 2010; published 8 Sep 2010

Subsection 3.3. In the final part of the experiments section, a source reconstruction experiment with the proposed acceleration strategy will be presented to show its application in BLT in Subsection 3.4. In the final part of the whole paper, we will give our comments and conclude the paper in Section 4.

2. Method

2.1. Diffusion approximation

In the near infrared light spectrum, photon scattering predominates over absorption in the biological tissue, the photon propagation can be modelled by diffusion approximation (DA) of the radiative transfer equation (RTE) [13]. The steady-state domain diffusion equation (DE) is:

$$\nabla\cdot\big(D(\mathbf{x})\nabla\Phi(\mathbf{x})\big)+\mu_a(\mathbf{x})\Phi(\mathbf{x})=S(\mathbf{x})\ (\mathbf{x}\in\Omega) \tag{1}$$

In DE, Ω stands for the problem domain, $\Phi(\mathbf{x})$ is the photon flux density [$Watts/mm^2$], $S(\mathbf{x})$ is the bioluminescent source power density [$Watts/mm^3$], $\mu_a(\mathbf{x})$ is the absorption coefficient [mm^{-1}] and $D(\mathbf{x})=(3(\mu_a(\mathbf{x})+\mu_s'(\mathbf{x})))^{-1}$ is the optical diffusion coefficient [mm], where $\mu_s'(\mathbf{x})=(1-g)\mu_s(\mathbf{x})$ is the reduced scattering coefficient, $\mu_s(\mathbf{x})$ is the scattering coefficient [mm^{-1}] and g is the anisotropy parameter.

For preventing the light from other light sources, the bioluminescence imaging experiments are usually performed in a totally dark environment. That is to say no external photon can travel into Ω through the boundary $\partial\Omega$. The Robin type boundary condition can be used [14, 15].

$$\Phi(\mathbf{x})+2A(\mathbf{x};n,n')D(\mathbf{x})\big(\mathbf{v}(\mathbf{x})\cdot\nabla\Phi(\mathbf{x})\big)=0\ (\mathbf{x}\in\partial\Omega) \tag{2}$$

2.2. Finite element method and the adaptive finite element framework

According to the finite element method (FEM) [16], in the Sobolev space $H^1(\Omega)$, we can get the weak solution of the flux density $Q(\mathbf{x})$ through Eqs. (1) and (2):

$$\begin{aligned}&\int_\Omega\big(D(\mathbf{x})\big(\nabla\Phi(\mathbf{x})\big)\cdot(\nabla\Psi(\mathbf{x})\big)+\mu_a(\mathbf{x})\Phi(\mathbf{x})\Psi(\mathbf{x})\big)\,d\mathbf{x}+\\ &\int_{\partial\Omega}\frac{1}{2A(\mathbf{x};n,n')}\Phi(\mathbf{x})\Psi(\mathbf{x})d\mathbf{x}=\int_\Omega S(\mathbf{x})\Psi(\mathbf{x})d\mathbf{x}\ \big(\forall\Psi(\mathbf{x})\in H^1(\Omega)\big)\end{aligned} \tag{3}$$

According to the adaptive finite element (AFE) framework introduced by Lv et al. [17], the matrix form of Eq. (3) for the l-th level of the mesh refinement process can be got as:

$$([K_l]+[C_l]+[B_l])\Phi_l=M_l\Phi_l=F_lS_l \tag{4}$$

In Eq. (4), the components of the matrices K_l, C_l and B_l can be obtained by

$$\begin{cases}k_{ij}^{(l)}=\int_\Omega D(\mathbf{x})\big(\nabla\varphi_i^{(l)}(\mathbf{x})\big)\cdot\big(\nabla\varphi_j^{(l)}(\mathbf{x})\big)\mathrm{d}\mathbf{x}\\ c_{ij}^{(l)}=\int_\Omega\mu_a(\mathbf{x})\varphi_i^{(l)}(\mathbf{x})\varphi_j^{(l)}(\mathbf{x})\mathrm{d}\mathbf{x}\\ b_{ij}^{(l)}=\int_{\partial\Omega}\varphi_i^{(l)}(\mathbf{x})\varphi_j^{(l)}(\mathbf{x})/\big(2A(\mathbf{x};n,n')\big)\mathrm{d}\mathbf{x}\\ s_{ij}^{(l)}=\int_\Omega s_i^{(l)}\varphi_i^{(l)}(\mathbf{x})\varphi_j^{(l)}(\mathbf{x})\mathrm{d}\mathbf{x}\end{cases}$$

$k_{ij}^{(l)}, c_{ij}^{(l)}, b_{ij}^{(l)}$ and $s_{ij}^{(l)}$ are the elements of $K^{(l)}, C^{(l)}, B^{(l)}$ and $S^{(l)}$ with the row number i and the column number j, respectively. $\varphi_i^{(l)}(\mathbf{x})$ and $\varphi_j^{(l)}(\mathbf{x})$ are the interpolation basis functions. $s_i^{(l)}$ is the source density at i.

#128085 - $15.00 USD Received 6 May 2010; revised 22 Jul 2010; accepted 1 Sep 2010; published 8 Sep 2010

M_l is a symmetric positive-definite matrix and is invertible, while F_l is a nonsymmetric matrix and is not invertible. So Eq. (4) can be transformed to:

$$\Phi_l = M_l^{-1} F_l S_l = A_l S_l \tag{5}$$

2.3. GPU acceleration

The hardware structure of the NVIDIA GPU is shown is Fig. 1. On the graphic cards, there are N multiprocessors, while each multiprocessor contains M processors. The device memory is also called the global memory that can be accessed by all the processors in every multiprocessor. The shared memory of a certain multiprocessors can only be accessed by the processors inside the multiprocessors. As the shared memory is much faster than the global memory, the main programming skill on GPU is to divide the whole task into N absolutely irrelevant parts and process each part in one multiprocessor.

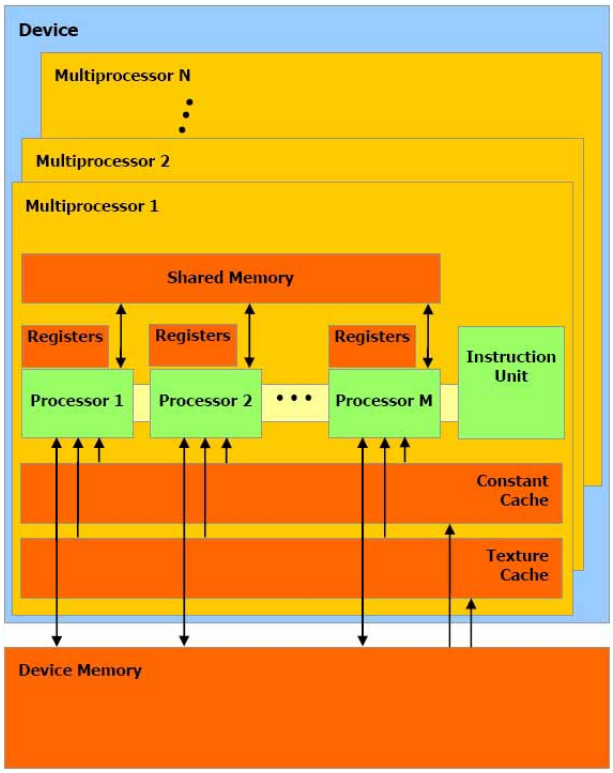

Fig. 1. The hardware model of GPU [11].

The typical programming model can be illustrated as:

(1) Transfer data from host memory (the system memory that is accessed by the CPU) to device memory. That is also to say transfer data from the system memory to the device memory on the graphic card.

(2) Divide the whole task into N parts.

(3) Load data from the device memory to the shared memory.

(4) Synchronize all the processors.

(5) Process the operations on the data in the shared memory.

(6) Write the results back to the device memory.

#128085 - $15.00 USD Received 6 May 2010; revised 22 Jul 2010; accepted 1 Sep 2010; published 8 Sep 2010

(7) Transfer the results in the device memory back to the host memory.

As the operations for getting $A_l = M_l^{-1} F_l$ in Eq. (5) are parallel and on floating point, we decide to use the GPU technology to accelerate them. The execution flow chart is shown in Fig. 2. The matrix M short for M_l^{-1} in Eq. (5) is firstly read from the system memory to the device memory on the graphic card, and then decomposed into a lower triangular matrix L and an upper triangular matrix U. After solving the linear equation $LY = I$, where matrix I is the identity matrix, we can get matrix Y. Then the linear equation $UM^{-1} = Y$ is solved for M^{-1}. Matrix F short for matrix F_l in Eq. (5) is read into the global memory for the multiplication. Then the matrix M^{-1} and F are divided into N parts and sent to the shared memory of each multiprocessor for the multiplication operation.

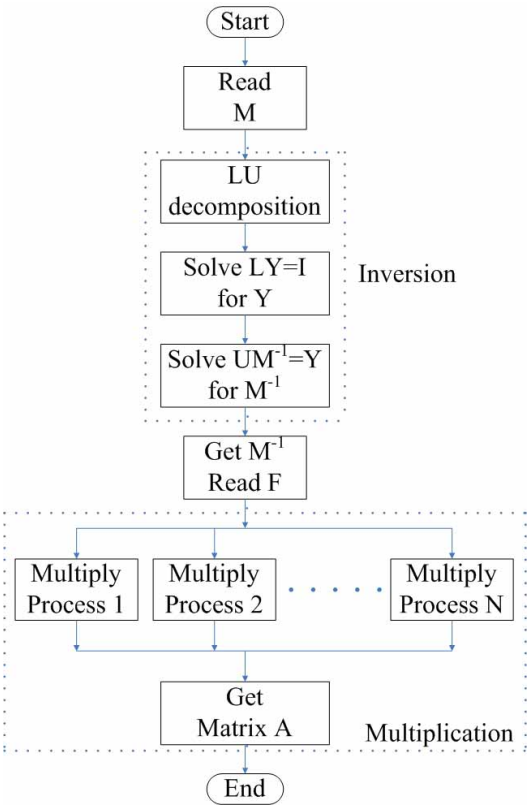

Fig. 2. The execution flow chart of matrix inversion and multiplication using GPU.

2.3.1. CUDA and CUBLAS

Coding on GPU in the graphics cards is not that simple as coding on CPU, as a lot of knowledge relating hardware and software environment are needed. So in November 2006, NVIDIA introduced a general purpose parallel computing architecture, compute unified device architecture (CUDA) for the NVIDIA graphics cards [11]. CUDA leverages the parallel compute engine in NVIDIA GPU to solve many parallelizable floating point operations in a more efficient way than on a CPU. CUDA comes with a software environment that the developers can use C as a high-level programming language.

Besides the CUDA, NVIDIA has also provided a CUBLAS library on top of the CUDA driver for the developers to do some basic linear algebra operations. CUBLAS is an implementation

#128085 - $15.00 USD Received 6 May 2010; revised 22 Jul 2010; accepted 1 Sep 2010; published 8 Sep 2010

of basic linear algebra subprograms (BLAS) and the "CU" in CUBLAS stands for CUDA. The CUBLAS library is self-contained at the application programming interface (API) level, that is to say no direct interaction with the CUDA driver is needed. The multiplication operation of $A_l = M_l^{-1} F_l$ in Eq. (5) is implemented by using the CUBLAS.

2.3.2. CULA

As there is no matrix inversion operation in CUBLAS, we have to ask help from CULA. The CULA is a GPU accelerated linear algebra library that utilizes the NVIDIA CUDA parallel computing architecture to dramatically improve the computation speed of sophisticated mathematics [12]. CLUA is an implementation of the Linear Algebra PACKage (LAPACK) interface for CUDA enabled NVIDIA GPU.

The CULA is a next generation linear algebra package that uses the GPU as a co-processor to achieve speedups over existing linear algebra packages. The CULA provides the same functionality you receive with your existing package, only at a greater speed. The CULA provides easy access to the NVIDIA computing resources available in the computer system. The CULA library is a self-contained package that enhances linear algebra programs with little or no knowledge of the GPU computing model.

Besides many advances that CULA has, the disadvantage is that it is a commercial library and only a small basic part of the library is free. But as there is a GESV (in the library it is called culaSgesv) function that can compute the solution to a real system of linear equations of $AX = B$ using the LU decomposition with partial pivoting and row interchanges, we can set B as an identity matrix to get X as the invert matrix of A.

The version that we use is CULA 1.2 Basic. As CULA 1.2 Basic is built on NVIDIA CUDA 2.3 and CUBLAS, the NVIDIA CUDA 2.3 and CUBLAS are used in the paper.

2.4. Trust region method for the optimization

After applying the permissible source region method [18], the linear relationship between the boundary measured photon flux density Φ_l^{meas} and the unknown source density in the permissible source region S_l^P can be obtained:

$$\Phi_l^{meas} = A_l^{ps} S_l^P \tag{6}$$

In Eq. (6), A_l^{ps} can be obtained by retaining those columns of $M_l^{-1} F_l$ corresponding to S_l^P and those rows corresponding to Φ_l^{meas}. Then, the objective function $f^l(x)$ of the l-th level can be got as

$$f^l(S_l^P) = \|A_l^{ps} S_l^P - \Phi_l^{meas}\|_2^2 \tag{7}$$

The optimization problem of Eq. (7) is solved by the trust region method (TRM) [19].

3. Experiments

3.1. Numerical simulation setup

For certificating the proposed method, we designed a heterogeneous cylindrical phantom. The cylindrical phantom was 30*mm* in height, 10*mm* in radius and consisted of four ellipsoids and one cylinder to represent muscle, lungs, heart, bone and liver, as shown in Fig. 3(a). The optical parameters of the phantom were all obtained from the literature [20] and listed in Table 1.

As well acknowledged by the community, the Monte Carlo (MC) method was a gold standard for photon transportation simulation because of its accuracy and flexibility [21,22]. So we used the MC method based molecular optical simulation environment (MOSE) [23] that could take 2D/3D analytical models, micro-CT and micro-MRI slices to define the object geometry to get

#128085 - $15.00 USD Received 6 May 2010; revised 22 Jul 2010; accepted 1 Sep 2010; published 8 Sep 2010

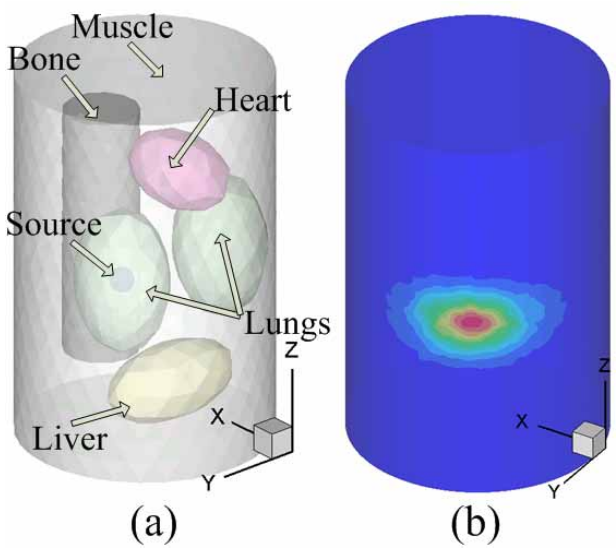

Fig. 3. Sub figure (a) is the heterogeneous cylindrical numerical phantom with single source, consisted of muscle (white), bone (black), heart (pink), lungs (green), liver (yellow) and a ball source (blue) in the right lung. Sub figure (b) is the surface light power distribution of the phantom in sub figure (a), which is generated by MOSE.

the surface photon distribution data. Furthermore, the MC method could avoid the *inverse crime* problem.

Table 1. Optical parameters of different tissues of the heterogeneous cylindrical phantom

Material	Muscle	Lung	Heart	Bone	Liver
$\mu_a[mm^{-1}]$	0.010	0.350	0.200	0.002	0.035
$\mu_s[mm^{-1}]$	4.000	23.000	16.000	20.000	6.000
g	0.900	0.940	0.850	0.900	0.900

During the MOSE simulation procedure, the bioluminescence light source was sampled by 10^6 photons and was assumed to obey the uniform distribution. The aforementioned heterogeneous phantom was discretized into 34072 triangles and 11499 surface measurement points with an average element diameter of about $0.5mm$ for MOSE simulation. For showing the need, feasibility and effect of the GPU acceleration, we discretized the cylindrical phantom with different average element diameter to form different dimension of the matrix M_l and F_l in Eq. (4) as shown in Table 2 and Table 3. The matrix M_l was a point number by point number symmetric matrix, while the matrix F_l was a point number by element number nonsymmetric matrix.

A solid sphere source of $1mm$ in radius and $0.238nano-Watts/mm^3$ in power density was centered at $(3,5,15)$ inside the right lung as shown in Fig. 3(a). To reduce the *ill-posedness* of the BLT inverse problem, we incorporated the permissible source region of

$$PS = \{\,(x,\,y,\,z)\,|\,13 < z < 17,\ (x,\,y,\,z) \in \textit{Right Lung}\}$$

as *a priori* information, according to the surface light distribution as shown in Fig. 3(b).

#128085 - $15.00 USD Received 6 May 2010; revised 22 Jul 2010; accepted 1 Sep 2010; published 8 Sep 2010

3.2. The need and feasibility of the GPU acceleration

We carried out a set of experiments on an Intel(R) Core(TM)2 Duo E4600 CPU (2.4GHz) platform with 2GB memory and a NVIDA Geforce GT240 graphics card (this hardware environment was called En. 1 for short) for showing the main time consuming parts in the AFE framework (these set of experiments were called Ex. 1 for short). The results were shown in Table 2. In Ex. 1, no mesh refinement of the AFE framework was performed.

In Table 2, "–" denoted that there was not enough system memory for holding the matrix M_l or F_l in Eq. (4). As a result, the calculation could not go on and there was no result.

In Ex. 1, all the operations of the AFE framework were single thread except for the "Projection" operation. The "Projection" operation was designed to project the light power density on the nodes of the fine mesh that were used in MOSE simulation with an average element diameter of about 0.5mm to the coarse meshes used in Ex. 1. The projection operation was already accelerated with the multi-thread technology. In Ex. 1, as there were 2 CPU cores in En. 1, the projection operation time in Table 2 were about half of those when single core was present.

The matrix inversion operation of matrix M_l in Ex. 1 was performed in the following sequence. Matrix M_l was first decomposed by the Cholesky factorization $M_l = LL^T$, where L was a lower triangle matrix. Then the inverse matrix $M_l^{-1} = (L^T)^{-1}L^{-1}$ could be easily got, as the inversion of the lower triangle matrix was easy to be got.

In the "Others Time" column, as the permissible source region method was incorporated and the permissible source region used were relatively small, the optimization time was short. But once the dimensions of matrix M_l and F_l were fixed, the time of matrix inversion and multiplication were also fixed.

Table 2. Statistics of main time consuming modules in the AFE framework

Experiment No.	Point No.	Element No.	Inversion Time (s)	Multiplication Time (s)	Projection Time (s)	Others Time (s)
1	1023	4565	2.1	51.8	35.2	3.1
2	1537	6878	7.7	173.9	34.7	5.0
3	1812	7965	12.9	282.5	36.8	7.3
4	2620	11253	34.0	837.7	34.7	12.8
5	3021	15095	55.9	1843.6	39.4	54.6
6	3517	15257	87.0	2261.3	45.4	41.3
7	4121	18286	143.2	–	–	–
8	5028	23579	–	–	–	–

From the data in Table 2, we could see that the matrix inversion and multiplication operations were the most time consuming operations in the AFE framework. For further analyzing, we could define $\mathscr{P}$ as the matrix inversion and multiplication time percentage for each case in Ex. 1 as:

$$\mathscr{P} = \frac{the\ matrix\ inversion\ and\ multiplication\ time}{total\ time\ of\ one\ mesh\ refinement\ step\ in\ the\ AFE\ framework} \times 100\,\%$$

The total time of one mesh refinement step in the AFE framework consisted of the time of matrix inversion, multiplication, projection and others. As shown in Fig. 4, as the matrix dimensions of M_l and F_l increased, the $\mathscr{P}$ in each case were also increasing. As a result, we could reach the conclusion that the GPU acceleration was necessary and feasible.

#128085 - $15.00 USD Received 6 May 2010; revised 22 Jul 2010; accepted 1 Sep 2010; published 8 Sep 2010

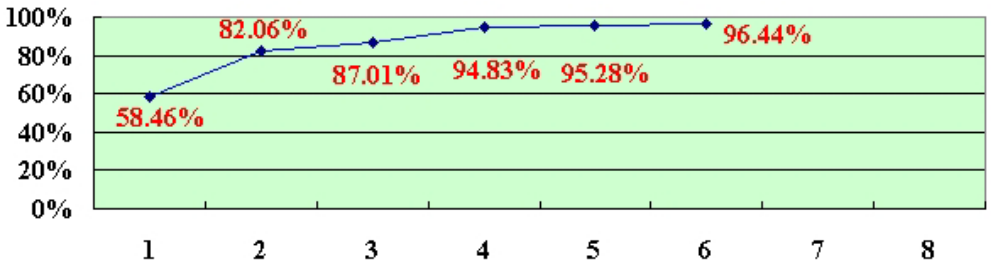

Fig. 4. The $\mathscr{P}$ in each case in Ex. 1. The horizontal axis in the figure was the case order and the vertical axis was $\mathscr{P}$.

3.3. The acceleration comparisons

Besides Ex. 1, we also carried out a set of experiments on an Intel (R) Xeon (TM) CPU (3.20GHz) workstation with 16GB memory (this hardware environment had 8 CPU cores and was called En. 2 for short) and En. 1, for showing the acceleration comparisons between multi-thread CPU technology and GPU technology (these set of experiments were called Ex. 2 for short). The results were shown in Table 3.

In Table 3, there were 2 modes in the "Acceleration Mode" column, the "CPU 8" and "GPU". "CPU 8" denoted that the experiment was performed in En. 2 with all the 8 cores involved in the calculation and the calculation had been accelerated by the multi-thread ZSM technique [9]. "GPU" denoted that the experiment was performed in En. 1 with the matrix inversion and matrix multiplication operations performed in the graphics card.

In Table 3, "–" in the "Inversion Time" column denoted that there was not enough memory in the system for holding the matrix M_l in Eq. (4); "G–" in the "Multiplication Time" column denoted that there was not enough memory in the graphics card for holding the matrix F_l; "–G–" in the "Multiplication Time" column denoted that there was enough memory neither in the system nor in the graphics card for holding the matrix F_l. As a result, the calculation could not continue and there was no result. The reason that there was enough memory in the system for holding the matrix M_l and F_l in the "CPU 8" "Mode" was that in "CPU 8" "Mode" the matrix M_l and F_l were stored in a sparse matrix data structure using the ZSM technique [9].

From Table 3, we could see that the processing time of matrix inversion and multiplication were reduced by the GPU acceleration technology and the multiplication time were reduced dramatically. In order to analyze the effect of the acceleration, we defined the speed up as:

$$SU = \frac{time\ cost\ of\ operation\ before\ the\ acceleration}{time\ cost\ of\ operation\ after\ the\ acceleration}$$

So we could have the

$$SU_{Cpu8GpuInv} = \frac{time\ cost\ of\ matrix\ inversion\ when\ 8\ CPU\ presented}{time\ cost\ of\ matrix\ inversion\ when\ using\ GPU}$$

and

$$SU_{Cpu8GpuMul} = \frac{time\ cost\ of\ matrix\ multiplication\ when\ 8\ CPU\ presented}{time\ cost\ of\ matrix\ multiplication\ when\ using\ GPU}$$

to describe the acceleration of GPU compared with the multi-thread CPU technology. We could also have the

$$SU_{CpuGpuInv} = \frac{time\ cost\ of\ matrix\ inversion\ when\ 1\ CPU\ presented}{time\ cost\ of\ matrix\ inversion\ when\ using\ GPU}$$

#128085 - $15.00 USD Received 6 May 2010; revised 22 Jul 2010; accepted 1 Sep 2010; published 8 Sep 2010

Table 3. Comparisons between multi-thread CPU acceleration and GPU acceleration on matrix inversion and multiplication

Experiment No.	Point No.	Element No.	Acceleration Mode	Inversion Time (s)	Multiplication Time (s)
1	1023	4565	CPU 8	28.875	55.765
			GPU	1.391	0.281
2	1537	6878	CPU 8	96.094	129.672
			GPU	2.843	0.844
3	1812	7965	CPU 8	120.25	207.172
			GPU	3.953	1.219
4	2620	11253	CPU 8	322.953	415.016
			GPU	10.156	3.141
5	3021	15095	CPU 8	444.235	662.219
			GPU	15.047	G–
6	3517	15257	CPU 8	635.016	734.453
			GPU	23.094	G–
7	4121	18286	CPU 8	845.563	1094.468
			GPU	36.387	–G–
8	5028	23579	CPU 8	1104.172	1506.812
			GPU	–	–G–

and

$$SU_{CpuGpuMul} = \frac{time\ cost\ of\ matrix\ multiplication\ when\ 1\ CPU\ presented}{time\ cost\ of\ matrix\ multiplication\ when\ using\ GPU}$$

to describe the acceleration of GPU compared with the original single-thread CPU technology. The results were shown in Fig. 5. Sub figures (a) and (b) in Fig. 5 were obtained according to Table 3, while sub figures (c) and (d) in Fig. 5 were obtained according to Table 2 and 3.

From Fig. 5, we could see that although the speed up of matrix inversion was not very great, the speed up of matrix multiplication was significant. The reason was that matrix multiplication was highly parallel, which made it more easily for the GPU to achieve significant acceleration.

From Fig. 5(a) and 5(b), we could see that both of the curves in the sub figures declined. The matrix M_l and F_l got sparser as the dimension of the matrix increased, so from case 1 to 8 the matrix M_l and F_l became sparser. As the multi-thread ZSM technique took the advantage of the sparse property of matrix M_l and F_l while the GPU technology not, the curves declined. We could also see that although the curves declined, the speed up was good.

From Fig. 5(c) and 5(d), we could see that both of the curves in the sub figures ascended. That was to say compared with the single thread matrix inversion and multiplication on dense stored matrix, the GPU acceleration could give out excellent performance when dealing with large dimensional matrix.

After all, the GPU acceleration could speed up the matrix inversion and multiplication and worked very well on accelerating the AFE framework.

One thing that we wanted to explain was the reason why we had carried out the experiments in Subsections 3.2 and 3.3 on 2 hardware environments, En. 1 and En. 2. That was because the graphics card that we used in En. 1 could not be installed on the main board of En. 2.

One more thing that we wanted to explain was the reason why the processing time of the multi-thread acceleration in Table 3 was longer than the single-thread ones in Table 2. That was because the CPU in En. 1 was much better than the CPU in En. 2. If the experiments had been

#128085 - $15.00 USD Received 6 May 2010; revised 22 Jul 2010; accepted 1 Sep 2010; published 8 Sep 2010

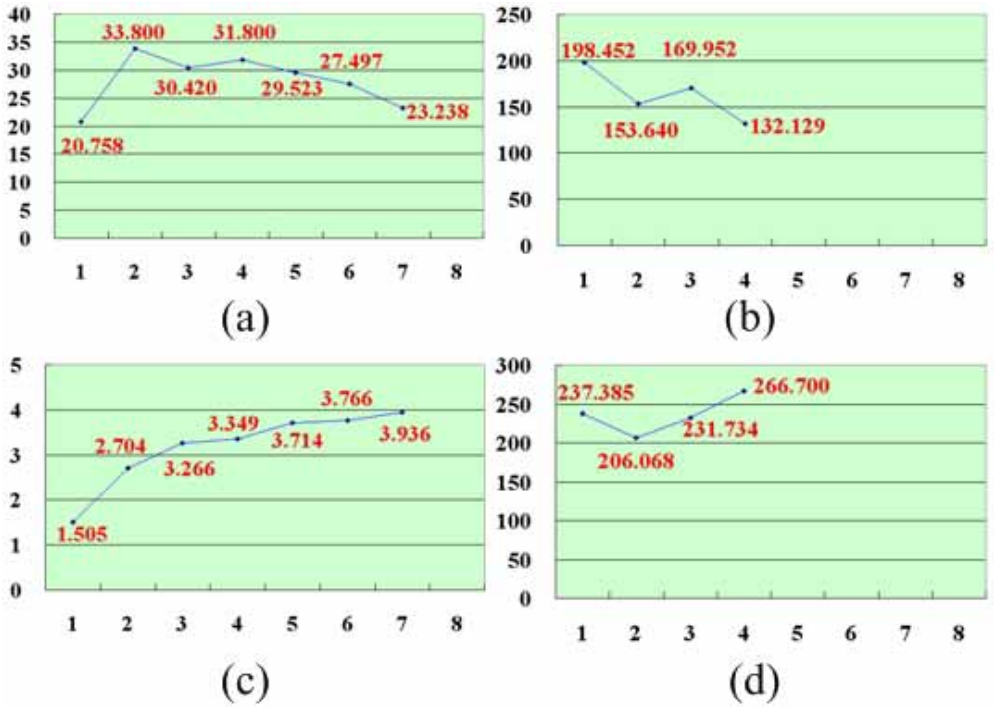

Fig. 5. GPU speed up to CPU. Sub figures (a) to (d) were $SU_{Cpu8GpuInv}$, $SU_{Cpu8GpuMul}$, $U_{CpuGpuInv}$ and $SU_{CpuGpuMul}$ for each case of Ex. 2, respectively. The horizontal axis in all the sub figures was the case order. The vertical axis of sub figures (a) to (d) were $SU_{Cpu8GpuInv}$, $SU_{Cpu8GpuMul}$, $U_{CpuGpuInv}$ and $SU_{CpuGpuMul}$, respectively.

carried on the same hardware platform, the multi-thread acceleration would have been better than the single-thread one when more CPU cores were presented as shown in literature [9].

3.4. Source reconstruction

Besides the analyzing experiments in Subsections 3.2 and 3.3, we had carried out a numerical single source reconstruction experiment using the GPU accelerated AFE framework with TRM handling the optimization procedure. The mesh refinement threshold was set to 7×10^{-3}. After one step of mesh refinement procedure of the AFE framework, we got the reconstruction results as shown in Fig. 6. The reconstructed power density was 0.271 and the reconstructed position was $(-3.316, 4.816, 13.432)$. The time cost of GPU acceleration on matrix inversion and multiplication in every mesh refinement procedure of the AFE framework were shown in Table 4. The reconstruction time of the first and the second step of the AFE framework using the trust region method were 0.203s and 3.906s respectively.

One thing that we wanted to explain was the reason why we had only carried out a numerical reconstruction experiment even though we had carried out real experiments on phantoms and nude mouse (refer literature [19] for more information). That was because the graphics card that we had on hand only had 512*MB* memory, which could not contain large matrix. In real experiments on phantoms and nude mouse, the matrix were all large. That was also the reason that we could not get acceleration time of case 6 to 8 in Ex. 2. The only solution was to incorporate graphics cards with large memory and Tesla that was introduced by NVIDIA was surely a good choice [11]. Tesla could have up to 6*GB* memory, which would be enough for small

#128085 - $15.00 USD Received 6 May 2010; revised 22 Jul 2010; accepted 1 Sep 2010; published 8 Sep 2010

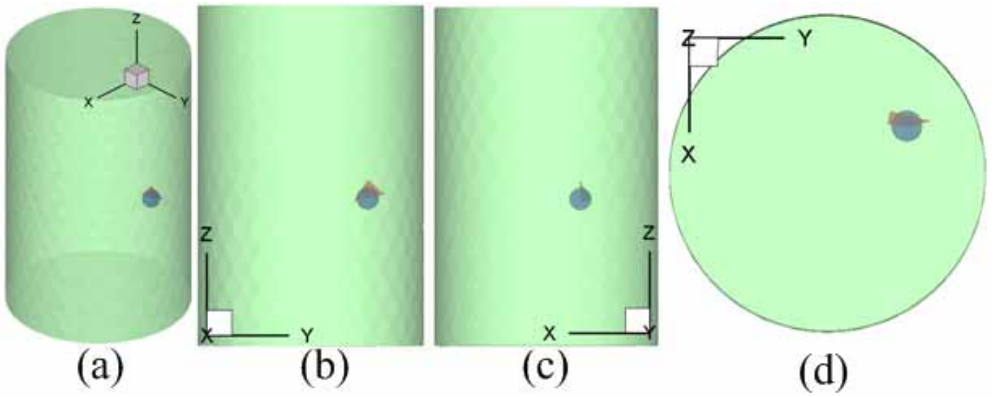

Fig. 6. Reconstruction results of single source numerical experiment. Sub figures (a) to (d) were the 3D views, front views, side views and top views, respectively. The blue ball in each sub figure denoted the real source and the red tetrahedron denoted the reconstructed source with the maximum density. For concision, only the real source and the reconstructed source were displayed.

animal experiments.

Table 4. Time cost of GPU acceleration on matrix inversion and multiplication for single source reconstruction experiment in every mesh refinement procedure of the AFE framework

Step No.	Point No.	Element No.	Inversion Time (s)	Multiplication Time (s)
1	1537	6878	2.734	0.828
2	2346	11384	6.906	2.703

4. Discussions and conclusions

The CUBLAS and CULA based GPU acceleration technology has been proposed the first time for the AFE framework in BLT, for getting a balance between cost and performance when dealing with the parallelizable floating point operations.

In order to evaluate the need and feasibility of the GPU acceleration, we've carried out a set of experiments on the main time consuming operations in the AFE framework. From the results of the experiments, we can reach the conclusion that besides the projection operation and the optimization operation, the matrix inversion and multiplication operations are the main time consuming operations and these operations are all parallelizable floating point operations.

In order to evaluate the effect of the GPU acceleration, we've carried out a set of comparison experiments on single thread operations, multi-thread accelerated operations and GPU accelerated operations. The results of the experiments can lead us to the conclusion that the GPU acceleration can improve the processing speed of the AFE framework very much.

In order to show the application of the proposed GPU acceleration, we've carried out a single source reconstruction on numerical phantom. Although we've carried out real experiments on phantoms and nude mouse, as the memory of the graphics card that we use is limited, the results are not shown in this paper. In the future, we will investigate a Tesla and carry out more biological experiments.

#128085 - $15.00 USD Received 6 May 2010; revised 22 Jul 2010; accepted 1 Sep 2010; published 8 Sep 2010

To sum up, the results of all the performed experiments can convince that the GPU acceleration works very well in the AFE framework for BLT. The processing speed of the AFE framework has been improved very much. The GPU technology can cooperate with multi-thread CPU technology to get high performance while keeping low cost.

Acknowledgments

This paper is supported by the Project for the National Basic Research Program of China (973) under Grant No.2006CB705700, Changjiang Scholars and Innovative Research Team in University (PCSIRT) under Grant No.IRT0645, CAS Hundred Talents Program, CAS scientific research equipment develop program under Grant No. YZ200766, the Knowledge Innovation Project of the Chinese Academy of Sciences under Grant No. KGCX2-YW-129, KSCX2-YW-R-262, the National Natural Science Foundation of China under Grant No. 30672690, 30600151, 60532050, 60621001, 30873462, 60910006, 30970769, 30970771, Beijing Natural Science Fund under Grant No.4071003, Science and Technology Key Project of Beijing Municipal Education Commission under Grant No.KZ200910005005.

#128085 - $15.00 USD Received 6 May 2010; revised 22 Jul 2010; accepted 1 Sep 2010; published 8 Sep 2010