GPU accelerated simplified harmonic spherical approximation equations for three-dimensional optical imaging

Shenghan Ren (任胜寒), Xueli Chen (陈雪利), Xu Cao (曹 旭), Shouping Zhu (朱守平), and Jimin Liang (梁继民)*

Engineering Research Center of Molecular and Neuro Imaging of Ministry of Education & School of Life Science and Technology, Xidian University, Xi'an 710071, China

**Corresponding author: jimleung@mail.xidian.edu.cn*

Received March 3, 2016; accepted April 22, 2016; posted online May 1, 2016

Simplified spherical harmonics approximation (SPN) equations are widely used in modeling light propagation in biological tissues. However, with the increase of order N, its computational burden will severely aggravate. We propose a graphics processing unit (GPU) accelerated framework for SPN equations. Compared with the conventional central processing unit implementation, an increased performance of the GPU framework is obtained with an increase in mesh size, with the best speed-up ratio of 25 among the studied cases. The influence of thread distribution on the performance of the GPU framework is also investigated.

OCIS codes: 170.3660, 170.7050, 200.4960.
doi: 10.3788/COL201614.071701.

Three-dimensional (3D) optical imaging technology has been widely used in the field of biomedical imaging because of its significant advantages of noninvasive detection, high temporal resolution, and low cost[1–3]. The major application of 3D optical imaging is to study the propagation of light in biological tissues. This can be accurately described by the radiative transport equation (RTE) and the Monte Carlo (MC) method[4–6]. However, the RTE is difficult to solve analytically, especially for complex structures. The MC method is time consuming, as it requires a large number of photons for reliable simulation results[7]. To facilitate the development of 3D optical imaging, the diffusion equation (DE)[8,9], the first order of the simplified spherical harmonics approximation (SPN) of the RTE, has been widely used in modeling the light propagation in biological tissues because of its high computational efficiency. With the assumption of light propagating diffusively, the DE is not accurate when the observed regions are high absorption or low scattering tissues[10]. In the past decades, the SPN equations, the high order approximations for the RTE, have been studied by many groups[11–13]. Compared to DE, these equations provide much more accurate and reliable results for tissues with a larger variation of optical parameters, especially for high absorption and low scattering regions. The accuracy of the SPN equations for describing light propagation increases with its order N. However, with the increase of the order N, the computational burden would severely aggravate, which limits the applications of high order SPN equations in 3D optical imaging.

To reduce the computational burden of SPN equations, Li *et al.* proposed an extended finite element method (FEM) for acceleration[14]. Lu *et al.* proposed a parallel adaptive mesh evolution strategy to improve both the modeling precision and simulation speed[12]. However, they were implemented on a central processing unit (CPU) based on moderately parallel systems. Compared with the parallel system based on multi-core CPUs or multi-core clusters, the graphics processing unit's (GPU) based acceleration technique has a much higher raw processing power as well as a relatively lower cost. With the rapid development of the GPU hardware, the GPU-based acceleration technique has been applied to accelerate the MC simulations[15], making it a powerful tool for the simulation of light propagation in tissues. Taking the advantages of the GPU technique, a GPU-accelerated finite element solver for the DE was implemented[16], which was used for calculating the forward model of diffusion optical tomography. However, because of the inherent characteristics of the DE, the acceleration performance was not significant, and the applicability was limited, especially when the tissues are of high absorption or low scattering.

In this Letter, to facilitate the application of the SPN equations to be more efficient, a GPU-accelerated framework is proposed for the SPN equations (referred hereafter as the GPU-based method). The accelerated framework is implemented by using a compute unified device architecture (CUDA)[17]. In this framework, the SPN equations are solved with the FEM whose matrices are stored in a compressed sparse row (CSR) format that makes good use of the computing potential of the GPU hardware. The solution of the SPN equations is converted into the problem of a sparse linear system, which is then solved by the parallel conjugate gradient (CG) algorithm implemented on a GPU kernel.

The accuracy of the GPU-based method was first evaluated by comparing it with the MC simulation. Then, the

speed-up ratio between the GPU-based method and the conventional CPU computation (CPU-based method) was evaluated. The influence of the mesh size and mesh structure on the acceleration performance was also investigated. Finally, the performance of a parallel CG solver was evaluated with different thread organizations in the kernel function.

Based on the RTE, the SPN equations and the boundary conditions can be detailed as follows[11]:

$$\begin{cases} -\nabla \cdot \zeta_{i,\nabla\varphi_i}\nabla\varphi_i + \sum_{j=1}^{(N+1)/2}\zeta_{i,\varphi_j}\varphi_j = \zeta_{i,S}S_i \\ \sum_{j=1}^{(N+1)/2}\zeta^b_{i,\nabla\varphi_j}\mathbf{v}\cdot\varphi_j = \sum_{j=1}^{(N+1)/2}\zeta^b_{i,\varphi_j}\varphi_j i \in [1,(N+1)/2] \end{cases}, \tag{1}$$

where $\mathbf{v}$ is the unit outer normal vector, S_i is the ith composite moment of the light source in the scattering regions, N is the order of the equation, φ_i is the ith composite moment of the radiance of the light source, $\zeta_{i,S}$ is the coefficient of the illumination source, $\zeta_{i,\nabla\varphi_i}$, and ζ_{i,φ_i}, $\zeta^b_{i,\nabla\varphi_j}$, and ζ^b_{i,φ_j} are the coefficients which can be calculated from Ref. [11]. The SPN equations are solved using the FEM[18]. Assume that the domain of object Ω is discretized as a tetrahedral grid δ. The composite moments $\varphi_i(r)$ and the original light source $S_i(r)$ at a discrete point k can be given by the interpolation of the nodal coefficients $\varphi_{i,k}$ and $S_{i,k}$ using the piecewise polynomial shape function $v_k(r)$:

$$\varphi_i(r) = \sum_{k=1}^{N_\delta}\varphi_{i,k}v_k(r), \tag{2}$$

$$S_i(r) = \sum_{k=1}^{N_\delta}S_{i,k}v_k(r), \tag{3}$$

where N_δ is the total number of nodes on the entire discretized domain δ. When using the FEM for the SPN equations, they can be reformulated into the matrix equation:

$$\begin{bmatrix} M_{1\varphi_1} & \cdots & M_{1\varphi_{(N+1)/2}} \\ \vdots & \vdots & \vdots \\ M_{(N+1)/2\varphi_1} & \cdots & M_{(N+1)/2\varphi_{(N+1)/2}} \end{bmatrix}\begin{bmatrix} \varphi_1 \\ \vdots \\ \varphi_{(N+1)/2} \end{bmatrix} = \begin{bmatrix} b_{1\varphi_1} & & \\ & \ddots & \\ & & b_{(N+1)/2\varphi_{(N+1)/2}} \end{bmatrix}\begin{bmatrix} S_1 \\ \vdots \\ S_{(N+1)/2} \end{bmatrix}, \tag{4}$$

$$b_{i\varphi_j} = \int_\delta \zeta_{i,S}v_k v_l \mathrm{d}r, \tag{6}$$

where $\zeta_{i,S}$ is the coefficient of the illumination source. v_k, v_l are the piecewise polynomial shape functions. $f_{\mathbf{v}\cdot\varphi_i}(\cdot)$ can be obtained by solving a set of first-order equations. Thus, the linear relationship that links the unknown source distribution S and the boundary measurements φ could be obtained.

Incorporating the boundary conditions, the exiting partial current J on the boundary can be calculated as

$$J = \sum_{j=1}^{(N+1)/2}(\beta^b_{j,\nabla\varphi_j}\mathbf{v}\cdot\nabla\varphi_j + \beta^b_{j,\varphi_j}\varphi_j) \quad j \in [1,(N+1)/2], \tag{7}$$

where $\beta^b_{j,\nabla\varphi_j}$ and β^b_{j,φ_j} are the boundary coefficients and can be calculated according to Ref. [11].

From the above derivation of the finite element solver for SPN equations, the main procedure includes assembling the system matrix and solving the linear equations, which can be parallelized with the GPU technique.

The flowchart of the proposed GPU-based accelerated SPN equations is shown in Fig. 1. The main concerns of the GPU-based method are the parallelization of assembling the system matrices M, B, and S, and solving the linear relationship of Eq. (4) with the parallel CG solver. In Fig. 1, the mesh data of objects and optical properties of tissues are prepared in the host memory on the CPU, and then copied to the device memory on the GPU. Because programs run on the GPU are highly influenced by the storage allocation and memory access, the mesh and optical properties data are loaded into the texture cache for frequent access. Then, the element of system matrices M, B, and S are assembled in each thread using CUDA. The size of the system matrices would become extremely large if the order of SPN equation is too high. Consequently, the storage of the system matrices in the normal format will take a large amount of memory. Thus, the CSR format is adopted for the storage of system matrices. There are also some other formats for matrix storage, such as the ELLPACK (ELL)[19] or Hybrid[20]. However, these formats usually require a large amount of memory for storage. After the system matrices are constructed, a GPU-based CG solver is applied to calculate the radiance of the light source φ. Finally, the exiting partial current J is calculated via the coping φ from the device to the host memory.

The major feature of the CG iterative solver is the fast calculation of the product and addition of vectors, which is heavily influenced by the memory access and data distribution on the GPU. The kernel function for calculating

$$M_{i\varphi_j} = \begin{cases} \int_\delta\{\zeta_{i,\nabla\varphi_i}\nabla v_k \cdot \nabla v_l + \zeta_{i,\varphi_i}v_k v_l\}\mathrm{d}r - \int_{\partial\delta}\zeta_{i,\nabla\varphi_i}f_{\mathbf{v}\cdot\varphi_i}(v_k)v_l\mathrm{d}r & \text{if } i=j \\ \int_\delta\zeta_{i,\nabla\varphi_i}v_k v_l\mathrm{d}r - \int_{\partial\delta}\zeta_{i,\nabla\varphi_i}f_{\mathbf{v}\cdot\varphi_i}(v_k)v_l\mathrm{d}r & \text{if } i\neq j \end{cases}, \tag{5}$$

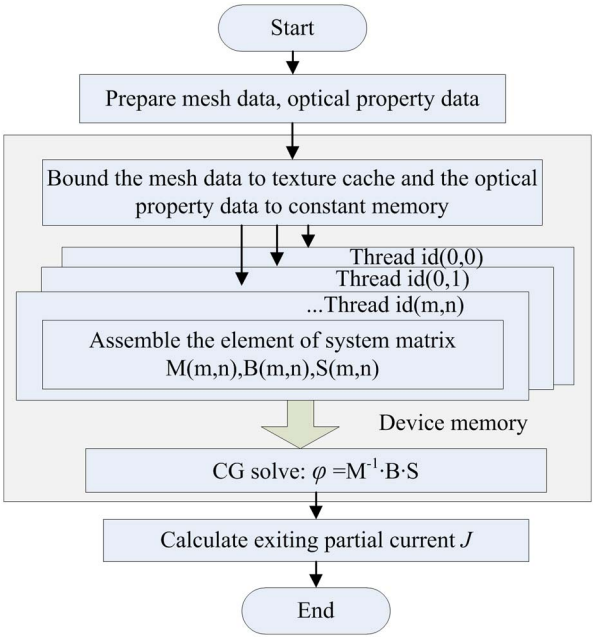

Fig. 1. Flowchart of the GPU-accelerated framework for the SPN equations.

the product of the sparse matrix and vector is optimized by using the shared memory of each block in this study. To accelerate the process, each row of the sparse matrix is set in a warp to calculate the product and addition[20]. Thus, the block size (blockSize) should be a multiple of the warp size (currently 32 of NVIDIA device). The kernel function of matrix-vector multiplication for the CSR format is shown in Fig. 2. When performing the multiplication with the elements of vector x, the elements are grouped by cooperating threads which access adjacent elements of the row. From here on, the number of cooperating threads assigned to each row is indicated by coopSize. Each group of cooperating threads is multiplied by the vector and added up into the shared memory. All of the elements on the shared memory are then summed up to obtain the multiplication result. For the additional step of shared memory, a conflict-free implementation of parallel reduction is adopted in our work to improve the performance.

```
__global__ void spmv_csr_vector_kernel ( const int num_rows ,
const int * ptr , const int * indices , const float * data ,
const float * x, float * y)
{
    __shared__ float vals [];
    int thread_id = blockDim.x * blockIdx.x
                  + threadIdx.x ; // global thread index
    int warp_id = thread_id / 32; // global warp index
    int lane = thread_id & (32 - 1); // thread index within the warp
    // one warp per row
    int row = warp_id ;
    if ( row < num_rows ){
    int row_start = ptr [row ];
    int row_end = ptr [ row +1];
    // compute running sum per thread
    vals [ threadIdx.x ] = 0;
    for ( int jj = row_start + lane ; jj < row_end ; jj += coopSize)
    vals [ threadIdx.x ] += data [jj] * x[ indices [jj ]];
    // parallel reduction in shared memory
    for(int ii = coopSize/2 ; ii >0 ;ii>>=1)
    vals [ threadIdx.x ] += vals [ threadIdx.x + ii];
    // first thread writes the result
    if ( lane == 0)
    y[ row ] += vals [ threadIdx.x ];}
}
```

Fig. 2. Kernel function of matrix-vector multiplication for the CSR sparse matrix format using 32 thread warp per matrix row.

We evaluated the performance of the proposed GPU-based method with the CPU-based method on a computer with an Intel Xenon 5440 processor of 2.4 GHz and an NVIDIA Tesla C2050 GPU. Both the CPU and GPU implementations made use of a double-precision floating-point format.

Firstly, the accuracy of the GPU-accelerated SPN equations was validated on two kinds of phantoms by comparing it with the MC simulation[18,21], which was considered as the gold standard for describing light propagation in tissues. For all of the simulations, the photon number of the light source was set to be 1×10^7 to get reliable simulation results. The average relative error (ARE) was used to estimate the discrepancy quantitatively between the calculated results d^i_{SPn} of the GPU-accelerated SPN equations and the simulated results d^i_{mc} of the MC method:

$$\mathrm{ARE} = \frac{1}{N_d}\sum_{i=1}^{N_d} \mathrm{abs}(d^i_{\mathrm{mc}} - d^i_{\mathrm{SPn}})/\max(d^i_{\mathrm{mc}}), \qquad (8)$$

where N_d is the dimension of the results.

As shown in Figs. 3(a) and 3(b), we selected a homogeneous cylindrical phantom and a digital mouse for validation. The cylindrical phantom had a radius of 5 mm and a height of 10 mm in which a spherical light source was located near the center of the phantom (0, 1.5, and 0 mm). The phantom was discretized into 56664 tetrahedral elements and 11161 nodes. The optical parameters of the phantom, described by the absorption coefficient (μ_a), scattering coefficient (μ_s), anisotropy coefficient (g), and refractive index (n) of the phantom, were 0.3 mm^{-1}, 5 mm^{-1}, 0.9, and 1.37, respectively. The comparison between the GPU-based method and the MC method was observed and the curve at the position of $z = 0$ mm was extracted, as shown in Fig. 3(c). The ARE between the transmittance results of the GPU-accelerated SPN equations and the MC method were 0.0539, 0.0114, 0.0113, and 0.0108 for $N = 1$, 3, 5, and 7, respectively.

The digital mouse model was used to demonstrate the capability of the proposed method in handling light propagation in the tissues with a complex structure. The organs included in the digital mouse were shown in Fig. 3(b), and the related optical properties at the wavelength of 650 nm were listed in Table 1. A spherical light source with a radius of 1.7 mm was located in the liver of the digital mouse. The transmittance results at a height of $z = 22$ mm were extracted for the comparison. The comparative result plotted versus the observed points was shown in Fig. 3(d).

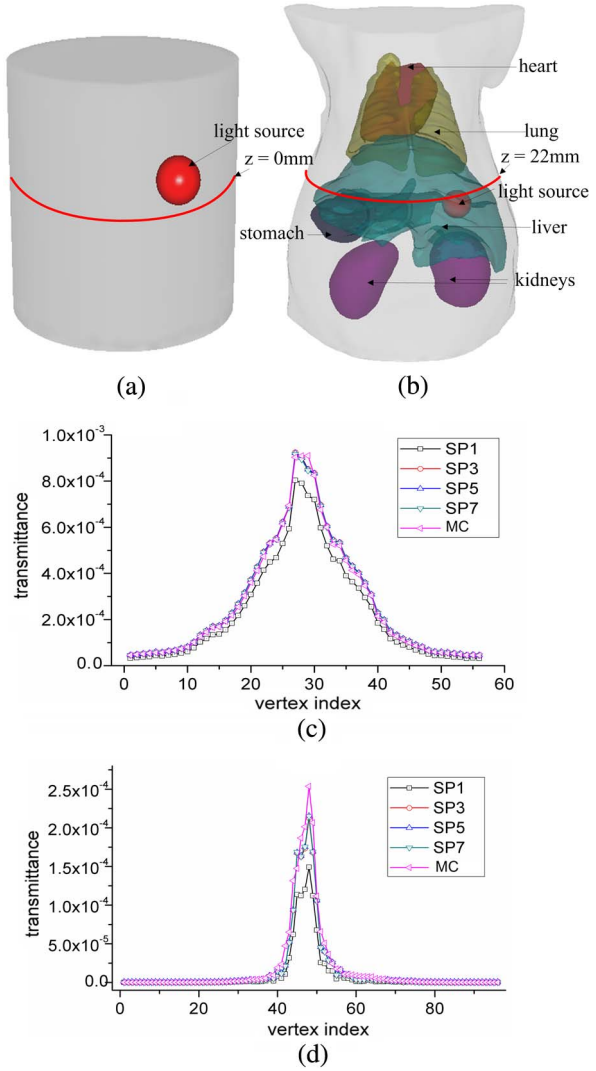

Fig. 3. Phantoms used in accuracy validation: (a) homogeneous cylindrical phantom. (b) Digital mouse model based phantom. (c) and (d) comparative results between the GPU-accelerated SPN equations ($N = 1$, 3, 5, and 7) and the MC method of the homogeneous cylindrical phantom and digital mouse, respectively.

Table 1. Optical Properties of Digital Mouse Phantom at the Wavelength of 650 nm, including Absorption Coefficient (μ_a), Scattering Coefficient (μ_s), Anisotropy Coefficient (g), and Refractive Index (n)

Tissue	μ_a (mm^{-1})	μ_s (mm^{-1})	g	n
Muscle	0.11636	4.6735	0.9	1.37
Liver	0.47078	6.999	0.9	1.37
Lung	0.26296	36.818	0.94	1.37
Heart	0.07859	6.7104	0.85	1.37
Kidney	0.08811	16.846	0.86	1.37
Stomach	0.01504	18.497	0.92	1.37

Secondly, the acceleration performance of the GPU-based method was investigated by comparing it with the CPU-based method. The phantom used in this investigation had the same size and optical properties as that used in the first phantom of accuracy validation. The influence of the size of system matrix on the acceleration performance was evaluated by discretizing the phantom into different numbers of tetrahedrons from 3421 to 94528. The ratio between the total time cost of the CPU-based SPN method and that of the GPU-based one was shown in Fig. 4(a). The speed-up ratio increased with the number of tetrahedral meshes, and a best speed-up ratio could be up to 25 in the observed cases.

The acceleration performance of the GPU-accelerated CG solver with a different coopSize and blockSize in kernel function was also investigated. A cylindrical phantom with the same size and optical properties as that used in the above validation was adopted in this investigation, which consisted of 79626 tetrahedrons. The speed-up ratio of the GPU-accelerated CG solver over the CPU one for solving the system matrix of SP7 equations was shown in Fig. 4(b), where the blockSize was from 32 to 256, and the blockSize was the integer times the warp size. The result showed that the best speed-up ratio (13.867) was obtained when the coopSize and blockSize were set to be 8 and 192, respectively. The speed-up ratio increased with the number of the blockSize when it was smaller than

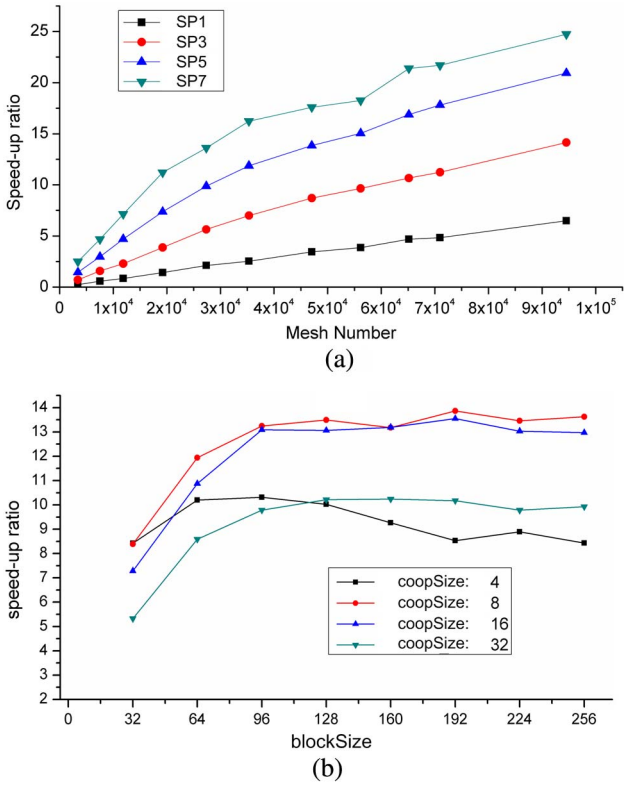

Fig. 4. (a) Speed-up ratio of the total processing time using the GPU-accelerated SPN method over the CPU-based one. (b) Speed-up ratio of the GPU-accelerated CG solver over the CPU one for solving the system matrix of SP7 equations.

96. However, the speed-up ratio remained steady when the blockSize was bigger than 96. The coopSize of 8 or 16 provided a better performance.

During the experiments, there were some other findings about the GPU-accelerated CG solver. Although the CG solver for solving the linear problem is efficient and stable in most cases, it could become divergent for the case of a large scale system matrix, especially for high order SPN equations. The acceleration is highly influenced by the filling fraction or the sparsity of the system matrix for different order SPN equations. We find that the non-zero elements of the system matrix for the SP1 equation gather closer to the diagonal line compared with the SP3, SP5, and SP7 equations. The non-zero elements of the system matrix for the SP7 equations have the most decentralized distribution, and the size of the sparse system matrix for SP7 is 15 times larger than that of SP1. As a result, the CG solver may be divergent for the large scale matrix and high order SPN equations.

The kernel function of the CG solver is highly influenced by coopSize. The coopSize of 8 or 16 provides a better performance in this study. This may be attributed to the sparsity of the system matrix. Each row of the system matrix is processed in a warp on the GPU. However, when the number of non-zero elements in each row is less than warp size (32) or even smaller, the threads will be idle. So, we defined the coopSize to avoid thread idling when the non-zero elements were less than the warp size.

In conclusion, a GPU-based acceleration framework for SPN equations is proposed to study the light propagation of 3D optical imaging. The accuracy validation experiments demonstrate that the proposed GPU-accelerated method has a good agreement with the MC simulation. Furthermore, the acceleration performance investigation experiments illustrate that the proposed GPU-accelerated method has an excellent acceleration performance over the CPU-based method, with a best speed-up ratio of 25 for the observed cases. The performance of the proposed GPU-accelerated method proved that it is a powerful tool for 3D optical imaging.

This work was partly supported by the National Natural Science Foundation of China (Nos. 81227901, 81571725, 61471279, and 61405149), the Natural Science Basic Research Plan in Shaanxi Province of China (Nos. 2015JZ019 and 2015JQ6249), and the Fundamental Research Funds for the Central Universities (Nos. NSIZ021402 and NSIY061406).

References

1. V. Ntziachristos, J. Ripoll, L. H. V. Wang, and R. Weissleder, Nat. Biotechnol. **23**, 313 (2005).
2. E. E. Graves, J. Ripoll, R. Weissleder, and V. Ntziachristos, Med. Phys. **30**, 901 (2003).
3. J. K. Willmann, N. van Bruggen, L. M. Dinkelborg, and S. S. Gambhir, Nat. Rev. Drug Discovery **7**, 591 (2008).
4. M. S. Patterson, B. C. Wilson, and D. R. Wyman, Lasers Med. Sci. **6**, 155 (1991).
5. Q. Fang, Biomed. Opt. Express **1**, 165 (2010).
6. M. Jia, S. Cui, X. Chen, M. Liu, X. Zhou, H. Zhao, and F. Gao, Chin. Opt. Lett. **12**, 031702 (2014).
7. H. Shen and G. Wang, Phys. Med. Biol. **55**, 947 (2010).
8. J. Zhang, J. Shi, S. Zuo, F. Liu, J. Bai, and J. Luo, Chin. Opt. Lett. **13**, 071002 (2015).
9. W. Li, H. Zhao, X. Qu, Y. Hou, X. Chen, D. Chen, X. He, Q. Zhang, and J. Liang, Chin. Opt. Lett. **10**, 021701 (2012).
10. Z. Yuan, Q. Z. Zhang, E. Sobel, and H. B. Jiang, J. Biomed. Opt. **14**, 054013 (2009).
11. A. D. Klose and E. W. Larsen, J. Comput. Phys. **220**, 441 (2006).
12. Y. J. Lu, B. H. Zhu, H. O. Shen, J. C. Rasmussen, G. Wang, and E. M. Sevick-Muraca, Proc. SPIE **7892**, 78920F (2011).
13. K. Liu, Y. J. Lu, J. Tian, C. H. Qin, X. Yang, S. P. Zhu, X. A. Yang, Q. S. Gao, and D. Han, Opt. Express **18**, 20988 (2010).
14. W. Li, H. J. Yi, Q. T. Zhang, D. F. Chen, and J. M. Liang, Comput. Math. Methods Med. **2012**, 394374 (2012).
15. N. Ren, J. Liang, X. Qu, J. Li, B. Lu, and J. Tian, Opt. Express **18**, 6811 (2010).
16. M. Schweiger, J. Biomed. Imaging **2011**, 10 (2011).
17. C. Nvidia, *Compute Unified Device Architecture Programming Guide* (2007).
18. Y. J. Lu, H. B. Machado, A. Douraghy, D. Stout, H. Herschman, and A. F. Chatziioannou, Opt. Express **17**, 16681 (2009).
19. C. J. Ribbens, G. G. Pitts, and L. T. Watson, Comput. Syst. Eng. **4**, 531 (1993).
20. N. Bell and M. Garland, "Efficient sparse matrix-vector multiplication on CUDA," NVIDIA Technical Report NVR-2008-004, NVIDIA Corporation (2008).
21. S. Ren, X. Chen, H. Wang, X. Qu, G. Wang, J. Liang, and J. Tian, Plos One **8**, e61304 (2013).