Accelerating frequency-domain diffuse optical tomographic image reconstruction using graphics processing units

Jaya Prakash
Venkittarayan Chandrasekharan
Vishwajith Upendra
Phaneendra K. Yalavarthy
Indian Institute of Science
Supercomputer Education and Research Centre
Bangalore, 560 012 India

Abstract. Diffuse optical tomographic image reconstruction uses advanced numerical models that are computationally costly to be implemented in the real time. The graphics processing units (GPUs) offer desktop massive parallelization that can accelerate these computations. An open-source GPU-accelerated linear algebra library package is used to compute the most intensive matrix-matrix calculations and matrix decompositions that are used in solving the system of linear equations. These open-source functions were integrated into the existing frequency-domain diffuse optical image reconstruction algorithms to evaluate the acceleration capability of the GPUs (NVIDIA Tesla C 1060) with increasing reconstruction problem sizes. These studies indicate that single precision computations are sufficient for diffuse optical tomographic image reconstruction. The acceleration per iteration can be up to 40, using GPUs compared to traditional CPUs in case of three-dimensional reconstruction, where the reconstruction problem is more underdetermined, making the GPUs more attractive in the clinical settings. The current limitation of these GPUs in the available onboard memory (4 GB) that restricts the reconstruction of a large set of optical parameters, more than 13,377.

Keywords: near-infrared diffuse optical tomography; image reconstruction; parallel processing; graphics processing units.

Paper 10276R received May 24, 2010; revised manuscript received Sep. 12, 2010; accepted for publication Sep. 15, 2010; published online Nov. 19, 2010.

1 Introduction

Diffuse optical imaging has received heightened attention in the last decade because of its capability to provide functional images of the tissue under investigation using nonionizing near-infrared light (600–1000 nm). Specifically, imaging of brain and breast has been the primary applications of diffuse optical tomography. The critical step in obtaining these images is estimating internal distribution of optical properties of the tissue using the measurements made on the tissue boundary.

Because the scattering is the dominant mechanism for near-infrared light (NIR) interaction with tissue, the estimation problem, also known as the inverse problem, is nonlinear, ill-posed, and some times underdetermined. Thus, solving an inverse problem necessitates the use of computationally intensive models. The computed data using these models are matched with the experimental data iteratively in the least-squares sense to obtain the optical properties of the tissue. The major challenge in terms of obtaining these optical images in real time is the computational cost associated with these advanced computational models, because these are used repeatedly. These iterative techniques used in the inverse problem relies on the calculation of modeled data and Jacobian (or its variant) at each iteration to obtain an update of optical properties. Depending on the model used, these calculations can span up to several hours, especially in three-dimensional (3-D) cases. There were attempts earlier to accelerate these calculations by using parallel computers, which have been shown to give a speedup of factor $n$, with $n$ being the number of parallel processors used. The limitation with the use of parallel computers is the cost associated with achieving higher speedups and also the complex approach in parallelizing these computational models. Here, we aim to take advantage of general purpose graphics processing units (GPUs) in massively parallelizing these calculations. This work specifically aims to present accelerating the frequency domain diffuse optical image reconstruction using a cost-effective (~$1200) programmable GPU (NVIDIA Tesla C 1060). This is achieved using an open-source GPU-accelerated linear algebra library (CULA) that utilizes the NVIDIA’s compute unified device architecture (CUDA).

Earlier works have shown the parallel computation capability of GPUs in performing high-speed Monte Carlo simulation of photon propagation in tissue and proven to give a speedup of ~100 for simple cases and ~10 in heterogeneous tissues when compared to the implementation on a modern central processing unit (CPU). Also, the standard filtered backprojection (Radon transform-based) algorithm used in computed tomography (CT) has been shown to give a speedups of 100 compared to standard CPUs. These kinds of massively parallel...
programmable GPUs have been used in the context of optical projection tomography and shown to give 300-fold acceleration compared to traditional CPUs.\textsuperscript{14} For iterative algorithms that are used in CT reconstruction, such as simultaneous algebraic reconstruction, the reported speedups are 12-fold per iteration, comparing GPU to CPU.\textsuperscript{15} The image reconstruction algorithms that use Fourier transform or its variant have been shown to give speedups on the order of 100 or more using GPUs, and the reconstruction algorithms that mainly involve matrix computations have been shown to give speedups on the order of 10 or more.\textsuperscript{14,15} The main aim of this work is to demonstrate the parallel computing power of these GPUs for performing diffuse optical tomographic image reconstruction.

Here, we used a finite element (FE) method (FEM)--based computational model that solves the frequency-domain diffusion equation (DE) requiring complex arithmetic. The number of unknowns in the inverse problem will be equal to 2N\(_x\) (N\(_x\) absorption coefficients and N\(_x\)-diffusion coefficients), with N\(_x\) being the number of finite element nodes used in the FE mesh to obtain the modeled data (described later). To obtain the modeled data, Jacobian, and update of optical properties at every iteration, a set of linear system of equations (order of 2N\(_x\)) needs to be solved along with the matrix-matrix multiplications (order of 2N\(_x\)), which will take typically O\((2N_x)^3\) operations. We will show that these operations could be done in parallel using a GPU (NVIDIA Tesla C 1060) to accelerate the diffuse optical tomographic image reconstruction procedure.

As diffuse optical image reconstruction relies on numerical calculations, specifically system solve (i.e., solving \(x = Ax = b\)) and matrix multiplications. Several packages,\textsuperscript{16–24} listed in Table 1, exist that provide options for carrying out the calculations on GPUs. The FEM-based forward models based on frequency-domain DE results in sparse symmetric linear system of equations (complex type). This, in turn, limits the use of sparse numerical computations on GPUs in these cases (more discussion to follow). The only package that is capable of dealing with sparse system solve for the real type is Cusp,\textsuperscript{19} allowing only matrix-vector computations. It will be shown that even with the use of the full (nonsparse) matrices, the GPUs are capable of giving an acceleration of up to 7 (for completing a start-to-end single iteration of the diffuse optical image reconstruction) compared to CPU sparse computations. Note that the Linux-based platform is used to carry out the computations performed in this work.

### 2 Methods

#### 2.1 Diffuse Optical Tomographic Image Reconstruction

Diffuse optical tomographic image reconstruction is typically performed using Newton-type algorithms, where the modeled data \([G(\mu)\text{ with } \mu \text{ representing the set of optical properties}]\) is matched to the experimental data \((\gamma)\), iteratively, in the least-square sense.\textsuperscript{4,7} The most popular technique among the full Newton-type algorithms is Levenberg–Marquardt minimization, described in detail in Ref.\textsuperscript{7}. The FEM-based frequency-domain diffusion model for the calculation of \(G(\mu)\) is described in Refs.\textsuperscript{25 and 26}; here, it is only briefly reviewed. The frequency-domain DE is given by\textsuperscript{25}

\[
-\nabla D(r)\nabla \Phi(r, \omega) + \left( \mu_a(r) + \frac{i\omega}{c} \right) \Phi(r, \omega) = q_o(r, \omega),
\]

where \(\Phi(r, \omega)\) photon density (complex values) at position \(r\) for the light modulation frequency of \(\omega = 2\pi f\) with \(f = 100 \text{ MHz}\). The light source, represented by \(q_o(r, \omega)\), is modeled as isotropic and \(c\) represents the speed of light in tissue. The absorption coefficient is represented by \(\mu_a(r)\) and the diffusion coefficient by \(D(r)\), defined as

\[
\begin{array}{c}
\mu_a(r) = \mu_a^\text{sc} + \mu_a^\text{linear} + \mu_a^\text{anisotropic}, \\
D(r) = D^\text{sc} + D^\text{linear} + D^\text{anisotropic}.
\end{array}
\]

### Table 1

Comparison of features of available CUDA-based numerical linear algebra packages along with their source.

<table>
<thead>
<tr>
<th>Package</th>
<th>Open Source</th>
<th>Complex Arithmetic</th>
<th>Precision</th>
<th>Sparse</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Single</td>
<td>Double</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Real</td>
<td>Complex</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Matrix Multiplication</td>
<td>System Solve\textsuperscript{a}</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Ref.</td>
</tr>
<tr>
<td>CULA Basic</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>CULA Premium/Commercial</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Magna</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Cusp</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CUBLAS</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GPU Lib</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>GPU mat</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>ViennaCL</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

\textsuperscript{a}Solves for \(x \text{ in } Ax = b\).

\textsuperscript{b}Only for real cases.

\textsuperscript{c}CULA is also part of Jacket.
The diffusion equation is given by

\[ D(r) = \frac{1}{3[\mu_\alpha(r) + \mu'_\alpha(r)]}, \tag{2} \]

with \( \mu'_\alpha(r) \) representing the reduced scattering coefficient. The \( \mu \) in this work represents \([D(r)]:\mu_\alpha(r)\). A Robin (type III) boundary condition is typically used to take care of the refractive-index mismatch at the tissue boundary.\(^{25}\)

In the FEM framework, the imaging domain is discretized into linear triangular elements (for two dimensions) or linear tetrahedral elements (for three dimensions) connected at \( N_v \) vertex nodes. Then the computational (forward) model for solving the diffusion equation reduces to\(^{25,26}\)

\[ K \Phi = q, \tag{3} \]

where \( K \) is known as the mass matrix with a dimension of \( N_N \times N_N \) (symmetric matrix) and is a function of \( \mu \) [i.e., \( \Phi = K^{-1}q = F(\mu) \)], with \( K \) assembled over all elements of the finite element mesh. \( q \) represents the forcing, including the source term \([q_0(r,o)]\) and the boundary condition.\(^{25}\) The modeled data \([G(\mu)]\) is obtained by sampling of \( \Phi \) at the measurement position [i.e., \( G(\mu) = S(\Phi) = S(F(\mu)) \), where \( S \) represents the sampling matrix (containing source/detector positions) and \( F \) is the forward model].\(^{27}\) Note that \( K \) is highly sparse (with a banded structure in case of bandwidth-optimized FE meshes) and typically sparse matrix solvers are used to obtain \( \Phi \). This process of solving for \( \Phi \) [Eq. (3)] involves a decomposition method (iterative procedure) preambled by a preconditioning step, because \( K \) is a large sparse complex matrix with a high numerical condition.

The most important steps pertaining to the image reconstruction procedure are given as a flowchart in Fig. 1. The iterative image reconstruction procedure starts with an initial guess for the optical properties (\( \mu_0 \)) typically obtained using the calibration procedure of experimental data \((y)\).\(^{31,32}\) Using this \( \mu_0 \), the forward model is solved to obtain \( G(\mu) \) and, more importantly, the Jacobian \((J = \delta G(\mu)/\delta \mu)\), which gives the rate of change in the modeled data with respect to optical properties. \( J \) is typically obtained using the adjoint formulation,\(^{25}\) and the most important computations pertained to the calculation of \( J \) are given in Fig. 2. The computation times for each step (in percent) in calculation of \( J \) with adjoint formulation is also given in Fig. 2. It could be seen that calculation of \( J \) needs both \( \Phi \) and \( \Phi^* \) (adjoint fluence, obtained by interchanging the source and detector positions) and typically takes \( \sim 35\% \) of time in the preconditioning [i.e., calculation of \( P^{-1}K \) with the order of calculations as \( O([2N_N]^3) \)], as both \( P^{-1} \) and \( K \) are complex [for the real case, it will be \( O(N_M N_N^2) \)].\(^{29}\) Obtaining \( \Phi \) in Eq. (4) is performed using LU decomposition, and the number of operations required are \( O(2 \times [2N_N]^3) \),\(^{29,30}\) contributing to \( \sim 25\% \) of total computation time (Fig. 2) taken for calculation of \( J \).

As the Rylov approximation is used in the work, the frequency domain data becomes, \( y = [\ln(A); \theta] \), where \( \ln(A) \) is the natural logarithm of amplitude \( (A) \) and \( \theta \) is the phase of the frequency domain signal, making the \( J \) a real valued matrix (dimension of \( 2N_M \times 2N_N \), where \( N_M \) is the number of measurements).\(^{4,7}\) The procedure involved in the calculation

---

**Fig. 1** Flowchart presenting the important steps involved in reconstructing optical properties in diffuse optical tomography along with associated computation time (in percent of total time per iteration). The steps that used GPU-based computations are indicated with a double arrow.

**Fig. 2** Flowchart showing the important computations for calculation of the Jacobian \( J \). The computation time in terms of percentage of total time taken in calculation of \( J \) at each step is also indicated. Note that the calculation of \( J \) accounts for 75% of the total time taken in completing one iteration of diffuse optical image reconstruction (also indicated in Fig. 1).
of real valued $J$ using complex $\Phi$ is given in Ref. 25. Note that the calculation of $J$ and $G(\mu)$ consumes $\sim 75\%$ of total computation time taken for completing a single iteration (see Fig. 1). To obtain an update of the optical properties ($\Delta \mu$), the Levenberg–Marquardt (LM) minimization is used and the objective function for this minimization scheme is given as

$$\Omega = \| y - G(\mu) \|^2.$$  

(6)

The objective of the LM minimization scheme is to match $y$ with $G(\mu)$ in the least-squares sense, by changing $\mu$. The details of LM minimization scheme are discussed in Ref. 7. The update equation (for getting $\Delta \mu$) for the LM minimization becomes

$$[ J^T J + \lambda J ] \Delta \mu = J^T [ y - G(\mu) ],$$  

(7)

where $J^T$ represents the transposed $J$ and $I$ is the identity matrix. $\lambda$ is the regularization parameter, chosen empirically (starts at 10 multiplied by the maximum of the diagonal values of $J^T J$ and reduced by 3.35 by a factor of $10^3$ at every subsequent iteration). Computing $\Delta \mu$ with the use of $J$ and $G(\mu)$ typically consumes $20\%$ of total time in any given iteration (see Fig. 1). LU factorization is used in solving Eq. (7) to obtain $\Delta \mu$, with the number of operations of order $O(2 \times (2N_i)^3/3)^{29,30}$ The procedure for calculation of $J$, $G(\mu)$, and subsequently $\Delta \mu$ is repeated until the relative difference in the objective function [Eq. (6)] does not improve by $>2\%$ (the same is indicated in Fig. 1). Because $95\%$ of total time per iteration is spent on calculation of $J$, $G(\mu)$, and $\Delta \mu$, the emphasis of this work is to accelerate these computations, which involve either matrix-matrix multiplications [calculation of $P^{-1} K$ in Eq. (4) and $J^T J$ in Eq. (7)] and solving the system of linear equations [Eqs. (4) and (7)], using GPUs.

### 2.2 GPU-Based Diffuse Optical Tomographic Image Reconstruction

The GPU typically consists of $100$–$200$ stream processors that employ single-instruction multiple thread execution to give massive parallel computation power. As this processing of large data sets is executed in parallel, the GPU accelerates the computation in cases such as large matrix computations, including multiplications and decompositions.

In this work, a NVIDIA GPU, which is CUDA enabled, is used in parallelizing these matrix computations. The specification of this GPU card along with the CPU that is compared against it is given in Table 2. As indicated earlier, the matrix computations are carried out using CULA. The NIRFAST package is used for the frequency-domain diffuse optical tomographic image reconstruction. Because NIRFAST is built on Matlab-based routines, the CULA functions (written in C, which use CUDA libraries) are wrapped to form Matlab executable (mex) files (details are given in Ref. 36). The routines that are used in this work along with examples of its usage are given in Table 3. Note that in this work only the basic package of CULA, which is open source, is used. This basic version of the CULA package is limited to single precision, which is sufficient for the diffuse optical tomographic image reconstruction (also shown later).

As indicated in Fig. 1, the developed GPU routines (given in Table 3) based on CULA are used at six instances in a single

### Table 2 Specifications of GPU and CPU used in this work along with its limitation on finite element mesh size (based on the available memory) in terms of number of nodes ($N_N$).

<table>
<thead>
<tr>
<th>Processing unit</th>
<th>Model No.</th>
<th>Cores</th>
<th>Clock Rate (GHz)</th>
<th>Cost (in U.S. $)</th>
<th>Memory (GB)</th>
<th>$N_N$ allowed</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPU</td>
<td>NVIDIA Tesla C1060</td>
<td>240</td>
<td>1.33</td>
<td>$\sim 1200$</td>
<td>4</td>
<td>13,377</td>
</tr>
<tr>
<td>CPU</td>
<td>Intel Xeon E 5410</td>
<td>8</td>
<td>2.33</td>
<td>$\sim 2400$</td>
<td>4</td>
<td>13,377</td>
</tr>
</tbody>
</table>

### Table 3 GPU-based Matlab executable (mex) CULA functions that are used in this work.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>culaGemm</td>
<td>Computes the multiplication of two complex matrices $[C = A^* B]$</td>
<td>$[C] = \text{culaGemm}(A,B)$; where matrices $A$ and $B$ are of type complex and single precision.</td>
</tr>
<tr>
<td>culaCsv</td>
<td>Computes the solution to a complex linear system of equations $[Ax = B]$ using LU decomposition</td>
<td>$[x] = \text{culaCsv}(A,B)$; where matrices $A$ and $B$ are of type complex and single precision.</td>
</tr>
<tr>
<td>culaSgemm</td>
<td>Computes the multiplication of two real matrices $[C = A^* B]$</td>
<td>$[C] = \text{culaSgemm}(A,B)$; where matrices $A$ and $B$ are of type real and single precision.</td>
</tr>
<tr>
<td>culaSv</td>
<td>Computes the solution to a real linear system of equations $[Ax = B]$ using LU decomposition</td>
<td>$[x] = \text{culaSv}(A,B)$; where matrices $A$ and $B$ are of type real and single precision.</td>
</tr>
</tbody>
</table>
The difference in the L2-norm of the estimated data-model misfit \( \delta \) and estimate optical properties \( \mu \) with respect to iteration number between single- and double-precision computations using CPU are plotted in Fig. 3 for the test object described in Sec. 2.3. It is evident from Fig. 3 that the maximum difference in terms of percentage in either \( \delta \) or estimated \( \mu \) is less than 0.005%, asserting the fact that single precision is sufficient for carrying out the diffuse optical tomographic image reconstruction computations. Because Rytov approximated data are used in this work [i.e., using \( \ln(A) \) rather than \( A \)] the single-precision calculations were adequate. The main advantages of single- over double-precision computations are twofold. First, given the limitation on the available GPU/CPU memory, diffuse optical tomographic image reconstruction could be performed on meshes twice as large. Second, the basic version of CUDA (open source), which has only single-precision capability, could be used to perform the GPU computations.

The optical images obtained with integration of CUDA-based functions (Table 3 with the instances given in Sec. 2.2)
into the diffuse optical image reconstruction procedure using the test case described earlier were identical (visually and numerically) compared to images obtained using single-precision CPU-based implementation. The observed difference (in percent) in the $L_2$-norms of the data-model misfit ($\delta$) and estimate optical properties ($\mu$) between the GPU implementation (with CULA basic) and traditional CPU implementation (with double precision) are same as the observed values plotted in Fig. 3.

A 2-D test case reconstruction results along with the target distribution using the three strategies (CPU-Full, CPU-Sparse, and GPU-Full, discussed in Sec. 2.3) are given in Fig. 4. The number of iterations to converge to a solution for each strategy is equal to 10. Obtained $\mu_a$ and $\mu_s'$ distributions were correlated to find a similarity measure among the three strategies, resulting in a correlation coefficient of 1 (with in the limits of single precision), asserting that the reconstructed results are identical.

The time taken per iteration [start to end, including data transfer between CPU (host) and GPU (device) and vice versa] on GPU (NVIDIA Tesla C 1060) and CPU (Intel Xeon E 5410) for different mesh sizes both in two and three dimensions were plotted in Fig. 5 for the three strategies (GPU-Full, CPU-Full, and CPU-Sparse, discussed in Section 2.3). In the 2-D case, it is clear from Fig. 5(a) that the computation time per iteration increases with increasing $N_N$ for all three strategies, with CPU-Full taking the maximum time. Among the three strategies, GPU-Full took the shortest computation time. As stated earlier, because sparse complex arithmetic is not supported in NVIDIA CUDA, the GPU-Sparse (GPU-based computation using sparse representation of matrices for instances 1–4, described Section 2.2) strategy was not attempted. Comparing the three strategies in this 2-D case, the speedup (acceleration) with the use of GPU was up to 7.5 comparing CPU-Full to GPU-Full. The maximum speedup has been lowered to 2 in comparing CPU-sparse to GPU-Full. For the 3-D case [Fig. 5(b)], a similar trend in the 2-D case was observed with the computation time per iteration being higher than 2-D case for the same $N_N$. In the 3-D case, the speedup (acceleration) with the use of GPU was up to 40 in comparing CPU-Full to GPU-Full, and was lowered to 7 in comparing CPU-sparse to GPU-Full.

The main emphasis of this work is to prove that the GPU computing offers a considerable speedup compared to CPU computing and the comparison has been only performed by the time taken per iteration. The diffuse optical image reconstruction typically requires about 10 iterations, and for a typical three-dimensional problem (Table 4), the least total time taken by the CPU (CPU-sparse) to converge to a solution is 290 min, and 48 min for GPU. The gain in terms of total computation time is significant in nature (comparing $\sim$5 h to 0.75 h), making the GPU computing very attractive to be used in the real time, especially in multiwavelength cases (the problem gets scaled by the

![Fig. 3](image-url) Difference (in percent) in the $L_2$ norm between the single- and double-precision computation over the iterations in (a) data-model misfit ($\delta$) and (b) estimated optical properties ($\mu$).

![Fig. 4](image-url) Two-dimensional reconstruction results obtained with 1% noisy data using three strategies—CPU-Full, CPU-Sparse, and GPU-Full—discussed in this work. The target 2-D distribution is given in column 1 for (a) $\mu_a$ images and (b) $\mu_s'$ images. The number of nodes along with computational time taken for each strategy in a given iteration is given in Table 4.
number of wavelengths), where the aim is to obtain functional images.

Even though it is not possible to attempt sparse complex arithmetic on NVIDIA CUDA-enable GPUs, we have attempted to test the capabilities of parallelizing sparse real computations (with the aid of Cusp, refer to Table 1) using a steady-state diffusion equation (where \( K \) and \( \Phi \) are real) to estimate \( \mu_\text{r} \) using intensity-based measurements in both 2-D and 3-D cases. Even though the GPU is able to provide speedup (up to 2) for calculation of \( \Phi \) or \( \Phi^* \) in solving sparse system of linear equations, the overall computation time taken per iteration from start to end (including the overhead of transferring sparse matrices from the CPU memory to GPU memory) is increased (at least by 70%) in comparison of GPU to CPU in these sparse cases (similar to observed trend in Refs. 40 and 41). The sparse cases have large overhead for GPU calculations in comparison to the CPU ones because the GPU compute kernels are not fully optimized. This overhead can become negligible when the problem size (number of nonzero entries) becomes bigger (typically >40,000). For fully bandwidth-optimized FEM meshes, the number of nonzero entries for the GPU-limited mesh (with \( N_N = 13,377 \)) is ~30,000, making the GPU-sparse calculations not attractive compared to CPU-sparse calculations. Thus, only CPU-sparse implementation is attempted in this work. Also, note that sparse computations on currently available GPUs were mainly centered around matrix-vector computations (encountered in iterative solvers to deduce \( x \) in \( Ax = b \)), which are not as straightforward to implement and integrate into the existing domain-specific applications, such as optical image reconstruction. 42, 43  

Fortunately, the next generation of GPUs will be able to perform sparse computations with similar ease as current real computations and support sparse complex computations (including matrix-matrix computations). Although it is not attempted for the fair comparison of the CPU and GPU computations, the optimal computations will utilize both CPU (for sparse matrix computations) and GPU (for full matrix computations).

Table 4 gives the computational time taken in a given iteration for the mesh of similar size (\( N_N \approx 9200 \)) both in 2-D and 3-D cases. Although the GPU calculations took similar time between 2-D and 3-D calculations, there is a factor-of-20 difference between the 2-D and 3-D calculations with CPU-Full strategy, especially in solving for \( \Phi \) in Eq. (4). This is primarily due to the higher numerical condition number of 3-D mass matrix \( [K] \) in Eq. (3), which takes more iterations to converge to a solution \( [\Phi] \) in Eq. (4)] compared to their counterparts in 2-D cases for the same size \( K \). 44 This higher condition number in the 3-D case could result from the nonuniform nature of 3-D mesh. The tetrahedral elements are formed using 3-D delaunay, which makes a uniform mesh more complex to generate, 45 as compared to the 2-D mesh. Also, the condition number of the mass matrix \( (K) \) depends on the number of source fibers and their placement. Within this work, where a ring type of data acquisition is performed, the regions far away from the fibers have negligible contribution as opposed to regions close to source/detector fibers. Therefore, the 3-D problem has a higher condition number, requiring more iterations. Hence, this leads to longer computation time, as compared to the 2-D problem, for decomposing \( P^{-1}K \) and then solving for \( \Phi \) in Eq. (4). Because the decomposition primarily involves matrix-vector computations, with GPU offering massive parallelization, results in similar computation time for the same size matrices both in the 2-D and 3-D cases (Table 4) with GPU giving peak performance in the 3-D case. Note that with \( J \) being a full matrix, computations for the last two instances (5 and 6) in Table 4 were not applicable to use sparse computations. These computations were carried out using CPU-Full.
Because modern day CPUs have multiple cores, it is possible to take advantage of the parallel computing power of these multicore CPUs. Recent work by Borsic et al. for a similar problem (electrical impedance tomography) like diffuse optical image reconstruction, the achievable speedup using a dual quad-core Intel Xeon processor was 7.6. But this required rewriting/optimizing the routines used in the computing of Jacobian and forward problem, where as the GPU calculations performed in this work used already preexisting routines. The speedup obtained using a multicore processor with out the optimization is only 4.6. Also, the typical cost of dual quad-core processor is at least twice expensive compared to the top of the line GPU used in this work (Table 2).

It is important to note that traditional parallel computing uses large number of processors connected in parallel to form a cluster, with an overhead of cost and demanding large footprint in the lab settings. These bulky parallel computer systems are not very attractive in the clinical settings. The GPU boards are comparatively low cost and fit into a traditional desktop computer, resulting in an desktop parallel computer (with a very small footprint), could become very attractive in the clinic to perform image reconstruction/analysis tasks. Usage of these GPU cards to accelerate FE meshing using multimodal imaging data that could be used in the NIR imaging studies is currently being explored. The CULA-based mex programs used in this work along with necessary documentation for installation and compiling are provided as open source.

### Table 4

<table>
<thead>
<tr>
<th>Instance</th>
<th>No.</th>
<th>Strategy</th>
<th>2-D</th>
<th>3-D</th>
</tr>
</thead>
<tbody>
<tr>
<td>$N_N$</td>
<td>–</td>
<td>CPU-Full</td>
<td>9223</td>
<td>9211</td>
</tr>
<tr>
<td>$P^{-1}K$</td>
<td>1</td>
<td>CPU-Full</td>
<td>376.9</td>
<td>373.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPU-Sparse</td>
<td>0.0015</td>
<td>0.0035</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPU-Full</td>
<td>36.8</td>
<td>37.1</td>
</tr>
<tr>
<td>$\Phi$ in Eq. (4)</td>
<td>2</td>
<td>CPU-Full</td>
<td>161.4</td>
<td>3307.3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPU-Sparse</td>
<td>7.7</td>
<td>669.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPU-Full</td>
<td>14.8</td>
<td>18.7</td>
</tr>
<tr>
<td>$P^{-1}K^*$</td>
<td>3</td>
<td>CPU-Full</td>
<td>377.3</td>
<td>374.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPU-Sparse</td>
<td>0.0016</td>
<td>0.0032</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPU-Full</td>
<td>37.1</td>
<td>38.2</td>
</tr>
<tr>
<td>$\Phi^* = P^{-1}q^*$</td>
<td>4</td>
<td>CPU-Full</td>
<td>160.1</td>
<td>3305.8</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPU-Sparse</td>
<td>8.1</td>
<td>671.4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPU-Full</td>
<td>14.2</td>
<td>18.9</td>
</tr>
<tr>
<td>$fJ$</td>
<td>5</td>
<td>CPU-Full</td>
<td>20.37</td>
<td>20.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPU-Sparse</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPU-Full</td>
<td>7.33</td>
<td>6.0</td>
</tr>
<tr>
<td>$\Delta\mu$ in Eq. (7)</td>
<td>6</td>
<td>CPU-Full</td>
<td>282.1</td>
<td>282.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPU-Sparse</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPU-Full</td>
<td>31.5</td>
<td>29.2</td>
</tr>
<tr>
<td>Total time</td>
<td>–</td>
<td>CPU-Full</td>
<td>1389.2</td>
<td>7768.9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPU-Sparse</td>
<td>381.6</td>
<td>1737.1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GPU-Full</td>
<td>183.5</td>
<td>286.2</td>
</tr>
</tbody>
</table>

Because general purpose computing using GPUs is becoming attractive in many areas of medical image reconstruction/processing, this work demonstrated that with the use of GPUs the speedups for a single iteration (start to end) of diffuse optical tomographic image reconstruction could be up to 40 compared to traditional CPU calculations. Also, the GPU computations are carried out using an open-source package (CULA) and appropriate code wrappers were used to integrate these routines into the existing diffuse optical image reconstruction package (NIRFAST). The current limitation to use GPU on a daily basis for diffuse optical tomographic image reconstruction is the available onboard memory (currently it is limited to 4 GB), which puts a restriction on mesh size that could be used. Moreover, the sparse complex calculations are not supported in GPU computing, making it less attractive for use in FE-based models. Because there is tremendous growth in the capabilities of GPU computing, this limitation might vanish, making GPU computing capabilities similar to those of CPU in applications that require more memory and sparse computations. More importantly, along with the cost effectiveness, these GPU cards are integrated with traditional desktop computers, giving massive parallel computing power at the desktop. Specific to diffuse optical tomography, GPU computing holds the promise of making the image reconstruction procedure match the time line, with the data acquisition, leading to dynamic NIR imaging viable in the clinic.

4 Conclusions

Because general purpose computing using GPUs is becoming attractive in many areas of medical image reconstruction/processing, this work demonstrated that with the use of GPUs the speedups for a single iteration (start to end) of diffuse optical tomographic image reconstruction could be up to 40 compared to traditional CPU calculations. Also, the GPU computations are carried out using an open-source package (CULA) and appropriate code wrappers were used to integrate these routines into the existing diffuse optical image reconstruction package (NIRFAST). The current limitation to use GPU on a daily basis for diffuse optical tomographic image reconstruction is the available onboard memory (currently it is limited to 4 GB), which puts a restriction on mesh size that could be used. Moreover, the sparse complex calculations are not supported in GPU computing, making it less attractive for use in FE-based models. Because there is tremendous growth in the capabilities of GPU computing, this limitation might vanish, making GPU computing capabilities similar to those of CPU in applications that require more memory and sparse computations. More importantly, along with the cost effectiveness, these GPU cards are integrated with traditional desktop computers, giving massive parallel computing power at the desktop. Specific to diffuse optical tomography, GPU computing holds the promise of making the image reconstruction procedure match the time line, with the data acquisition, leading to dynamic NIR imaging viable in the clinic.
Acknowledgments

P. K. Y. acknowledges the IISc start-up grant [Grant No. IISc-Part2(2A) Eleventh plan, 23/SERC] and the Apple Laureate award. The authors are thankful to the chairman of the Supercomputer Education and Research Centre, Indian Institute of Science, for providing the GPU cards (NVIDIA Tesla C 1060) for carrying out this work.

References