Medical Physics
The International Journal of Medical Physics Research and Practice

A nonvoxel-based dose convolution/superposition algorithm optimized for scalable GPU architectures

J. Neylon, K. Sheng, V. Yu, Q. Chen, D. A. Low, P. Kupelian, and A. Santhanam

Citation: Medical Physics **41**, 101711 (2014); doi: 10.1118/1.4895822
View online: http://dx.doi.org/10.1118/1.4895822
View Table of Contents: http://scitation.aip.org/content/aapm/journal/medphys/41/10?ver=pdfcov
Published by the American Association of Physicists in Medicine

A nonvoxel-based dose convolution/superposition algorithm optimized for scalable GPU architectures

J. Neylon,[a] K. Sheng, and V. Yu
Department of Radiation Oncology, University of California Los Angeles, 200 Medical Plaza, #B265, Los Angeles, California 90095

Q. Chen
Department of Radiation Oncology, University of Virginia, 1300 Jefferson Park Avenue, Charlottesville, California 22908

D. A. Low, P. Kupelian, and A. Santhanam
Department of Radiation Oncology, University of California Los Angeles, 200 Medical Plaza, #B265, Los Angeles, California 90095

(Received 24 February 2014; revised 9 July 2014; accepted for publication 27 August 2014; published 1 October 2014)

Purpose: Real-time adaptive planning and treatment has been infeasible due in part to its high computational complexity. There have been many recent efforts to utilize graphics processing units (GPUs) to accelerate the computational performance and dose accuracy in radiation therapy. Data structure and memory access patterns are the key GPU factors that determine the computational performance and accuracy. In this paper, the authors present a nonvoxel-based (NVB) approach to maximize computational and memory access efficiency and throughput on the GPU.

Methods: The proposed algorithm employs a ray-tracing mechanism to restructure the 3D data sets computed from the CT anatomy into a nonvoxel-based framework. In a process that takes only a few milliseconds of computing time, the algorithm restructured the data sets by ray-tracing through precalculated CT volumes to realign the coordinate system along the convolution direction, as defined by zenithal and azimuthal angles. During the ray-tracing step, the data were resampled according to radial sampling and parallel ray-spacing parameters making the algorithm independent of the original CT resolution. The nonvoxel-based algorithm presented in this paper also demonstrated a trade-off in computational performance and dose accuracy for different coordinate system configurations. In order to find the best balance between the computed speedup and the accuracy, the authors employed an exhaustive parameter search on all sampling parameters that defined the coordinate system configuration: zenithal, azimuthal, and radial sampling of the convolution algorithm, as well as the parallel ray spacing during ray tracing. The angular sampling parameters were varied between 4 and 48 discrete angles, while both radial sampling and parallel ray spacing were varied from 0.5 to 10 mm. The gamma distribution analysis method (γ) was used to compare the dose distributions using 2% and 2 mm dose difference and distance-to-agreement criteria, respectively. Accuracy was investigated using three distinct phantoms with varied geometries and heterogeneities and on a series of 14 segmented lung CT data sets. Performance gains were calculated using three 256 mm cube homogenous water phantoms, with isotropic voxel dimensions of 1, 2, and 4 mm.

Results: The nonvoxel-based GPU algorithm was independent of the data size and provided significant computational gains over the CPU algorithm for large CT data sizes. The parameter search analysis also showed that the ray combination of 8 zenithal and 8 azimuthal angles along with 1 mm radial sampling and 2 mm parallel ray spacing maintained dose accuracy with greater than 99% of voxels passing the γ test. Combining the acceleration obtained from GPU parallelization with the sampling optimization, the authors achieved a total performance improvement factor of >175 000 when compared to our voxel-based ground truth CPU benchmark and a factor of 20 compared with a voxel-based GPU dose convolution method.

Conclusions: The nonvoxel-based convolution method yielded substantial performance improvements over a generic GPU implementation, while maintaining accuracy as compared to a CPU computed ground truth dose distribution. Such an algorithm can be a key contribution toward developing tools for adaptive radiation therapy systems. [http://dx.doi.org/10.1118/1.4895822]

Key words: GPU, dose calculation, convolution, superposition, nonvoxel-based

1. INTRODUCTION

Radiotherapy has seen a major push toward treatment plans that are tailored to the patient and adapted to their radiation response.[1–4] Ignoring inter and intratreatment changes in tumor size and position can lead to target under-dosing and excessive exposure of healthy tissue.[3,5] Real-time adaptive therapy has been infeasible due in part to the time and computational effort required for such tasks.[6]

In recent years, graphics processing units (GPUs) have gained widespread use in scientific computing, due to its massive parallelization, allowing thousands of times more floating point operations per second than a typical CPU.[7,8] There are several hurdles along the path of a GPU implementation, but their acceleration capabilities have made radiation oncology challenges such as live tumor tracking and real-time dose estimations into realistic possibilities.[9]

Advantages of using GPUs for dose calculations have been previously examined, specifically in regard to the convolution/superposition algorithm. Three independent groups have implemented the superposition/convolution onto GPU architecture. Hissoiny *et al.* reported acceleration of 10–20× in 2009 and later improved to nearly 30× when compared to an optimized commercial CPU implementation.[10,11]

In 2011, GPU acceleration was pushed above 100× compared to an optimized dual core CPU.[12,13] Dose calculation accuracy of GPU and CPU implementations was compared by using 48 zenithal angles and 96 azimuthal angles. The accuracy, calculated as the percent dose difference between corresponding voxels relative to the maximum dose, agreed to within 2%–5%.

While these methods employed voxel-based calculations, Chen *et al.*[14,15] employed a nonvoxel-based (NVB) broad beam framework first proposed by Lu[16] to perform the calculations prior to convolution but did not extend it to the actual convolution. Acceleration factors of 1000–3000 were reported using their exponential kernel on GPU compared to a tabulated kernel on CPU.

As the computational capabilities of GPUs continue to improve, the performance bottlenecks have shifted from hardware considerations, such as data transfer and maximum number of parallel threads, to software and code design considerations. The GPU architecture has a unique memory hierarchy with variable data retrieval speeds and scopes.[7,8] In addition, the pattern in which the memory is accessed on the GPU also forms an important design consideration. To fully utilize the potential computing power of the GPU, the memory design aspects must be considered, requiring approaching old problems from new viewpoints. Convolution/superposition still provides the best compromise between speed and accuracy when performing dose calculations in heterogeneous materials. However, because of its inherent memory access pattern, the convolution process is performance-limiting when trying to port the algorithm to the GPU, specifically the spherical sampling pattern about the point of interest.

In this paper, a GPU-accelerated superposition/convolution is presented that employs an improved memory assignment optimization. Specifically, a NVB GPU-accelerated superposition/convolution algorithm and its dependence on sampling parameters are presented. Converting the dose convolution calculation to new coordinate systems aligned along each convolution direction allows for fully optimized memory access patterns along each step of the algorithm and provides a significant computational speedup. We also introduce a fourth sampling parameter, the spacing between parallel rays when resampling for the NVB coordinate system, alongside the traditional spherical sampling variables of the convolution algorithm. Utilization of greater sampling rates prolongs computational times and so was previously avoided for CPU based dose calculation frameworks. Characterizing the accuracy and performance effects of varying coordinate system parameters allows greater control over the convolution, further optimizing the algorithm.

2. MATERIALS AND METHODS

In this section, we first describe the collapsed cone convolution (Sec. 2.A). It is followed by a discussion on the convolution sampling, how it dictates the memory access patterns, and the resultant performance considerations (Sec. 2.B). We then present the nonvoxel-based algorithm (Sec. 2.C), detail the experiments to quantify the nonvoxel-based algorithm's accuracy (Sec. 2.D), and compare its performance to a voxel-based CPU algorithm and a generic GPU implementation (Sec. 2.E).

2.A. Collapsed cone convolution/superposition algorithm

Collapsed cone convolution/superposition (CCCS) has been well documented.[17–19] In this section, we present a brief review of its mathematics.

2.A.1. TERMA calculation

In order to calculate the total energy released in matter, or TERMA, the equivalent depth in water must be known for each voxel in the target. For calculation purposes, the beam was assumed to be originating from a point source 1 m above the isocenter. Siddon's ray-tracing algorithm was ported to GPU architecture for this task.[19] In order to compute the primary energy deposition, the attenuation path of each ray was corrected for density heterogeneities. This effective radiological path length in water was calculated from source to voxel by summing the contributions of each voxel along the ray path

$$d_i = \sum_j l_j \rho_j, \tag{1}$$

where i was the point of interaction, j was the voxel the ray intersected, l_j was the intersection length of the ray and the voxel, ρ_j was the voxel density relative to water and therefore unitless.

Equations (2) and (3) show the discrete formulas for the TERMA with a beam hardening correction summed over the

discretized energy spectrum, E,

$$T(i) = \sum_E \Psi_E \mu_E e^{-\mu_E d_i}, \tag{2}$$

$$T'(i) = \left(\frac{\sum_E \Psi_E \mu_{\text{en},E} e^{-\mu_E d_i}/T(i)}{\sum_E \Psi_E \mu_{\text{en},E} e^{-\mu_E d_0}/T(0)} \right) * T(i), \tag{3}$$

where i was the point of interaction, Ψ was the energy fluence, μ was the mass attenuation coefficient, and μ_{en} was the mass energy absorption coefficient. Equation (3) shows the correction factor for beam hardening using the unattenuated values.[20] The attenuation coefficients were drawn from the National Institute of Standards and Technology database.[21]

2.A.2. Cumulative-cumulative kernel generation

The CCCS dose distribution was calculated by convolving a polyenergetic cumulative-cumulative dose deposition kernel (CCK) with the TERMA volume computed using Eqs. (2) and (3).[17,18] The kernel files were precomputed, Monte Carlo generated, monoenergetic differential deposition distribution kernel (DK) about a point interaction. For each geometric location, the kernel files described the energy dispersal due to the type of interaction (T): primary interaction, first scatter, second scatter, multiple scatter, and bremsstrahlung/annihilation.[22] To create monoenergetic cumulative kernels (CK), the initial differential kernels were summed over the interaction type and integrated over the spherical sampling space.[23] The cumulative kernels were then integrated over the sampling space again and then summed over the energy spectrum (E) according to their spectrum weight (w_E), constructing a single polyenergetic CCK Refs. 23 and 24

$$\text{CK}(\theta,\varphi,r) = \int \left(\sum_T \text{DK}_T(\theta,\varphi,r) \right) dr, \tag{4}$$

$$\text{CCK}(\theta,\varphi,r) = \int w_E \left(\int \text{CK}(\theta,\varphi,r)\, dr \right) dE. \tag{5}$$

2.A.3. CCK dose convolution

The superposition method was employed to scale the kernel

$$\text{Dose}(v) = \int T'(v') \text{CCK}(\bar{\rho}_{v-v'} * v - v') dv',$$
$$\text{where} \int dv' = \iiint d\varphi d\theta dr, \tag{6}$$

where v was the interaction point, v' was the voxel being sampled, $\bar{\rho}_{v-v'}$ was the heterogeneity correction applied to the kernel, r was the radial component, θ was the zenith angle, and φ was the azimuthal angle. The CPU algorithm tackled this process using nested loops which cycled through each voxel, v, within the beam and then sampled the surrounding volume ($\iiint v' dV$), before moving on to the next voxel.

2.B. Convolution sampling and memory access patterns

The discretized convolution algorithm employed during the CCCS calculations [Eq. (6)] required spherical sampling about the voxel of interest and summing the dose contributions of the surrounding volume. In practice, the dose at the point of interaction was calculated by summing the contributions of the discretely sampled surrounding volume according to these three parameters: the number of zenithal angles (Θ), the number of azimuthal angles (Φ), and the size of the radial increment (P). The number of sampling points and the computation time were linearly related to Θ and Φ and inversely related to P. The limit of sampling resolution was set by the kernel file parameters. The dose deposition kernels were segmented into 24 concentric circles with varying radii from 0.1 to 60 cm and 48 zenithal segmentations equally spaced from 0° to 180°. This effectively created a ceiling to the zenithal and radial sampling during convolution. Azimuthal sampling was limitless in theory because the CCK was computed for a homogenous material. This resulted in symmetric dose deposition about the azimuth, and therefore the information was only recorded for two dimensions. However, when applying the heterogeneity correction, azimuthal sampling could have a profound effect on computation accuracy.

2.B.1. Generic GPU implementation

A generic method to parallelize the algorithm was developed initially similar to the first published GPU implementation of the convolution/superposition algorithm.[10,12] This simplistic approach launched a GPU function for each zenithal and azimuthal angle combinations, unrolled the outermost loops which cycle through each voxel, and convolved them simultaneously. Each voxel within the volume was assigned a thread and traced a ray from that voxel in the direction specified by the zenithal angle, Θ, and the azimuthal angle, Φ, sampling the density and TERMA at radial intervals of P and applying the CCK, scaled by the density for heterogeneity. For the ground truth sampling parameters of 48/48, this amounted to 2304 function launches.

2.B.2. Performance considerations and bottlenecks

The conventional GPU algorithm presented several hurdles when attempting to optimize memory access patterns for GPU architecture. The GPU contained several memory types with varying scopes and access speeds. Global memory had the largest capacity but also had the greatest latency when accessing data, typically between 400–600 clock cycles. Shared memory offered access speeds 100–150 times faster than global memory, but had scope limited to a single block of threads, and a much reduced capacity.[8] Global access speeds could approach shared access speeds if the memory fetches were coalesced. This facilitated a group of adjacent threads to simultaneously read from a group of adjacent memory addresses in the global memory space. The compiler will then

combine these into a single larger memory fetch, greatly reducing the latency.[7] Another design limitation is that the GPU's shared memory cannot be written into directly by the CPU. The threads of the block must read in the data from global memory first and fill the shared memory space. Therefore, the most efficient way to attack the convolution is to organize the threads along the convolution ray direction, utilize the coalesced global memory fetches to write into shared memory, and then use shared memory to perform the convolution.

The problem here is that the coalesced access is only possible in one direction while the convolution rays can have any arbitrary direction as defined by Θ and Φ. Texture memory is located in the global memory space, but is cached for locality and also provides an intrinsic linear interpolation in three dimensions. This makes it ideal when coalesced accesses are not possible, but the memory reads are patterned predictably. However, it is read-only unless it is created using a specialized array that can be bound to a surface object. This feature is only available on more recent generations of devices with compute capabilities of 3.0 or higher.

2.C. Nonvoxel-based algorithm

In this section, we present the framework of our nonvoxel-based algorithm. To take advantage of the different memory spaces and maximize efficiency, we split the convolution into four components: ray tracing, transposition, line convolution, and summation. These four steps were performed for every zenithal direction less than or equal to 90° and every azimuthal direction. Figure 1 illustrates the movement of data between GPU memory spaces during the process.

2.C.1. Ray tracing

We first converted the density and TERMA data volumes from voxelized Cartesian coordinates into a nonvoxel-based coordinate system aligned with the convolution ray direction. To do this, the density and TERMA data (already residing in the GPU's global memory from the TERMA calculation) were bound to 3D textures in the GPU's texture memory. It was then possible to trace through the volumes with a grid of parallel rays, equally spaced by a distance, Δ. During ray tracing, the volumes were sampled at intervals equal to the predefined radial step size of the convolution, P. Figure 2(a) illustrates the process. The parallel rays were incident on the TERMA map at a zenithal angle, θ. The parallel rays were evenly spaced in a 2D grid. The spacing of the parallel rays is the fourth sampling parameter introduced by the NVB convolution method. As the rays traced through the volume, they sampled the TERMA data at regular intervals defined by the radial sampling step size.

By utilizing texture memory and its intrinsic linear interpolation to resample the TERMA and density, the computational complexity remained at a maximum of $O(p)$, where p was the total number of sampling points as dictated by the radial step size and parallel ray-spacing parameters and cropped to the size of the field plus penumbra. The number of rays was dynamically allocated depending on the data size and the angle of incidence. The new nonvoxel-based data volumes

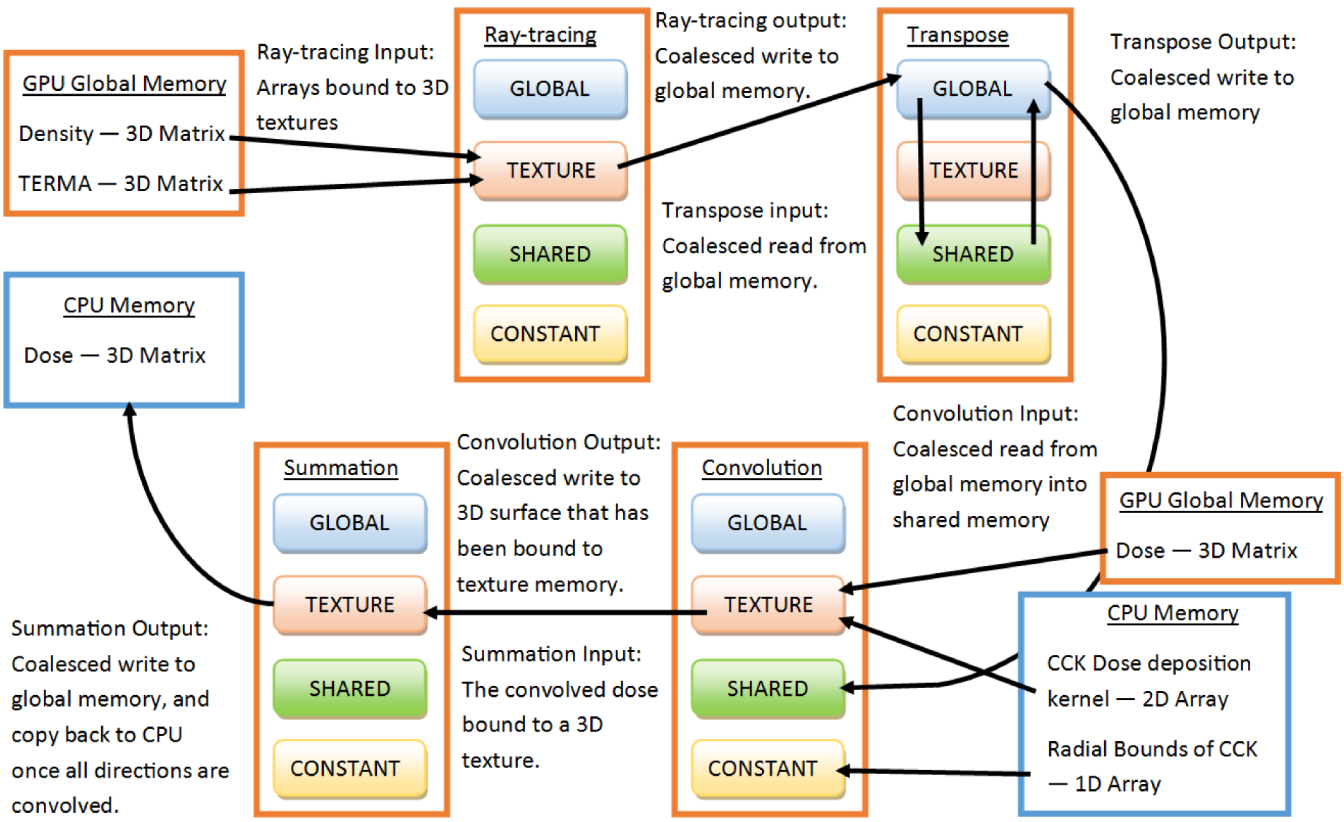


FIG. 1. GPU memory flowchart for the NVB dose convolution algorithm.

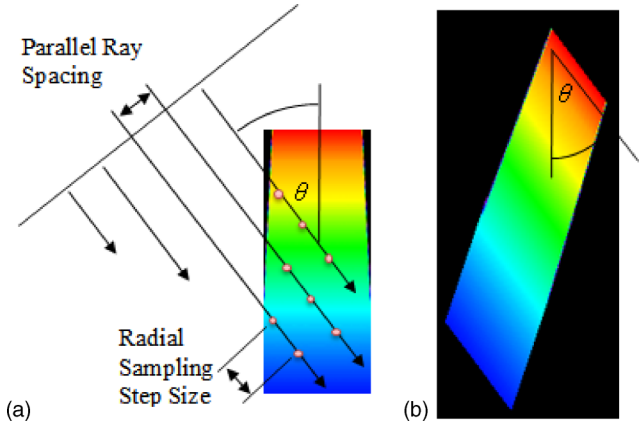


FIG. 2. NVB algorithm ray tracing. (a) An example TERMA map with the convolution angle determined by θ. (b) Ray tracing. The TERMA resampled along the convolution direction.

were written back into global memory using a coalesced write, such that adjacent memory addresses represent adjacent rays. Figure 2(b) displays the TERMA map from Fig. 2(a) in the new nonvoxel-based coordinate system.

2.C.2. Transposition

These data volumes were transposed to facilitate a coalesced memory read where adjacent memory addresses represented the sampling points along a single ray. The memory access patterns of the ray-tracing write and the line convolution read are illustrated in Fig. 3. This was done by doing coalesced reads into shared memory tiles, transposing the tiles, and performing a coalesced write back into global memory. The spatial complexity of the transposition was also $O(p)$.

2.C.3. Line convolution

Now that the data were aligned along the convolution direction, a simple line convolution was performed.[25] The data

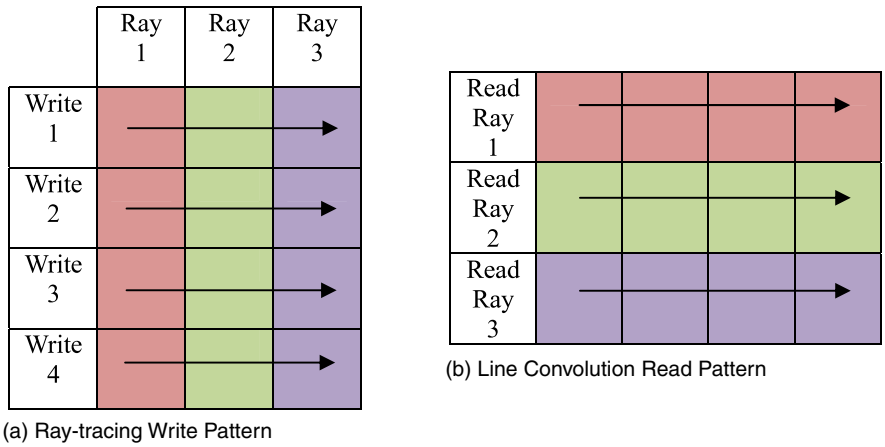


FIG. 3. NVB algorithm transposition and coalesced memory access. The nature of the ray-tracing algorithm only allows a coalesced memory write by assigning adjacent memory locations to adjacent rays, as in (a) where each column represents a different ray as it traces through the volumes. However, in order to perform a coalesced read into shared memory for the line convolution, adjacent memory locations must represent adjacent sampling points along the same ray, as illustrated in (b), where the data has been transposed such that each row now represents a different ray.

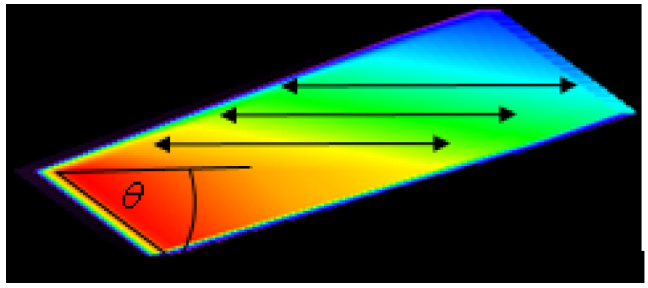


FIG. 4. NVG algorithm line convolution. The figure shows the transpose of the reformatted TERMA map in Fig. 1(b). The line convolution was performed along each ray as indicated by the arrows.

were again loaded into shared memory using a coalesced read. Each thread in the block represented one sampling point along a given ray. Figure 4 illustrates the line convolution displaying the result of convolving the transposed TERMA map.

Each GPU thread then stepped away from itself in both directions along the ray, accumulating the dose by multiplying the TERMA from each voxel with the CCK and applying the heterogeneity corrections by sampling the density. The value of the CCK was calculated by first comparing the effective radiological distance of the current voxel to an array of the radial boundaries of the CCK in the GPU's constant memory. The CCK was loaded as a 2D array into the GPU's texture memory to take advantage of the intrinsic linear interpolation on the GPU, which kept the computational complexity of the convolution step at $O(p \cdot m)$, where m was the number of radial steps along each convolution ray sampled during the convolution. By convolving both directions at once, it reduced the number of function launches and increases performance. The result was written directly into the GPU's texture memory using a surface write functionality.

2.C.4. Summation

The NVB dose data resided in the GPU's texture memory after the surface write at the end of the line convolution kernel.

A GPU thread was launched for every voxel from the original data set in the Cartesian coordinate system. The voxel's location in the convolution ray coordinate system was computed in order to sample the NVB dose data. The dose contribution of a single convolution direction converted back to Cartesian coordinates is shown in Fig. 5(a) by reading from the convolved dose that resided in texture memory. The intrinsic interpolation of texture memory was once again utilized to keep the computational complexity of this step to $O(n)$, where n was the total number of voxels in the original data set. The final dose distribution was found by accumulating the contributions from each convolution direction, as shown in Fig. 5(b).

2.D. Quantifying GPU convolution accuracy and the effect of the sampling parameters

To quantify the accuracy of the GPU implementation, we compared dose distributions for three digital phantoms with varying geometries, referred to hereafter as the accuracy phantoms, and a series of 12 segmented patient lungs. Shown in Fig. 5 are axial slices of the data sets used for the accuracy comparisons. Phantom A was a simple, homogenous block of water equivalent material. Phantom B, shown in Fig. 6(a), contained a cylinder and a box of water equivalent material (density of 1 g/cm^3) surrounded by empty space/vacuum. Phantom C, shown in Fig. 6(b), introduced a lower density region (0.317 g/cm^3) within the cylinder and serves as a simple lung phantom. The classical slab phantom and mediastinum phantom were also used for the accuracy study. The classical slab phantom, shown in Fig. 6(c),[18] contained layers of adipose tissue, muscle, bone, and lung. The mediastinum phantom, shown in Fig. 6(d), has two low density boxes surrounded by water, simulating the lungs in the chest cavity. The resolution of the accuracy phantoms was isotropic 2 mm and the size of the matrix was $128\times128\times128$.

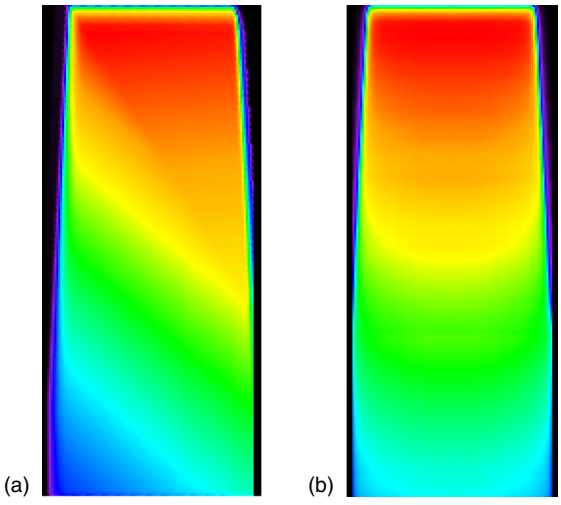


FIG. 5. NVB algorithm summation. The convolved TERMA map for the current convolution direction is (a) converted back to the original voxelized coordinate system and (b) summed with all other directions to obtain the final dose distribution.

Real patient anatomy was also used for the accuracy calculations, and a sample of the data sets are shown in Figs. 6(e)–6(h). The lungs were segmented out and exported into 128 cube data blocks with voxels of in-plane resolution of 0.12 cm and slice thickness of 0.3 cm. The volume surrounding the lungs was set to have the density of 1 g/cm^3. The tested beam configuration was an open, square field whose isocenter was placed at the volumetric center of the data set. The spectrum was a discretization of a typical 6 MV treatment beam with a flattening filter. The spectrum was divided into 14 monoenergetic bins. All dose distributions were evaluated using a 3D implementation of the gamma dose distribution comparison test,[15,26,27] in addition to direct dose comparisons. The gamma value is the Euclidean distance between the reference dose distribution and the evaluated distribution. The gamma test has two test criteria; dose difference and distance to agreement which were 2% and 0.2 cm, respectively. All gamma evaluations were performed on percent dose distributions, normalized to the maximum delivered dose. These criteria were well within clinical tolerances.[28]

The NVB convolution method has four sampling parameters that can be optimized. The zenithal, azimuthal, and radial sampling of the original convolution, along with the parallel ray spacing, introduced during ray tracing. Setting the radial sampling and parallel ray spacing allowed the dose computation to be performed at a predetermined resolution, independent of the CT resolution. For the phantom studies, gamma results for voxels with zero density were ignored. The accuracy percentages represented the fraction of voxels within the volume of interest with accumulated dose that failed the gamma test. Ground truth was taken to be the CPU based calculation employing the highest number of zenithal and azimuthal sampling rates.

2.E. Performance comparisons

To gauge the performance increases for the nonvoxel-based GPU algorithm, we performed a series of tests on three homogenous water phantoms, hereafter referred to as the performance phantoms. Each performance phantom was a 256 mm cube, with isotropic resolutions of 1, 2, and 4 mm, which resulted in dimensions of 256^3, 128^3, and 64^3 voxels, respectively. The GPU algorithm was designed using NVIDIA's CUDA, compute unified device architecture. GPU simulations were performed using an NVIDIA GTX 680 GPU which has 1536 cores and 2 GB of memory. The CPU was an Intel Core i7-3820 @ 3.60 GHz with 8 GB RAM. For an inter-GPU comparison, we also employ an NVIDIA GTX 780 Ti GPU which has 2800 cores and 3 GB of memory.

3. RESULTS AND DISCUSSION

In this section, the accuracy and performance of the NVB algorithm are reported. The results differentiate between the effect of the GPU parallelization and the effect of the sampling parameters. Section 3.A discusses the accuracy of the parallelization method when comparing similar sampling pa-

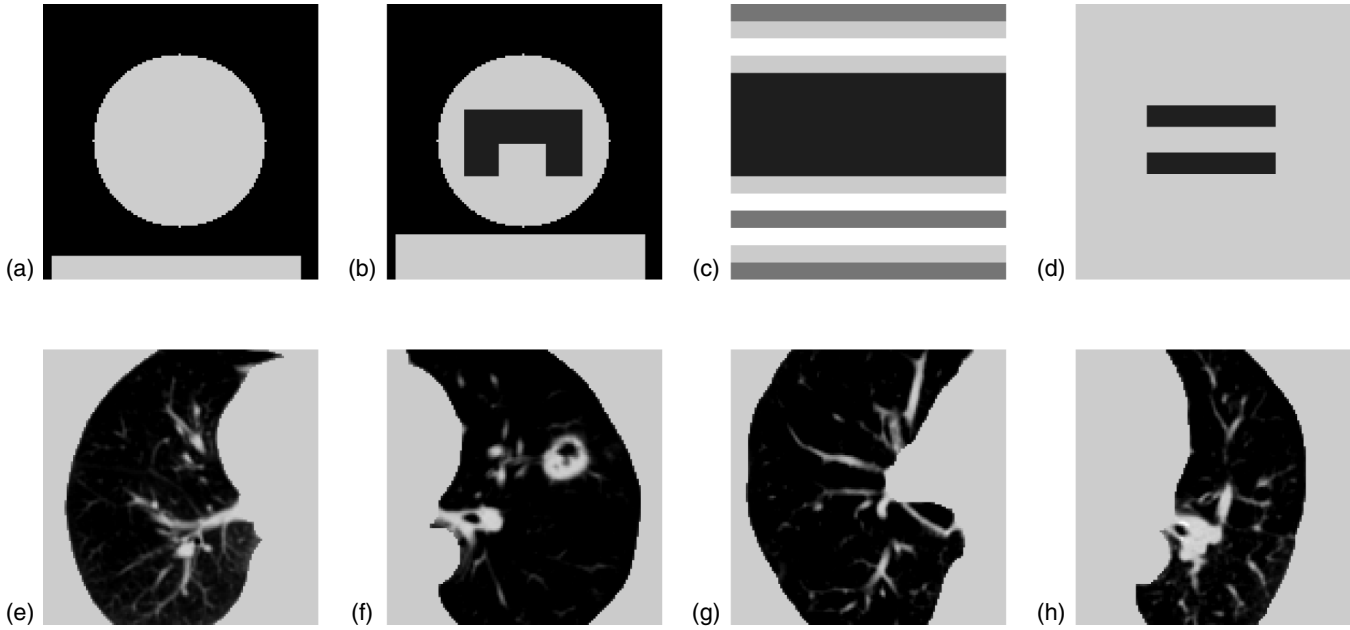


FIG. 6. Density maps for the phantom data sets. Phantoms B, C (a) and (b), the classical slab phantom (c) and mediastinum phantom (d) were used for the dose accuracy studies. A sampling of the segmented patient lungs are also displayed (e)–(h). The segmented lungs were artificially surrounded with uniform water equivalent material. The homogenous water phantoms are not displayed due to simplicity.

rameters. Section 3.B reports the effect of reducing the sampling parameters on the accuracy of the dose convolution. Section 3.C describes the performance of the NVB algorithm in comparison to the ground truth CPU algorithm and a generic GPU parallelization method. Section 3.D provides the detail on the NVB algorithm's dependence on the size of the field and the size of the target data set. The performance gain from reducing the sampling parameters is described in Sec. 3.E.

3.A. GPU accuracy

The accuracy of parallelizing the convolution algorithm was verified by examining the percent depth dose (PDD) and the profile at 10 cm depth using direct dose comparisons. Figure 7 displays the percent depth dose and cross profiles for the classical slab phantom and mediastinum phantom. Several beam sizes were examined, and the percent error was less than 1% for all voxels except for those with high dose gradients such as the penumbra and the buildup region. Much of the

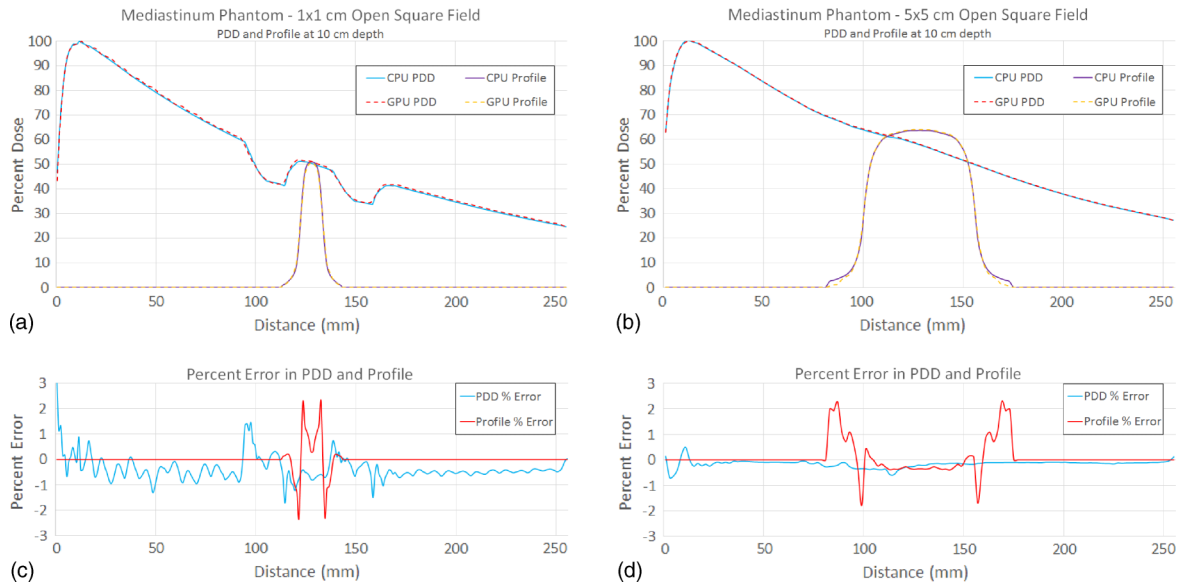


FIG. 7. Percent depth dose and cross profile comparisons. The PDD and profile for both the CPU convolution and the NVB GPU convolution for the mediastinum phantom irradiated with a 1×1 cm field is shown in (a). The corresponding percent error between the curves is displayed in (c). The same curves are displayed for the classical slab phantom irradiated with a 5×5 cm field in (b), with its respective percent error in (d).

error seen between the CPU and GPU implementations can be attributed to the fact that the convolution is being calculated on different resolution grids. The NVB algorithm resamples the data according to the parallel ray spacing and radial step size variables, and therefore is not convolving with exactly the same resolution as the voxel-based CPU algorithm. Figure 8(a) displays the results of convolving a 1×1 cm^2 field on the classical slab phantom at three different resolutions. Figure 8(b) shows the percent error in the PDD between 1 and 2 mm resolutions of the CPU, compared to the error seen between the CPU and the NVB GPU algorithms. The error between the two CPU resolutions is on the same order as the error seen between CPU and GPU. The NVB algorithm was run using 1 mm radial step size and 1 mm parallel ray spacing. When the resolution of the CPU is comparable to the NVB coordinate system, the average error decreases from 0.26% to 0.15%. Again, the largest error is seen in the high dose gradient regions. Due to the intrinsic differences that arise from convolution on different resolution calculation grids, we employed the 3D gamma dose distribution comparison tool when studying the effect of the sampling parameters on the accuracy of the NVB algorithm.

Gamma analyses were performed over the entire spectrum of angular sampling combinations, using multiple field sizes and targets. When using the same sampling parameters as the ground truth calculations performed on the CPU, we observed that all voxels calculated using the nonvoxel-based GPU parallelization passed the gamma test at 2% and 2 mm. Such a result shows that the algorithm presented in this paper provided the same accuracy as that of clinically used dose convolution implementations.

3.B. Accuracy as a function of sampling parameters

The plots in Fig. 9 display the percentage of voxels within the calculation cone that failed the gamma criteria as a function of angular sampling. Ground truth data were computed on the CPU using 48 zenithal and 48 azimuthal directions, with a 1 mm radial step size. Figures 9(a)–9(c) shows the results for the accuracy phantom data sets. For an angular sampling combination of 8 zenithal and 8 azimuthal directions, the average failure percentage for the phantoms was just 0.012% with a maximum of 0.082% for Phantom C. Figure 9(d) displays the average failure rates for the segmented lung data sets. The average failure rate was 0.74% for 8 zenithal and 8 azimuthal directions.

From the surface plots (Fig. 9), it is clear that for homogenous volumes such as phantom A, increasing the azimuthal sampling has little effect on the accuracy due to rotational symmetry about the beam direction. However, reducing the number of zenithal angles below 8 resulted in quickly increasing error because of the directionality of the kernel. They also show that for increasingly complex geometries, the total error became more dependent on the sampling rate. This was particularly evident in the azimuthal direction, as shown when comparing the patient lung data sets to phantom A, which had negligible error due to azimuthal sampling.

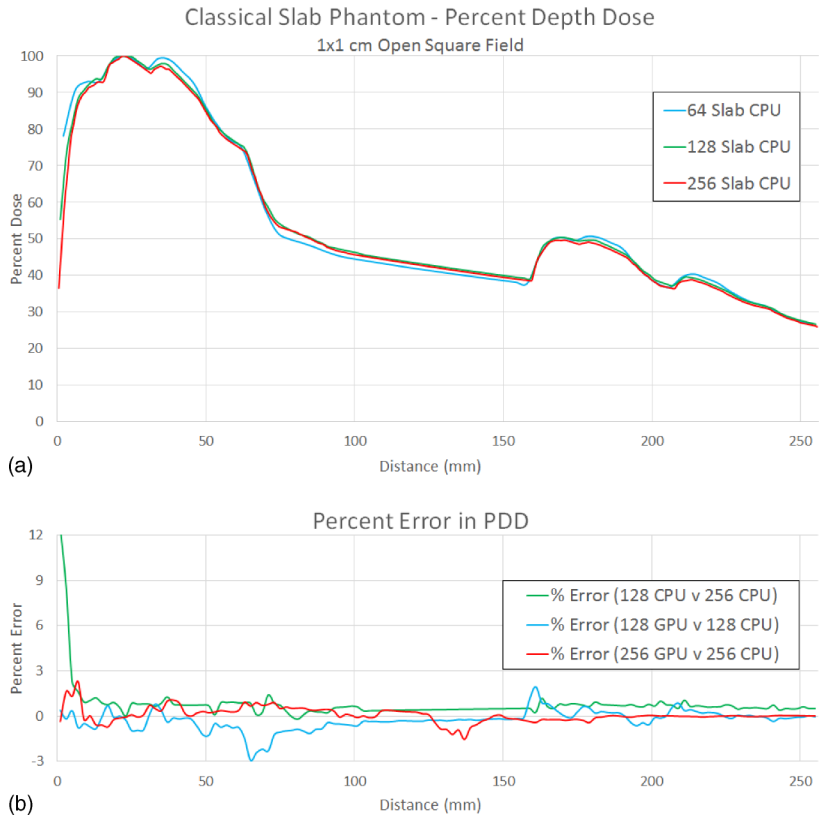


FIG. 8. Percent dose difference for different resolution calculation grids during convolution. The percent depth dose curve for a 1×1 cm field on the classical slab phantom at three resolutions: 1, 2, and 4 mm isotropic voxels (a). (b) displays the percent dose difference between the 1 and 2 mm convolutions on the CPU, as well as the difference between CPU and NVB GPU convolutions when both are calculated at 1 and 2 mm.

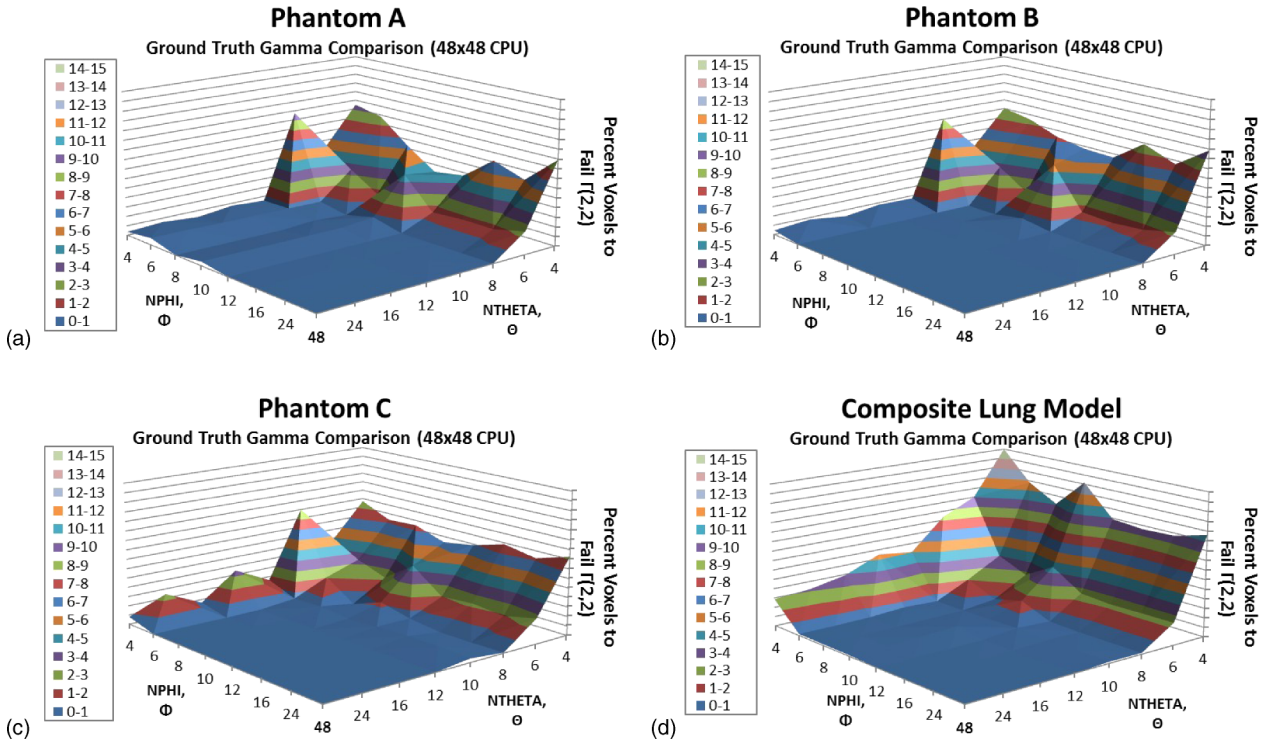


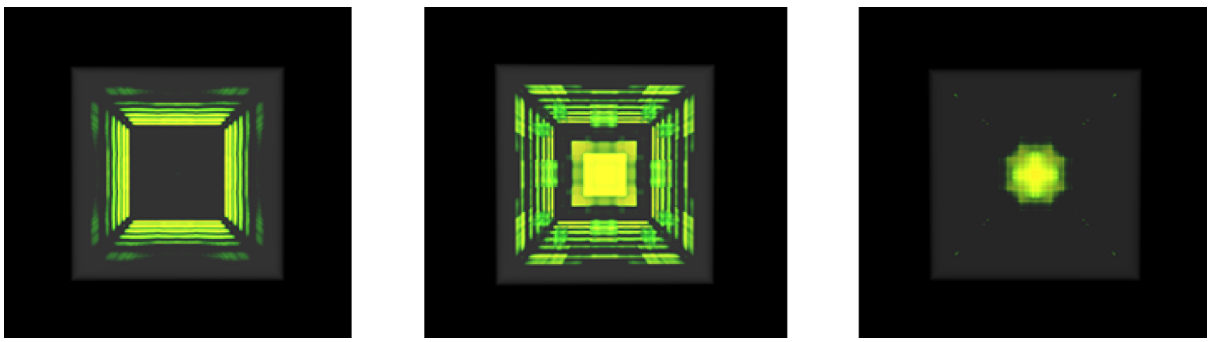
FIG. 9. Dose accuracy as a function of angular sampling. The surface plots above display the percentage of voxels to fail a gamma test with criteria of 2% and 2 mm. The number of convolution rays in the zenithal and azimuthal directions was varied from 4 to 48. Ground truth was taken to be 48 × 48 rays. Plots (a), (b), and (c) display the plots for their respective phantoms, while plot (d) shows the failure percentage averaged over all twelve lung models.

Figure 10 displays a 3D rendering of the gamma results for Phantom A from a beam's eye point of view. The volume itself can be seen as a gray cube, while the failing voxels are overlaid with a heat map. Three angular sampling combinations are displayed, illustrating the effects of reduced sampling in both the zenithal and azimuthal directions. Reducing the azimuthal sampling increased the discrepancies in the penumbra regions of the beam, while reducing the zenithal sampling causes more significant deviations along the beam axis.

Figure 11 plots the percent of voxels to fail a gamma test at 2% and 2 mm when increasing the radial sampling and parallel ray spacing. Increasing either the radial step size or the parallel ray spacing any higher than 2 mm caused rapidly increasing dose distribution modeling errors. The best results were seen when the radial step size was the same as the ground truth at 1 mm, and the parallel ray spacing was less than or equal to 2 mm.

To further illustrate the influence of the parallel ray spacing and radial step size on the integrity of the dose calculation, Figure 12 displays the percent dose difference in the PDD for a 1 × 1 cm field on the classical slab phantom.

The plots in Fig. 13 show the percent dose difference for the depth profile along the central beam axis and the beam profile perpendicular to the beam through isocenter for different sets of sampling parameters. Comparing the percent dose difference between the 48 × 48 and the 8 × 8 angular sampling



(a) 48 zenithal angles x 4 azimuthal angles

(b) 4 zenithal angles x 4 azimuthal angles

(c) 4 zenithal angles x 48 azimuthal angles

FIG. 10. Locality of dose discrepancies as a function of angular sampling. The figures display the results of a gamma test with criteria of 2% and 2 mm for a 128^3 phantom with 1 mm isotropic voxels irradiated with a 100 × 100 mm field at isocenter. (a) shows that reducing the azimuthal sampling causes errors in the penumbra region. (c) shows that reducing the zenithal sampling causes higher error along the beam axis. (b) shows the result for reduced sampling in both the zenithal and azimuthal directions.

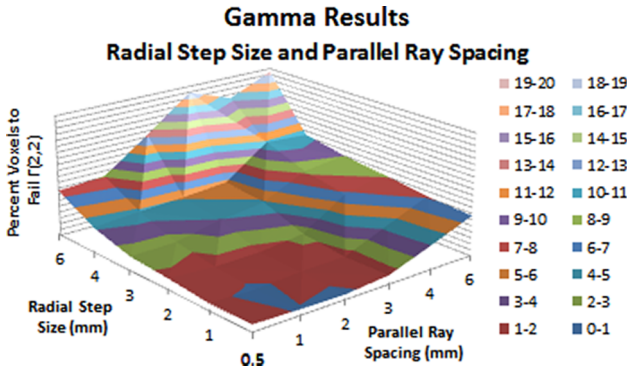


FIG. 11. Dose accuracy as a function of radial sampling and parallel ray spacing. The surface plot displays the percentage of voxels to fail the Gamma test with criteria of 2% and 2 mm. The radial sampling and parallel ray spacing define the new nonvoxel coordinate system. The size of the sampling steps has a drastic effect on the accuracy of the dose calculation.

combination, the error was less than 1% for the majority of the depth profile and beam profile and the maximum error was less than 2%. This bolstered the conclusion that the 8×8 angular sampling combination provided clinically acceptable accuracy even without considering the distance to agreement criterion of the gamma distribution analysis method. Figures 13(a) and 13(b) illustrate how reducing the radial sampling caused large errors in the buildup region and penumbra, even when the increase was as small as 1 to 2 mm. Figures 13(c) and 13(d) display similar plots where radial sampling was constant at 1 mm, and the parallel ray spacing was varied between 1 and 2 mm. As shown by the two lines with 8×8 angular sampling, increasing the parallel ray spacing caused little to no increase in the error.

3.C. GPU performance

Table I gives the computation time for the generic GPU implementation and the nonvoxel-based implementation fully optimized for the GPU architecture, calculated on three homogenous water performance phantoms. This computation times reported for both the CPU and the GPU encompass only the convolution calculation. For the GPU, this includes all kernel calls (four per convolution direction) and for the CPU, this includes all calculations within the outermost loop of the convolution. The TERMA was calculated previously and already resided in the global memory of the GPU. The average time of the TERMA calculation was 1 ms. Similarly, the density matrix was also residing in the GPU's global memory, so the extra overhead due to memory copies from CPU to GPU was minimal. On average, including the TERMA computation and memory copies added 5 and 6 ms to the overall computation time. The times are displayed for combinations of 24 zenithal angles with 16 azimuthal angles for 384 total rays (commonly used parameters for comparison testing[14] and 8 zenithal angles with 8 azimuthal angles for 64 total rays, using a 100 mm^2 field at isocenter. The computation times for both the generic GPU and NVB algorithms were linearly related to the number of convolution rays. For the most computationally expensive calculation, the NVB algorithm improved the calculation time from 45 to 2 s, a speed factor increase of over 22. The accuracy study presented above showed that an angular sampling

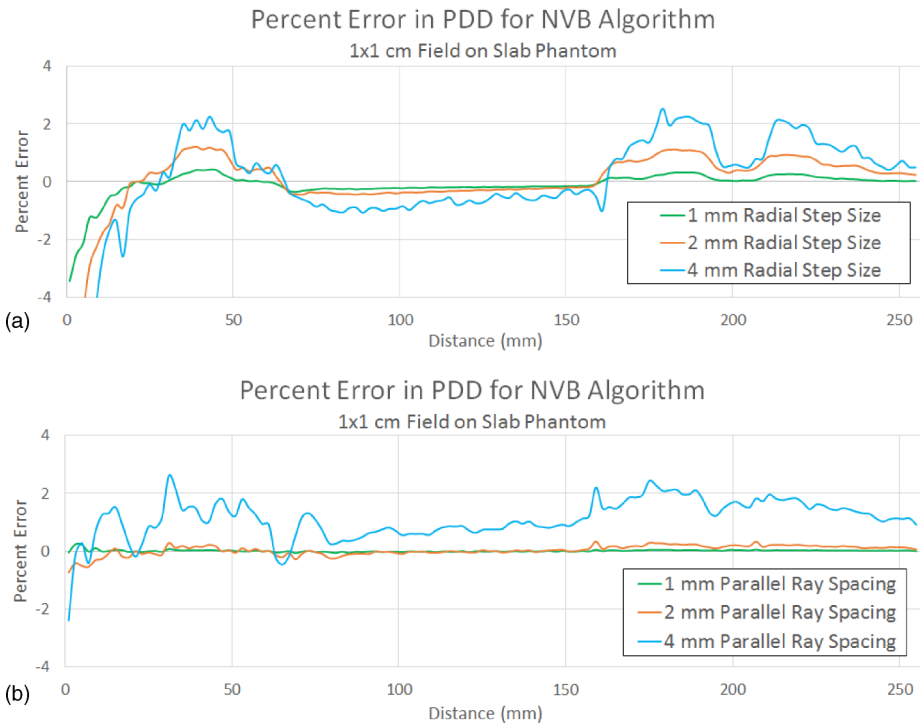


FIG. 12. Effect of parallel ray spacing and radial step size on the percent dose difference as a function of depth. In these experiments, the classical slab phantom was irradiated with a 1×1 cm^2 field. (a) shows the effect of increasing the radial step size, where 0.5 mm is taken as ground truth and all other parameters are held constant. (b) shows the effect of increasing the parallel ray spacing, where again 0.5 mm is taken as ground truth and all other parameters are held constant.

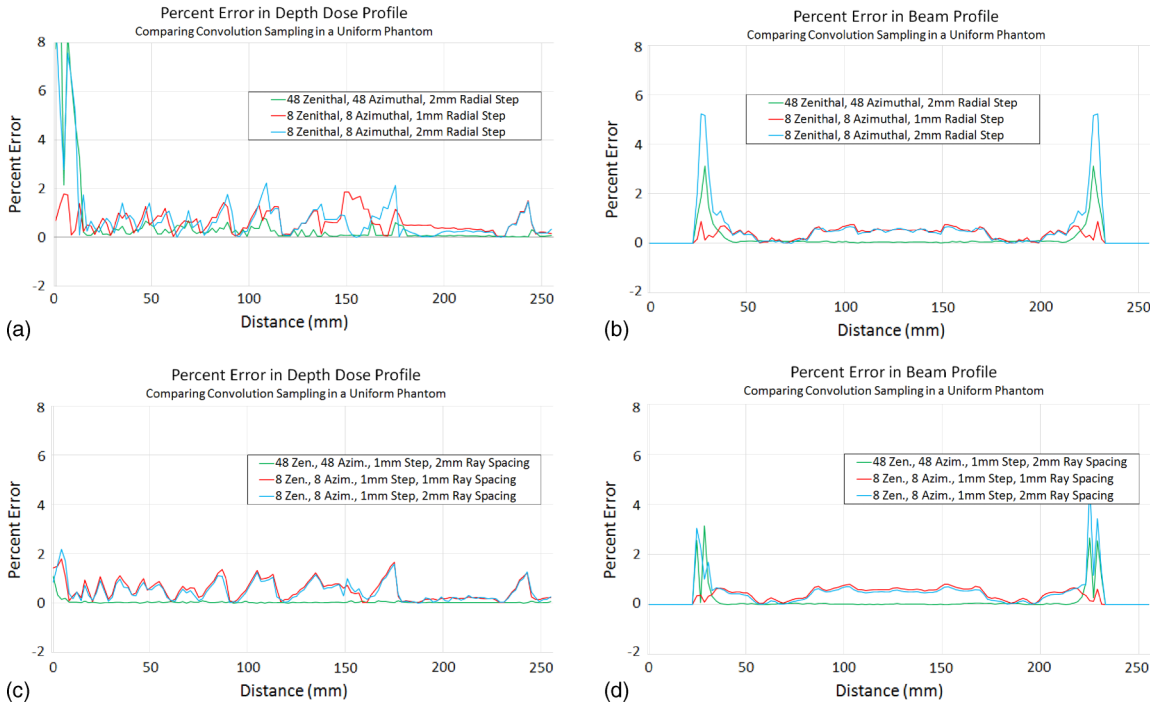


FIG. 13. Percent difference in dose profiles due to sampling. Plot (a) shows the percent difference from ground truth for the depth profile along the beam axis for three combinations of zenithal, azimuthal, and radial sampling. (b) shows the percent difference along the beam profile through isocenter for the same sampling combinations. Plots (c) and (d) introduce the effect of the parallel ray spacing during the ray-tracing step in the nonvoxel-based algorithm.

combination of 8×8 produced acceptable results, and the total convolution time for the highest resolution phantom, a 256 voxel cube with 1 mm isotropic voxels, was less than 350 ms for the nonvoxel-based algorithm.

Table II presents the performance gains when comparing identical sampling parameters across all three algorithms. The results presented were averaged over every angular sampling combination and are shown with the standard deviation. For the 64^3 phantom, the generic GPU parallelization technique provided an acceleration factor of nearly 60 over the CPU. The NVB implementation boosted the performance to more than 86 times over the CPU. The comparison showed the NVB implementation increased performance 1.46× over the generic GPU implementation on average. This advantage grew as the data size and computational complexity increased.

TABLE I. Computation times.

GPU hardware directions	Generic GPU GTX 680 384 rays (s)	NVB GPU GTX 680 384 rays (s)	Generic GPU GTX 680 64 rays (s)	NVB GPU GTX 680 64 rays (s)
$256 \times 256 \times 256$ (1 mm resolution)	45.03	2.04	7.42	0.343
$128 \times 128 \times 128$ (2 mm resolution)	8.01	1.61	1.30	0.274
$64 \times 64 \times 64$ (4 mm resolution)	2.50	1.70	0.42	0.282

As seen from the ratios of the CPU over the generic GPU convolution times, the advantage of the GPU increased with an increase in the data resolution because the resolution was directly related to computational effort. However, the ratios of the generic GPU implementation convolution time over the NVB implementation convolution time showed that the optimized memory accesses of the NVB method were able to maintain significantly higher throughput as computational effort increased. The NVB method was nearly 22 times faster than the generic GPU implementation for the 256^3 phantom. This resulted in a total acceleration factor of more than 4000 when comparing the CPU algorithm against the NVB GPU parallelization technique. The significant improvement from the generic GPU method to the NVB method could be attributed to an intrinsic data size independence of the NVB technique, which will be further discussed in the next section.

3.D. Field size and data size dependence

An advantage of transforming the convolution into a nonvoxel-based coordinate system was that the calculation grid

TABLE II. Average acceleration using the GPU parallelization.

Acceleration phantom resolution	64^3 phantom 4 mm voxels	128^3 phantom 2 mm voxels	256^3 phantom 1 mm voxels
CPU/generic GPU	59.26 ± 1.66	113.2 ± 1.75	193.7 ± 12.7
Generic GPU/NVB GPU	1.46 ± 0.04	4.8 ± 0.14	21.6 ± 0.6
CPU/NVB GPU	86.63 ± 3.49	546.4 ± 20.3	4175.5 ± 354.9

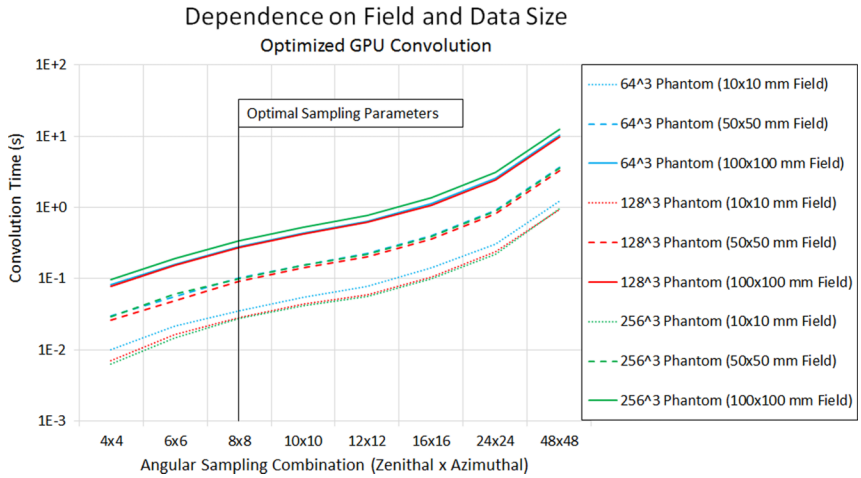


FIG. 14. Convolution computation time as a function of field size and data size. The plot displays the convolution time as a function of the angular sampling combination for three different field sizes on each of the performance phantoms. The results are clearly grouped by field size, while there is little distinction between the phantom data size.

was then controlled exclusively by the sampling parameters. The resolution of the grid in the nonvoxel-based system was determined by the radial step size and the parallel ray spacing, and the number of rays cast through the volume depended only on the size of the field and the parallel ray spacing. This eliminated the dependence of the computation time on the original data resolution. Figure 14 shows the convolution time for the nonvoxel-based algorithm using a 10×10 mm^2 field, a 50×50 mm^2 field, and a 100×100 mm^2 field, over a spectrum of angular sampling combinations. The data are clearly grouped by field size but more interestingly are the lack of separation across the data size. The nonvoxel-based algorithm proved to be independent of the data volume because it resampled the data according to the radial step size and the parallel ray-spacing parameters. This caused large performance gains over the CPU algorithm.

3.E. Sampling acceleration

The ground truth calculation time was taken as the maximum convolution sampling combination of 48 zenithal and azimuthal angles. The sampling acceleration was directly related to the number of rays used during convolution. Reducing the angular sampling of each angle by a factor of 2 resulted in four times speedup, and so forth. By reducing the angular sampling to 8 zenithal angles and 8 azimuthal angles, the performance was increased by a factor of 36. As shown in Fig. 15, combining the reduced sampling acceleration with the acceleration provided from the nonvoxel-based GPU algorithm pushed the maximum acceleration over 175 000 times for the 256 voxel cube phantom and a 100×100 mm^2 field. While the convolution times were very similar across data sizes for the NVB algorithm, Fig. 15 shows a fairly consistent increase in acceleration around one order of magnitude as the data size increased.

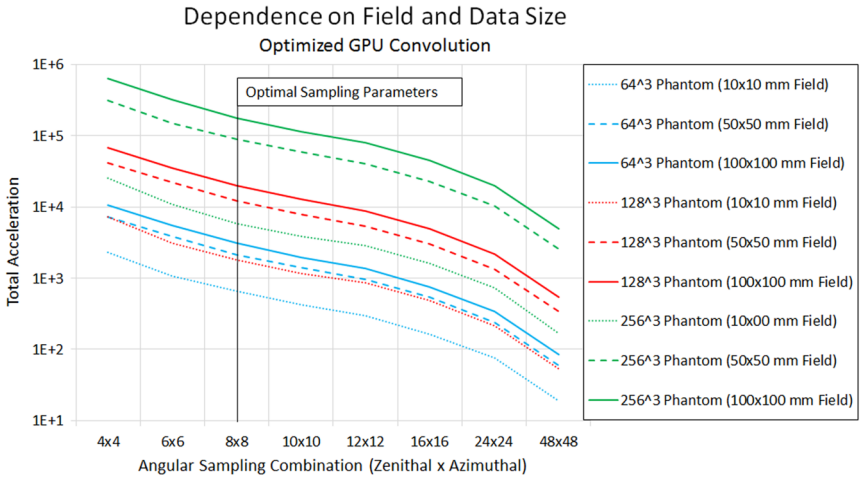


FIG. 15. Performance acceleration as a function of field size and data size. The plot shows the total acceleration due to parallelization on the GPU and reduced angular sampling. Greater computational effort results in greater acceleration. The highest resolution phantom irradiated with the largest field size results in the most voxels involved in the convolution calculation and the greatest acceleration. The data size has a strong correlation with acceleration, as the 10×10 mm^2 field size on the 256^3 phantom shows greater acceleration than the 100×100 mm^2 field size on the 64^3 phantom.

TABLE III. Acceleration using the optimal sampling parameters and GPU parallelization.

	64^3 phantom 4 mm voxels	128^3 phantom 2 mm voxels	256^3 phantom 1 mm voxels
CPU/generic GPU	2100	4100	8200
CPU/NVB GPU	3100	19 500	176 000

Also, the smallest field on the highest resolution phantom saw larger accelerations than the largest field on the lowest resolution phantom. The total combined acceleration for both the generic GPU and NVB implementations from GPU parallelization and reduced sampling are presented in Table III for each of the three acceleration phantoms.

4. DISCUSSION

The convolution is scalable to multiple GPUs. Theoretically, there was a direct relationship between the number of GPUs employed and the performance gains for large workloads.[29] These simulations used an open square field for verification and comparison. Utilizing multiple GPUs would allow calculating treatment plans with multiple fields. Additional aspects for investigation are the incorporation of complex beam geometries, multiple fields, multi-leaf collimators, and the varied fluence maps used in intensity modulated therapies.

Figure 16(a) shows the depth dose and cross profiles for a 5×5 cm field in a homogenous water phantom from a Monte Carlo generated dose distribution using the same energy spectrum as the convolution algorithms, a Monte Carlo generated distribution using the phase space energy spectrum data from a Varian TrueBeam® linear accelerator with flattening filter provided by Varian Medical Systems, the voxel-based CPU convolution used as ground truth in this paper, and the NVB GPU convolution presented in this paper. The Monte Carlo generated dose distributions were created using MCNP4C, with histories of 2×10^9 photons to achieve less than 2% statistical variation. Both the CPU convolution and the NVB convolution were calculated using 8 azimuthal angles and 8 elevation angles, with a radial step size of 1 mm. Additionally, the NVB convolution used a parallel ray spacing of 1 mm. Significant differences can be seen along each profile between the convolution methods and the other data. Figure 16(b) plots the percent dose difference for both the CPU convolution and the NVB GPU convolution. While the CPU convolution is regarding as ground truth in this paper, the plot shows that there is definitely room for improvement to more realistically recreate the actual dose distributions measured from the treatment machine and the gold standard Monte Carlo dose calculations.

With further performance enhancement, we should be able to deconstruct the polyenergetic kernel and calculate the energies independently, which will eliminate some assumptions and estimations currently used in the convolution/superposition algorithm, and produce a dose distribution closer to the Monte Carlo distribution.

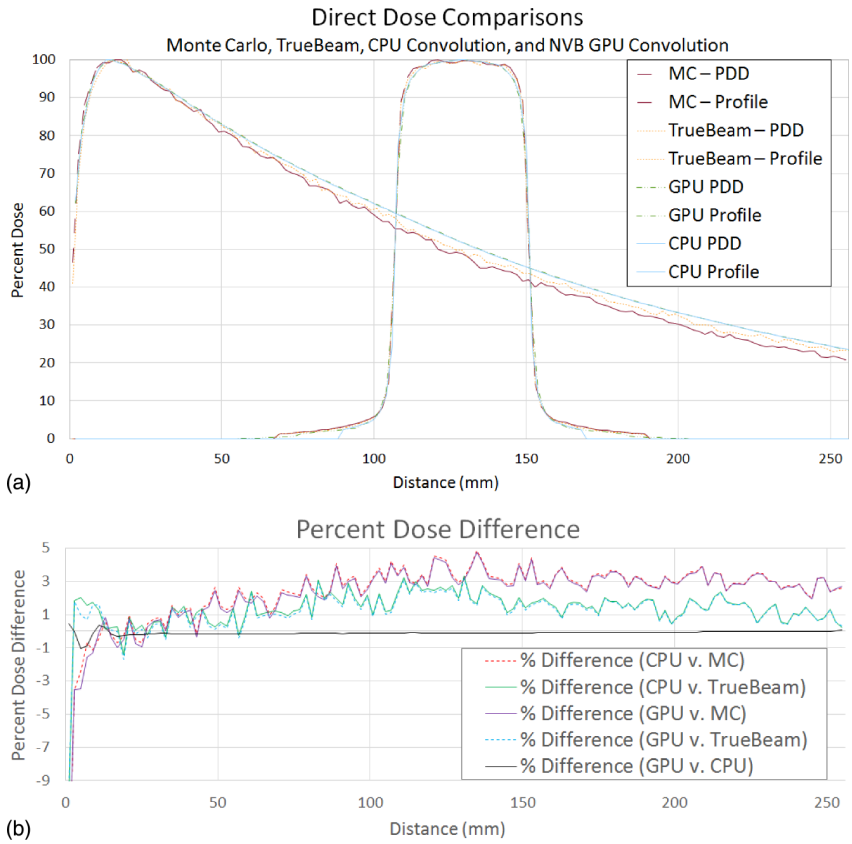


FIG. 16. Direct dose comparisons between Monte Carlo, TrueBeam, voxel-based CPU convolution, and NVB GPU convolution. The depth dose and cross profiles for a 5×5 cm^2 field at 10 cm depth are shown for each dose distribution source in (a). The percent difference for the depth dose profiles is shown in (b).

TABLE IV. Percentage of total GPU computation time for NVB convolution algorithm.

Acceleration phantom directions	64^3 phantom		128^3 phantom		256^3 phantom	
	384 rays	64 rays	384 rays	64 rays	384 rays	64 rays
Ray tracing	4.51	4.40	10.95	10.68	19.45	19.67
Transposition	4.60	4.50	3.96	3.92	2.96	2.96
Line convolution	90.06	90.27	82.17	82.54	61.10	61.02
Summation	0.83	0.83	2.92	2.85	16.49	16.35

Table IV tabulates the computation time dedicated to each of the four components of the nonvoxel-based algorithm for both the high and low convolution ray count and each of the performance phantoms. The performance bottleneck still resided with the convolution step due to the requirement for ray-casting along the line to accumulate the effective radiological distance when applying the heterogeneity correction.

We are currently investigating a method to stretch the data volume according to effective radiological distance during the initial ray tracing. The convolution could then be even further parallelized as each data point in the nonvoxel-based coordinate system would represent an equal amount of attenuation. The algorithm would no longer have to step along the rays to apply the heterogeneity correction but simply apply multiplication and summation reduction techniques which are much more suitable for parallel architecture.

As the computing hardware continually improves, the software design considerations discussed in this paper become more and more critical to maximizing performance. Just as Table I reported the improvement in computation time of the NVB algorithm compared to the generic GPU algorithm, Table V compares the performance of our NVB algorithm for the GTX 680, which the algorithm was developed on, and the newest card released by NVIDIA, the 780 TI. An average speedup over 1.8 times was seen for both high and low number of convolution rays on all three of the performance phantoms.

5. CONCLUSIONS

The convolution parameters (zenithal angle sampling, azimuthal angle sampling, radial step size, and parallel ray spacing) could be optimized for maximum acceleration with minimal loss of accuracy. This was demonstrated by performing dose calculations using five digital phantoms and twelve patient lung CTs. In both cases, a zenithal/azimuthal combination of 8/8 provided the best performance while maintaining accuracy. Both the phantoms and lung models passed a gamma test of 2% or 2 mm at better than 99%.

Splitting the acceleration between the sampling optimization and GPU implementation showed a consistent speedup of about 36 when reducing the convolution sampling from 48/48 to 8/8, while the GPU implementation provided a second improvement level between 86 and nearly 4200 times speedup depending on the data size and resolution. This resulted in total performance gains of just over 3000 times for the smallest 64 voxel performance phantom and over 175 000 times for the largest 256 voxel performance phantom when compared to a nonoptimized CPU algorithm. Optimizing the NVB algorithm for the GPU architecture also improved the performance significantly compared to a generic GPU implementation, providing nearly 22 times speedup for the 256 voxel performance phantom.

Future work will see the expansion of our nonvoxel-based convolution to a multi-GPU framework. Implementing the outlined optimization strategies and eliminating many of the assumptions and estimations currently employed by convolution/superposition to reduce computation times, this method can improve both accuracy and speed for computing on-the-fly dose distributions. These improvements are valuable for the clinical dosimetry efficiency and will facilitate real-time adaptive radiotherapy.

TABLE V. NVB computation times with improving hardware.

Directions	384 rays		64 rays	
GPU hardware	GTX 680 (s)	GTX 780 TI (s)	GTX 680 (s)	GTX 780 TI (s)
$256 \times 256 \times 256$	2.04	1.06	0.343	0.177
$128 \times 128 \times 128$	1.61	0.89	0.274	0.149
$64 \times 64 \times 64$	1.70	0.98	0.282	0.161

[a]Author to whom correspondence should be addressed. Electronic mail: jneylon@mednet.ucla.edu

[1]K. Britton, L. Dong, and R. Mohan, "Image guidance to account for interfractional and intrafractional variations: From a clinical and physics perspective," in *Image-Guided Radiotherapy of Lung Cancer*, edited by J. Cox, J. Chang, and R. Komaki (Informa Healthcare, Inc., New York, NY, 2007).
[2]X. A. Li *et al.*, "Development of an online adaptive solution to account for inter- and intra-fractional variations," Radiother. Oncol. **100**(3), 370–374 (2011).
[3]Liu *et al.*, "Characterization and management of interfractional anatomic changes for pancreatic cancer radiotherapy," Int. J. Radiat. Oncol., Biol., Phys. **83**(3), e423–e429 (2012).
[4]J. Stewart *et al.*, "Automated weekly replanning for intensity-modulated radiotherapy of cervix cancer," Int. J. Radiat. Oncol., Biol., Phys. **78**(2), 350–358 (2010).
[5]A. Bujold *et al.*, "Image-guided radiotherapy: Has it influenced patient outcomes?," Semin. Radiat. Oncol. **22**(1), 50–61 (2012).
[6]L. Xing, J. Siebers, and P. Keall, "Computational challenges for image-guided radiation therapy: Framework and current research," Semin. Radiat. Oncol. **17**(4), 245–257 (2007).
[7]J. Sanders and E. Kandrot, *CUDA By Example* (Pearson Education, Inc., Boston, MA, 2011).
[8]D. Kirk and W.-m. Hwu, *Programming Massively Parallel Processors* (Elsevier, Inc., Burlington, MA, 2010).
[9]L. Riha and H. El-Sayed, "Real-time motion object tracking using GPU," in *International Conference on Computer Systems and Applications* (IEEE Computer Society, Washington, DC, 2011), pp. 301–304.
[10]S. Hissoiny, B. Ozell, and P. Despres, "Fast convolution-superposition dose calculation on graphics hardware," Med. Phys. **36**(6), 1998–2005 (2009).
[11]S. Hissoiny *et al.*, "A convolution-superposition dose calculation engine for GPUs," Med. Phys. **37**(3), 1029–1037 (2010).
[12]R. Jacques *et al.*, "Towards real-time radiation therapy: GPU accelerated superposition/convolution," Comput. Methods Programs Biomed. **98**(3), 285–292 (2009).
[13]R. Jacques *et al.*, "Real-time dose computation: GPU-accelerated source modeling and superposition/convolution," Med. Phys. **38**(1), 294–305 (2010).

[14]Q. Chen, M. Chen, and W. Lu, "Ultrafast convolution/superposition using tabulated and exponential kernels on GPU," Med. Phys. **38**(3), 1150–1161 (2011).
[15]Q. Chen and W. Lu, "Validation of GPU based tomotherapy dose calculation engine," Med. Phys. **39**(4), 1877–1886 (2012).
[16]W. Lu, "A non-voxel-based broad-beam (NVBB) framework for IMRT treatment planning," Phys. Med. Biol. **55**, 7175–7210 (2010).
[17]Mackie, "A convolution method of calculating dose for 15-MV x-rays," Med. Phys. **12**, 188–196 (1985).
[18]A. Ahnesjo, "Collapsed cone convolution of radiant energy for photon ose calculation in heterogeneous media," Med. Phys. **16**, 577–592 (1989).
[19]R. Siddon, "Fast calculation of the exact radiological for a three-dimensional array," Med. Phys. **12**(2), 252–255 (1985).
[20]P. W. Hoban, "Accounting for the variation in collision kerma-to-terma ratio in polyenergetic photon beam convolution," Med. Phys. **22**(12), 2035–2044 (1995).
[21]J. H. Hubbell and S. M. Seltzer, "Tables of x-Ray mass attenuation coefficients and mass energy-absorption coefficients from 1 keV to 20 MeV for elements Z = 1 to 92 and 48 additional substances of dosimetric interest," Available URL: http://www.nist.gov/pml/data/xraycoef/index.cfm (2011).
[22]T. R. Mackie *et al.*, "Generation of photon energy deposition kernels using the EGS Monte Carlo code," Phys. Med. Biol. **33**(1), 1–20 (1988).
[23]W. Lu *et al.*, "Accurate convolution/superposition for multi-resolution dose calculation using cumulative tabulated kernels," Phys. Med. Biol. **50**(4), 655–680 (2005).
[24]P. W. Hoban, D. C. Murray, and W. H. Round, "Photon beam convolution using polyenergetic energy deposition kernels," Phys. Med. Biol. **39**(4), 669–685 (1994).
[25]B. Qin *et al.*, "GPU-based parallelization algorithm for 2D line integral convolution," Lect. Notes Comput. Sci. **6145**, 397–404 (2010).
[26]X. Gu, X. Jia, and S. Jiang, "GPU-based fast gamma index calculation," Phys. Med. Biol. **56**(5), 1431–1441 (2011).
[27]D. Low and J. Dempsey, "Evaluation of the gamma dose distribution comparison method," Med. Phys. **30**(9), 2455–2465 (2003).
[28]B. Fraass *et al.*, "American association of physicists in medicine radiation therapy committee task group 53: Quality assurance for clinical radiotherapy treatment planning," Med. Phys. **25**(10), 1773–1829 (1998).
[29]A. P. Santhanam *et al.*, "A multi-GPU real-time dose simulation software framework for lung radiotherapy," Int. J. Comput. Assisted Radiol. Surg. **7**(5), 705–719 (2011).