Parallel beamlet dose calculation via beamlet contexts in a distributed multi-GPU framework

Ryan Neph[a)], Cheng Ouyang, John Neylon, Youming Yang, and Ke Sheng
Department of Radiation Oncology, University of California Los Angeles, 200 Medical Plaza, #B265, Los Angeles, California 90095, USA

(Received 11 February 2019; revised 3 June 2019; accepted for publication 3 June 2019; published xx xxxx xxxx)

Purpose: Dose calculation is one of the most computationally intensive, yet essential tasks in the treatment planning process. With the recent interest in automatic beam orientation and arc trajectory optimization techniques, there is a great need for more efficient model-based dose calculation algorithms that can accommodate hundreds to thousands of beam candidates at once. Foundational work has shown the translation of dose calculation algorithms to graphical processing units (GPUs), lending to remarkable gains in processing efficiency. But these methods provide parallelization of dose for only a single beamlet, serializing the calculation of multiple beamlets and under-utilizing the potential of modern GPUs. In this paper, the authors propose a framework enabling parallel computation of many beamlet doses using a novel beamlet context transformation and further embed this approach in a scalable network of multi-GPU computational nodes.

Methods: The proposed context-based transformation separates beamlet-local density and TERMA into distinct beamlet contexts that independently provide sufficient data for beamlet dose calculation. Beamlet contexts are arranged in a composite context array with dosimetric isolation, and the context array is subjected to a GPU collapsed-cone convolution superposition procedure, producing the set of beamlet-specific dose distributions in a single pass. Dose from each context is converted to a sparse representation for efficient storage and retrieval during treatment plan optimization. The context radius is a new parameter permitting flexibility between the speed and fidelity of the dose calculation process. A distributed manager-worker architecture is constructed around the context-based GPU dose calculation approach supporting an arbitrary number of worker nodes and resident GPUs. Phantom experiments were executed to verify the accuracy of the context-based approach compared to Monte Carlo and a reference CPU-CCCS implementation for single beamlets and broad beams composed by addition of beamlets. Dose for representative 4π beam sets was calculated in lung and prostate cases to compare its efficiency with that of an existing beamlet-sequential GPU-CCCS implementation. Code profiling was also performed to evaluate the scalability of the framework across many networked GPUs.

Results: The dosimetric accuracy of the context-based method displays <1.35% and 2.35% average error from the existing serialized CPU-CCCS algorithm and Monte Carlo simulation for beamlet-specific PDDs in water and slab phantoms, respectively. The context-based method demonstrates substantial speedup of up to two orders of magnitude over the beamlet-sequential GPU-CCCS method in the tested configurations. The context-based framework demonstrates near linear scaling in the number of distributed compute nodes and GPUs employed, indicating that it is flexible enough to meet the performance requirements of most users by simply increasing the hardware utilization.

Conclusions: The context-based approach demonstrates a new expectation of performance for beamlet-based dose calculation methods. This approach has been successful in accelerating the dose calculation process for very large-scale treatment planning problems - such as automatic 4π IMRT beam orientation and VMAT arc trajectory selection, with hundreds of thousands of beamlets - in clinically feasible timeframes. The flexibility of this framework makes it as a strong candidate for use in a variety of other very large-scale treatment planning tasks and clinical workflows. [https://doi.org/10.1002/mp.13651]

Key words: beamlet, convolution, distributed computing, dose calculation, GPU

1. INTRODUCTION

Modern radiation treatment planning is powered by inverse optimization algorithms that require causal information connecting the plan delivery parameters to the resultant patient dose distribution. This information is encapsulated in a dose influence matrix, consisting of the dose of individual beamlets, which are the smallest deliverable unit whose geometry is typically determined by the multi-leaf collimator width. Monte Carlo (stochastic) simulation is regarded as the gold standard for dosimetric accuracy but remains impractically slow for very large scale (VLS) optimization problems. On

the other hand, deterministic approaches provide a faster approximation by convolving reusable dose spread kernels over analytically computed TERMA fields on each unique patient geometry. The popularity of deterministic convolution superposition (C/S) solvers such as collapsed-cone convolution superposition (CCCS)[1,2] and analytical anisotropic algorithm (AAA)[3] have enabled acceptably accurate clinical dose calculation that can typically be performed with an order of magnitude reduction in time required for general purpose (Geant4) and even special purpose (VMC++) Monte Carlo methods in some circumstances.[4]

In practice, the dose influence matrix calculation speed is generally acceptable with modern computers for IMRT plans involving only a few predetermined beams. For arc optimizations and TomoTherapy, two orders of magnitude greater number of beamlets are needed to construct the matrix. Owing to the evidence that non-coplanar beam orientations and automatic selection of beams and arc trajectories has been shown to produce improved plan quality,[5–12] there is great research interest in automatic orientation selection from a much larger set of beam candidates by learning-based[13] and dose-driven approaches.[14–17] One such method, non-coplanar IMRT with beam orientation optimization,[16] selects beams from several hundred candidates, escalating the requirement for dose calculation proportionally. More recently, dynamic collimator rotation[18] and non-coplanar VMAT[5] have been developed for further improved dosimetry and delivery efficiency, pushing the requirement of beamlet dose for optimization to be ~1000 times greater than that of fixed beam IMRT plans to account for the additional degrees of freedom. Dose calculation for the increasing number of beamlets can be a slow process by clinical standards, particularly when higher dosimetric accuracy is desired. There has not yet been an improvement to dose calculation processes using collapsed cone beamlet dose generation that would make these VLS treatment planning methods clinically tractable, despite the clear dosimetric benefits granted for heterogeneous geometries. The purpose of this work is to improve the dose calculation speed for these VLS planning methods so that standard clinical implementation may be achieved.

Since deterministic dose calculation is an embarrassingly parallel computational problem, graphics processing units (GPUs) with a large number of computational cores have found widespread success in accelerating dose calculation for treatment plan optimization and validation purposes. Chen et al. employed efficient GPU memory coalescing and analytical dose spread kernels to achieve 1000–3000× speedup over the central processing unit (CPU)-CCCS implementation for TomoTherapy dose calculation.[19,20] Neylon et al. further optimized memory access speeds during GPU-CCCS convolution by first transforming voxelized TERMA to a basis aligned with each collapsed-cone direction and subsequently carrying out efficient parallelized line convolutions, demonstrating further acceleration over the CPU method.[21] Tian et al. developed a GPU Monte Carlo dose calculator (goMC) based on the OpenCL GPU computing framework to enable widespread adoption of Monte Carlo simulation across all popular GPU hardware architectures.[22] Ziegenhein et al. delocalized the dose calculation process with an integrated cloud-based Monte Carlo framework that allows dynamic scaling of computational resources as needed to reduce workstation cost and complexity, improving the expectation for performance scaling with additional hardware on short-lived simulations where gains were previously pervasive.[23] Park et al. performed beamlet-based dose convolution with adaptive finite-sized pencil beam kernels to reduce the number of beamlets required to model arbitrary field shapes and accelerate volumetric dose verification for the optimized fluence maps.[24] Cho et al. validated the use of a GPU-accelerated convolution-superposition method for kilovoltage dose calculation in small animal irradiation research.[25]

The foundation for our approach is the nonvoxel-based (NVB) GPU dose calculation algorithm of Neylon et al.[21] which optimizes previous GPU-based CCCS methods[26–28] by employing efficient GPU memory handling practices. The NVB algorithm is an improvement over these algorithms in that it reduces latency in device memory access by successive transformation of the CT density data to align it with each collapsed-cone ray enabling efficient line convolution. Like the NVB approach of Neylon et al., we treat the convolution operation on a continuous domain with interpolation during dose kernel sampling and dose spread. Unlike the NVBB approach of Lu,[29] which treats TERMA and dose calculation in a continuous domain without discretely modeling beamlets, we maintain the standard voxel-based beamlet-superposition (VBS) representation in the output of our algorithm such that we follow the path of precalculating discrete, beamlet-specific dose distributions for use during plan optimization. We make this choice primarily to maintain compatibility with the variety of beamlet-based planning techniques.

Our contributions are twofold. First, we propose a novel modification to the existing GPU implementation of full beam deterministic dose calculation, enabling efficient low-level parallel computation of beamlet-specific dose using a *beamlet-context transformation*. Second, we implement our beamlet-based GPU dose calculation algorithm in a scalable distributed framework supporting flexible high-level multi-GPU acceleration. Our framework greatly improves the efficiency of the VBS method for use in VLS optimization problems such as dose-driven automatic IMRT beam orientation[16] and VMAT trajectory optimization,[5,18] π^6, and TomoTherapy[30] treatment planning. In this study, we introduce the framework for our method, provide some dosimetric characterization for standalone beamlets and their composition as a broad beam, measure its computational performance, and discuss the scalability across networked computational nodes. We also discuss the computational efficiencies enabled by our proposed method and how it could potentially benefit VLS optimization problems but recognize that classification of dosimetric accuracy in clinical treatment planning settings is a more involved matter and leave such investigation to future work.

2. MATERIALS AND METHODS

In this section, we describe our novel beamlet-context approach for efficient beamlet-based dose calculation on a GPU. Next, we show that our method may be further parallelized in a scalable manner across a network of multi-GPU compute nodes. Finally, we describe the experiments designed to quantify our framework's dosimetric accuracy against Monte Carlo and CPU-CCCS reference doses and computational speed in comparison to an existing GPU model-based method.

2.A. Nonvoxel-based dose calculation

2.A.1. Beamlet-based dose by intra-beam parallelization

Dividing the dose calculation problem into per-beam tasks is a trivial matter. While others have chosen to further separate the problem into per-beamlet tasks, we instead chose to calculate dose for these beamlets simultaneously. Our algorithm processes each beam as a unit, concurrently producing independent dose distributions for each of the beam's active beamlets before writing them to file and continuing with the next beam. This innovation is the key to achieving an efficient and scalable algorithm that minimizes GPU execution and memory management overhead. Details of the low-level parallelization are explained in the subsequent sections.

TERMA calculation: Dose calculation for each beam begins by first generating a binary fluence map where active beamlets are assigned a unit fluence. Active beamlets are defined by projecting the target onto the fluence plane at the isocenter [Fig. 1(a)] and detecting intersections with the target volume along each of nine rays configured for each beamlet as shown in Fig. 1(b). If any sample ray intersects any part of the target volume, the beamlet is considered active and its dose will be calculated. This approach is used frequently for beamlet dose calculation and effectively minimizes the computational complexity of the full problem without sacrificing plan quality.

The binary fluence map is used in calculating the TERMA by means of a path-length tracing procedure[31] that implements a modified version of Siddon's[32] algorithm better suited to the GPU. For every voxel, i, in the volume, a ray is traced between the source position and the voxel's center, along which, the radiological path length is accumulated, according to Eq. , for a line segment of variable length l_j through each voxel j ϵ R_i of density ρ_j.

$$d_i = \sum_{j \in R_i} l_j \rho_j \tag{1}$$

To maximize calculation speed, dose is calculated assuming a constant polyenergetic beam spectrum. Since beam hardening effects change the true beam spectrum within an attenuating medium, an auxiliary value, T_i^*, is calculated for each voxel, i, and used instead of the actual TERMA, T_i, during dose convolution. Equation shows the expression for T_i^*, with respect to T_i at voxel i, including corrections for beam hardening and the inverse-square effect of diverging beams.

$$T_i^* = \left(\frac{D_{s,a}^2}{D_{s,i}^2}\right) H_i T_i$$

$$T_i = \sum_E \Psi_E \left(\frac{\mu_E}{\rho}\right) e^{-\left(\frac{\mu_E}{\rho}\right) d_i} \tag{2}$$

where $D_{s,a}$ is the distance from the source to the rotational axis (isocenter), and D_{sv} is the distance from the source to voxel i, in the direction of the beam's central axis. Through the beamlet-context extraction process, described in the following section, kernel tilting is implicit and a corrective term $(D_{s,a}^2/D_{s,i}^2)$ for the inverse-square effects of a diverging beam is applied directly to each T_i, H_i is a voxel depth- and tissue density-dependent factor based on an effective x-ray attenuation coefficient, which corrects for beam hardening effects. It is interpolated from a table of precomputed values specific to each beam spectrum and material. For this study, a single fluence-attenuation-table (FAT) was calculated and used for a 6MV bremsstrahlung x-ray spectrum in water. Because the utilized dose kernels are precomputed in homogeneous water, the energy-dependent mass attenuation (μ_E/ρ) coefficients are constant and equal to those of water. Thus, to correct for material inhomogeneity, a standard C/S technique[33] is used, whereby the kernel is instead warped according to the

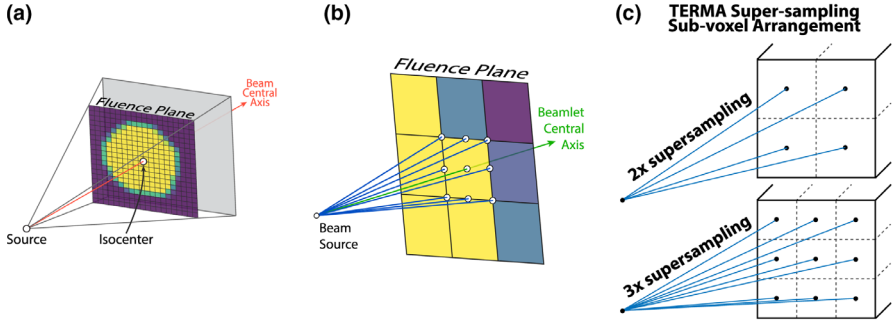

FIG. 1. (a) Intersection map for one beam orientation and target volume definition. Purple-colored cells have no target volume intersection and are excluded from beamlet dose calculation, yellow: full intersection, others: partial intersection. (b) Super-sampling ray layout for testing beamlet-target intersection. (c) cross-section of TERMA calculation sub-voxel arrangement for super-sampled averaging (2× and 3× options shown for one voxel).

material-dependent radiologic path length, d_i for each voxel (expressed in Eq.).

Anti-aliasing via uniform super-sampled averaging is employed during TERMA calculation with negligible cost to address aliasing (stair-stepping) otherwise observed along the beam and beamlet edges. Each voxel is divided into a set of k^3 sub-voxels, for the user-selected integer super-sampling level, k, as depicted by Fig. 1(c). The ray-based path length and TERMA calculations are performed for each sub-voxel, and the voxel's TERMA is assigned to the average of their values. The entire low-level process is presented for one compute node in Fig. 2(a), including the GPU modules and post-processing tasks. The distributed workflow in Fig. 2(b) is explained further in Section 2.A.2.

Beamlet-context extraction: The classical implementation of beamlet-based dose calculation treats each beamlet separately and performs dose calculation for each, one-by-one. While a functional solution, this approach results in a linear scaling of calculation times with the total number of beamlets as in Eq. 4.

$$\text{Calculation Time} \propto \sum_{b=1}^{B} n_b \quad \text{for} \begin{bmatrix} B : \#\text{ of beams} \\ n_b : \#\text{ beamlets in beams } b \end{bmatrix} \tag{3}$$

Our method instead reduces the time scaling factor to B by calculating individual dose for all beamlets in a beam at once. During dose calculation for one beamlet on the GPU, 3D dose spread is applied for every voxel in parallel. This approach to parallelizing the problem is sub-optimal because the random-access latency of globally stored data during convolution is high and the speed of the algorithm suffers. Trying to directly implement the NVB algorithm with support for beamlet-based dose calculation presents other difficulties such as introducing dose assignment race conditions[26] and inflating the memory footprint beyond feasibility with beamlet specific book-keeping. Attempting to store dose directly as a sparse array on the GPU to overcome memory consumption limits also introduces deleterious race conditions and memory access latencies since constant speed random-access of memory is no longer possible.

To circumvent these problems, we recognize that the dose resulting from common clinical x-ray spectra is spread locally around an interaction point with compact spatial support. For the purposes of radiation beam selection and fluence map optimization, a close approximation of the dose can be obtained by limiting the calculation of dose spread to the immediate neighborhood of each beamlet. This approximation known, as kernel or dose truncation, has been used previously for dose kernel generation[34] and simultaneous Monte Carlo beamlet dose simulation[35] to accelerate dose calculation. Utilizing this approximation, we construct a composite array of independent *beamlet contexts* (hereafter referred to as the *context array;* depicted in Fig. 3) that each contain only the density and TERMA data necessary for performing the CCCS convolution operations within its beamlet's confined surroundings. The long axis of each context is aligned in parallel to the long axis of its beamlet (called beamlets-eye-view; BEV) such that a minimum distance from any voxel of the beamlet to the boundaries of the context is enforced by the selected context radius, as described by Fig. 4. Aligning the contextual data in this way also implicitly corrects for beam divergence effects that would otherwise require costly kernel tilting to be individually applied for every beamlet. Since the contexts are independent and self-containing, their arrangement in the construction of the context array is unremarkable and therefore flexible. To construct each beamlet's context, we directly sample density from the global coordinate system, while mapping each voxel in the context to its corresponding global coordinate and directly calculating TERMA at this location using the method described in Section *TERMA calculation*. We ensure that only TERMA attributable to a context's beamlet is included

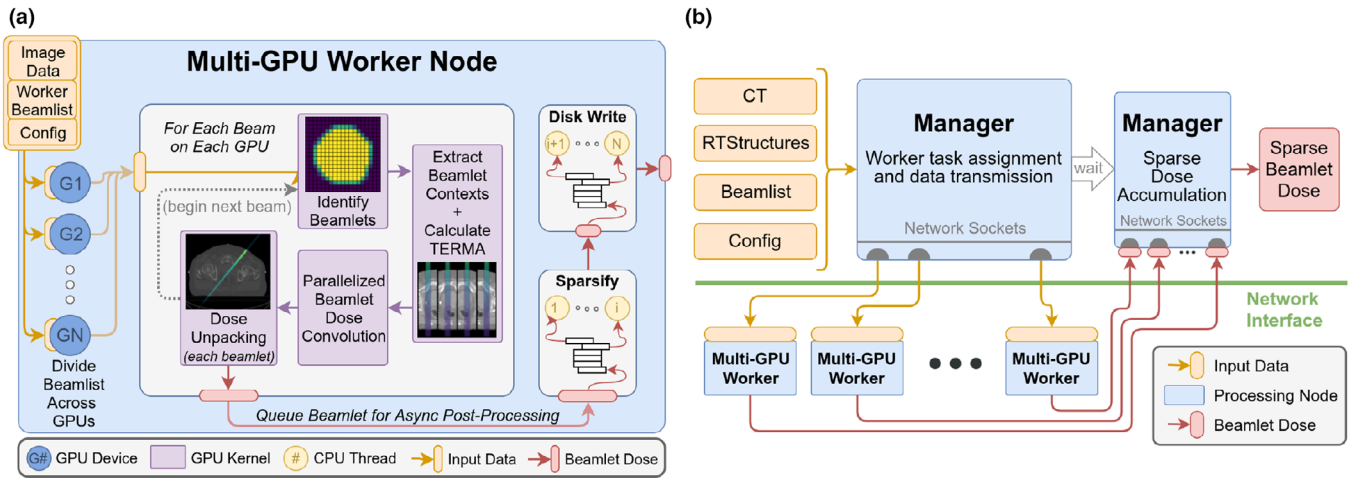

FIG. 2. (a) Beamlet dose calculation workflow. A single worker node processes beams in parallel across its resident graphical processing unit (GPU) devices. Beamlet processing is further parallelized on a GPU using beamlet contexts. (b) Distributed computing framework. The manager node prepares independent task lists for each worker node to process in parallel and receives the results for delivery to the requestor.

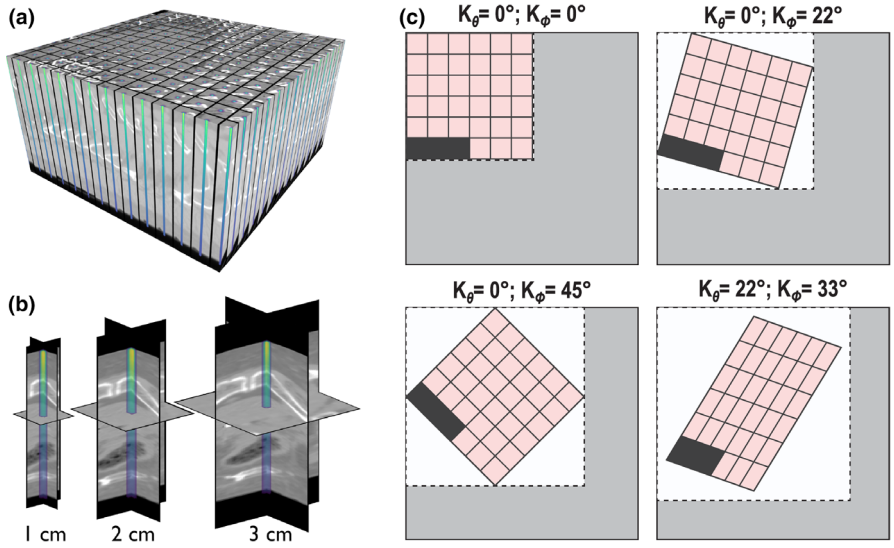

FIG. 3. (a) Visualization of the beamlet-context array for a single beam including contextual densities and beamlet-specific dose after calculation. (b) Beamlet-context cross sections for various context radii with dose overlaid. (c) Convolution ray-aligned context array (cross section) for various kernel rays. Gray area is allocated once and reused for all beams. White subregions are allotted for kernel ray-specific convolutions geometry. Black cells indicate unused space after packing beamlet contexts into the array. Convolution direction is into page.

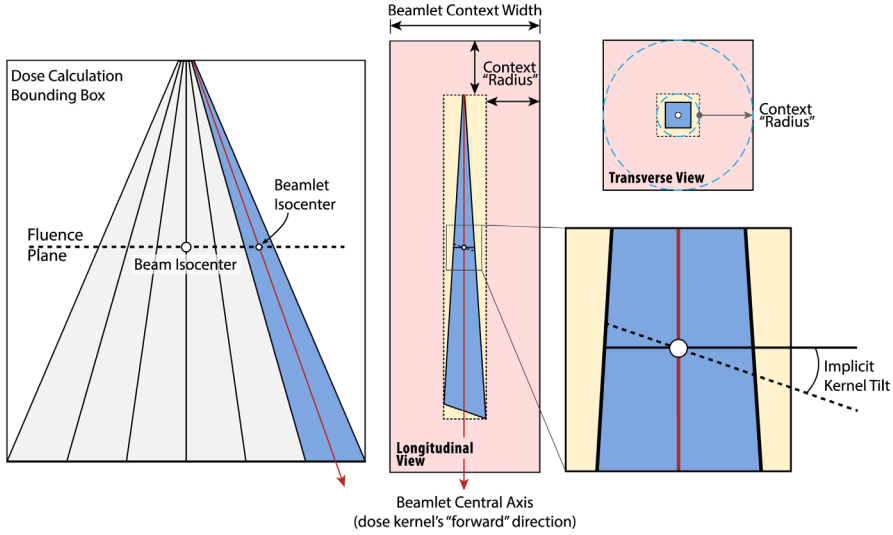

FIG. 4. Construction of one beamlet context with implicit kernel tilting. Blue region indicates the volume of nonzero TERMA for a single beamlet. The distance between the blue rings in the transverse view is representative of the context radius setting. The union of red and gold boxes represents the volume in which dose is computed.

in the context by projecting onto the fluence plane and testing membership in the fluence element corresponding to that beamlet using the procedure outlined previously.

Nonvoxel-based transformation: We combined our novel context transformation with the NVB algorithm[21] to calculate beamlet-specific dose in parallel. To understand the motivation behind the NVB algorithm, we briefly describe the structure of the Monte Carlo point spread kernels; a complete presentation can be found in the literature.[1,2,36] The kernels used in our approach contain coefficients calculated in a polar system of homogeneous water with 24 radial and 48 angular points. By collapsing the full cartesian sampling space into a set of radially divergent cones, the dose distribution may be closely approximated at a much lower computational cost. During convolution, the dose is only calculated for voxels intersecting the central axes of these cones rather than for a radial ray terminating at each of the original sampling points on the dose grid. This is the defining distinction of CCCS over more precise C/S algorithms.

One simple way to structure the CCCS algorithm on the CPU is to iterate over the volume, stopping at each voxel to spread dose to surrounding voxels before moving to the next iterate. In GPU computing, memory read and write latencies

usually present the greatest barrier to an efficient implementation. A direct translation of the CPU-based approach to GPU leads to substantial memory latency and introduces race conditions that force expensive synchronization of GPU threads to obtain correct results. Same as the NVB algorithm, we combat these issues by resampling (rotating) the context array along each kernel ray in turn, placing TERMA and density data into a new coordinate system referred to as the rays-eye-view (REV). We use tri-linear interpolation to cast the original density and TERMA data into the context-based REV and back into the global coordinate system. As such, each mapping is affine and performed on a continuous spatial domain. Furthermore, each mapping is invertible to the extent of the data retained after truncation by the selected context radius. After transformation, each row of the contextual data is arranged as a contiguous block of memory permitting its efficient access by coalesced GPU memory transactions during convolution. In doing so, we reduce the memory access overhead and amplify the benefits of GPU parallelization.

As there are many rays along which the dose will be spread during convolution with the dose kernel, the array of Fig. 3(a) is reconstructed in the REV specific to each convolution ray prior to dose calculation. The resulting dose is successively transformed into a fixed *common orientation,* which aligns every context central axis in parallel with the data column direction, for accumulation until dose for all rays has been calculated. To efficiently obtain beamlet-specific dose for every one of a beam's active beamlets in parallel, we maintain three arrays in the current REV coordinate system to store the contextual density, TERMA, and resulting dose. A fourth array is kept in the *common orientation* for accumulation of dose from each REV. GPU memory requirements for the context-based method are primarily determined by the sizes of these four arrays which change based on a number of user-controlled and geometry specific factors such as the number of active beamlets in each beam, and various quality settings (voxel size and context radius among others). To maximize computational efficiency, we predetermine these memory requirements for every beam before initializing the memory allocations. This allows us to instead allocate a single set of memory for these four arrays with sufficient size to fit all scenarios rather than repeatedly allocating and deallocating smaller memory segments and suffering from the significant overhead such CUDA API operations impose on overall runtime. The single allocation is represented by a gray box in Fig. 3(c), and convolution along each kernel ray uses its own subset of this memory (white box), dependent on the rotated geometry of the context array. To provide flexibility of our method to GPU hardware with lower available memory, we further implement optional *beamlet batching* which divides the complete context array for a beam into two or more sub-arrays to process successively. We allow explicit control over the number of batches for all beams when desired, and otherwise, dynamically detect when GPU memory restrictions necessitate batching for each beam on an individual basis.

Dose ray convolution: For each instance of the ray-specific REV-aligned context array, line convolution is carried out over the rows of the REV-aligned context array for every voxel along the kernel ray. By design, the density and TERMA accessed by the voxels in each row are restricted to the values coincident with each ray. These data are cached into shared memory for fast repeated access by neighboring voxels, offering hundreds of times less latency than global memory on average.[37] GPU thread race conditions are avoided by treating the CCCS operations from the "dose deposition point of view,"[1,26] enabling each thread to assign to its own voxel-specific memory address without conflicting with the data write operations of other threads. Each convolution is performed on a nonvoxel basis with linear interpolation using the cumulative kernel (CK) technique developed by Lu[38] and summarized by Neylon.[21] To obtain the full dose distribution for each beamlet, this line convolution procedure is performed along each kernel ray, and the dose from each is transformed into a common orientation and accumulated.

Beamlet-context dose extraction: The dense dose distribution attributed to each context's beamlet is stored in the context array in the *common orientation*. The selection of context radius determines the physical spatial extent to which the scattered dose is recorded. To represent this dose distribution in the original coordinate system, a beamlet specific affine transformation is applied to each context and the dose data are converted to a sparse representation in a two-column coordinate list (COO) format. One column contains the linearized volume index of each non-zero element, while the second column contains the corresponding value (dose). A threshold may be configured at this stage to exclude elements of negligible magnitude to further reduce storage size and improve data storage speeds. After conversion to the COO format, the dose data are written in a widely supported and flexible binary format (HDF5) to disk to be recalled and used during treatment planning.

2.A.2. Distributed parallelization

Additional high-level parallelization of the algorithm is achieved by embedding our context-based approach into a distributed multi-GPU framework. We harness the trivial separability of per-beam processing to build a network of computational workers, each of which may provide one or more GPUs. The division of labor among the worker nodes is simple and flexible with respect to the number available and is based on the computation of each beam as a standalone labor unit.

The beamlet dose calculation task is first executed on a managing node whose job is to prepare the static data (CT and configuration) and assign per-beam processing tasks to the workers. The manager node considers the availability of worker nodes and the number of GPUs provided before transferring the requisite data and task assignments to each. Upon receipt, the worker further assigns per-beam tasks to its resident GPUs which each take responsibility for one beam at a

time and run concurrently. Processing on each GPU proceeds as described in Section 2.A.1. The resulting beamlet dose data are immediately transferred over the network to the managing node for inclusion in the user-facing HDF5 file. The process flow detailing the distributed parallelization structure is described in Fig. 2(b).

2.B. Measuring computational efficiency

To quantify the performance of our context-based GPU-CCCS method for beamlet dose calculation, we measured and compared its computational efficiency against an existing GPU-CCCS implementation[6] that calculates beamlet dose in sequence. Beamlet doses for two representative 4π plans were calculated with isotropic 2 mm voxel sizes and the average calculation times for each beam were recorded. Each plan was composed of the same 1162 beam specifications distributed spherically around prostate and lung PTV definitions in two distinct CT geometries. The total number of beamlets for each plan was dependent upon the PTV shapes; 434 670 and 302 643 beamlets were calculated in total by each method in the prostate and lung targets, respectively. Both our context-based GPU-CCCS and the existing beamlet-sequential GPU-CCCS implementations shared CCCS quality settings that were set to match one another, such as the number of convolution rays ($N_\theta \times N_\phi$). The additional beamlet-context radius parameter of our method was tested at 1, 2 and 3 cm to demonstrate the flexibility provided in balancing speed and accuracy. The per-beamlet calculation bounding box margins for the sequential GPU-CCCS method were matched to the context radius to control for the effects of calculation over reduced volumes of different sizes when measuring the performance. N_θ and N_ϕ were set to 8×8 and 16×16 for both methods to quantify computational efficiency in these two common configurations. We also provide results for our method in distributed configurations with 1, 2, and 3 networked worker nodes, each employing two GPUs for a total workforce of 2, 4, and 6 GPUs, respectively. To compare peak GPU memory usage for each of the sequential and context-based GPU-CCCS methods, 20 random non-coplanar beam orientations were selected (10 from each of the prostate and lung CT geometries), and doses for rectangular fields composed of various quantities of 5×5 mm beamlets were calculated. Isotropic 2 mm voxels and 16×16 convolution rays were configured throughout testing, and the context radius of the context-based GPU-CCCS method was additionally varied between 0.3 and 3 cm. For each set of quality parameters, the same 20 beam orientations were processed in a single program execution and the peak GPU memory usage was recorded.

2.C. Measuring dosimetric accuracy

Accuracy comparisons between the beamlet-sequential GPU-CCCS algorithm[6] and the NVB algorithm[21] of which our method is an extension have already been analyzed and will not be repeated here. Instead, we provide an investigation of the accuracy of our context-based method against Monte Carlo and a reference CPU-CCCS implementation in two phantom geometries: one homogeneous water and one heterogeneous stack of slabs, each detailed in Fig. 5. Monte Carlo dose was obtained using Geant4 for a continuous emission spectrum of a diverging square monoenergetic photon beam, fit to the discrete spectrum used by CCCS. Monoenergetic dose kernels used in our context-based GPU-CCCS and the CPU-CCCS methods were previously synthesized[39] using an EGSnrc code, and the same emission spectrum was used to construct a polyenergetic kernel for dose convolution. Doses for 5 mm, 1 cm, and 2 cm wide beamlets were calculated. A voxel size of $1 \times 1 \times 1$ mm^3 was selected to compare the dose profiles of all beamlets more accurately. For the context-based GPU-CCCS method, the context radius was fixed at 4 cm. Central beamlet-axis percent depth dose (PDD) and lateral beamlet line profile at a depth of 10 cm were visualized along with the error of our method from the CPU-CCCS and Monte Carlo results.

To support the assumption that beamlet dose can be well estimated by calculating only within a limited interaction context, we varied the context radius parameter between 0.1 cm and 8 cm for each of the three previously tested beamlet widths in the water phantom. From these experiments, the lateral beamlet profile at 10 cm depth is included with the maximum volumetric error for each pairing of beamlet width and context radius compared to dose for the same beamlet without using the context-based approximation by the beamlet-sequential GPU-CCCS method.

We also constructed a 5×5 cm^2 broad beam by addition of context-based dose for 100 adjacent 5×5 mm^2 beamlets in the water phantom and compared the broad beam lateral dose profile to that of a broad beam composed of beamlet dose calculated without use of the context-based approximation. Lateral dose profiles were analyzed at depths of 5, 10, and 15 cm and the broad beam error associated with the context-based method was also reported.

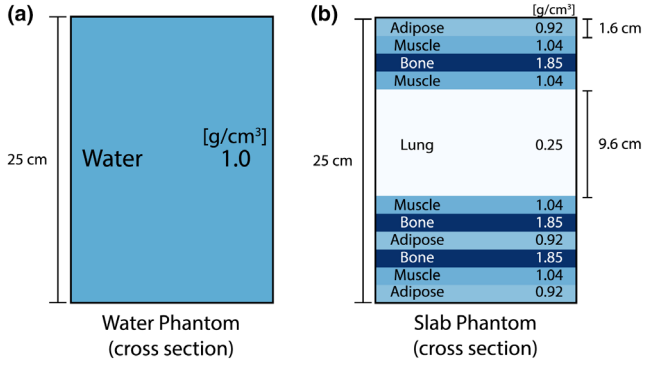

FIG. 5. Cross sections of phantom geometries with beam entering from the top; used to assess dosimetric accuracy. Materials and densities are provided.

Finally, the ability of our approach to scale to increased hardware availability and reducing the overall calculation time was investigated through timing experiments on multiple nodes and code execution profiling. Execution profile data were averaged across three independent application executions with 50 randomly selected 4π beams in each. All experiments were performed using NVIDIA GeForce TITAN X graphics cards from the Maxwell architecture. GPU programming was done using CUDA v9.0. For single-node performance evaluation, one node acted as both the manager and worker, employing an intel Xeon E5-2670 CPU with eight physical cores and a base clock speed of 2.6 GHz. For multi-node evaluation, workers having either an Intel i7-5820K CPU with six physical cores and a 3.3 GHz clock, or an Intel i7-7700K CPU with four physical cores and a 4.2 GHz clock were used. All data transfers between host and device memory were facilitated over 16-lane ($\times$16) PCIe 3.0 interfaces.

3. RESULTS

In Table I, we present the time required by each algorithm to calculate beamlet dose for one beam averaged over the set of 4π treatment beams.[6] Our framework is implemented such that we measured calculation time for a single GPU as well as in various distributed multi-GPU configurations, emulating simple deployment scenarios. Figure 6 shows how the performance of our approach scales in single-node and multi-node configurations as a function of the number of GPUs utilized. The colored dashed lines show the scaling performance in the single-node configuration, where GPUs are simply added to an existing compute node. Colored solid lines indicate performance gains when GPUs on additional worker nodes are introduced instead.

A decomposition of our algorithm into the fractions of time spent on each sub-procedure is given in Fig. 7 for various quality settings on a single node. Additional profiling results for use of various GPU counts on a single node are given in Fig. 8.

TABLE I. Per-beam calculation times (average, in seconds)

Treatment site	Prostate		Lung	
$N_\theta \times N_\phi$	8 × 8	16 × 16	8 × 8	16 × 16
Single node				
Sequential (1 GPU)	226.4	818.4	47.0	155.2
Context (1 GPU, 0.3 cm context)	2.362	2.799	1.686	2.016
(1 GPU, 1 cm)	3.066	6.322	2.312	4.958
(1 GPU, 2 cm)	5.159	13.731	3.385	9.142
(1 GPU, 3 cm)	9.119	29.765	5.343	17.137
Multi-node				
Context (1 × 2 GPU, 2 cm context)	3.130	11.977	1.964	7.466
(2 × 2 GPU, 2 cm)	1.643	6.288	1.031	3.919
(3 × 2 GPU, 2 cm)	**1.127**	**4.312**	**0.707**	**2.688**

Average per-beam dose calculation times (in seconds) for various hardware configurations, and quality settings.

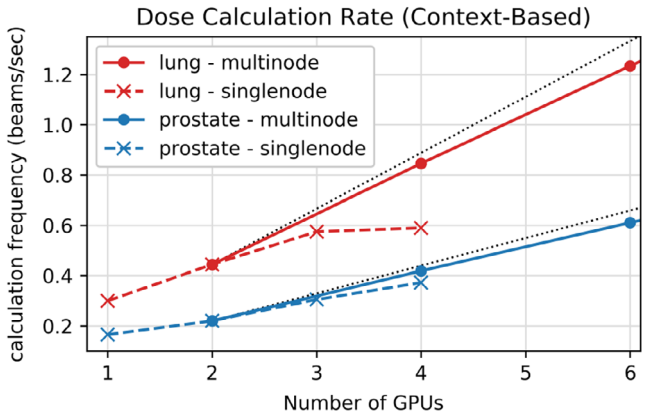

FIG. 6. Performance for single-node and multi-node scaling strategies. For multi-node measurements, each node was configured with two graphical processing units. The dotted black line indicates theoretical linear scaling in multi-node setups.

The memory usage recorded for both the sequential and context-based GPU-CCCS methods are listed in Table II. Figure 9 shows these results in graphical form.

Figures 10 and 11 give the PDD along the beamlet's central axis, the centered lateral line profile at 10 cm depth, and the error for each result compared with the CPU-CCCS and Monte Carlo methods. Results of our approach are given in color for each beamlet size in both the water and stacked slab phantoms. Monte Carlo outcomes are presented in gray, and CPU-CCCS dose is given as a dotted curve. Normalized error between our method and each of the Monte Carlo and CPU-CCCS methods are additionally given.

The maximum error of the context-based compared to noncontext-based beamlet dose resulting from various selections of context radius for the water phantom is given in Fig. 12 with the resulting lateral line profile at 10 cm depth. Absolute errors are expressed as percentages of the maximum reference volume dose calculated without the context-based approximation.

4. DISCUSSION

4.A. Performance

The performance improvements offered by our context-based method over the beamlet-sequential GPU-CCCS method are evident from Table I. When a 2-cm context radius is used on a single GPU for both methods, our approach offers 44–60$\times$ speedup and 14–17$\times$ speedup for the prostate and lung plans, respectively. These results demonstrate a clear efficiency advantage of our context-based processing. Even for a larger context radius of 3 cm, ours demonstrates 25–28$\times$ and 9$\times$ speedups over the beamlet-sequential GPU-CCCS method configured with its beamlet dose calculation margin matching the context radius; demonstrating pure acceleration without truncation-induced loss of dose fidelity. Analysis of the error reported in Fig. 12 shows that <1% beamlet-specific PDD error could be achieved in the water

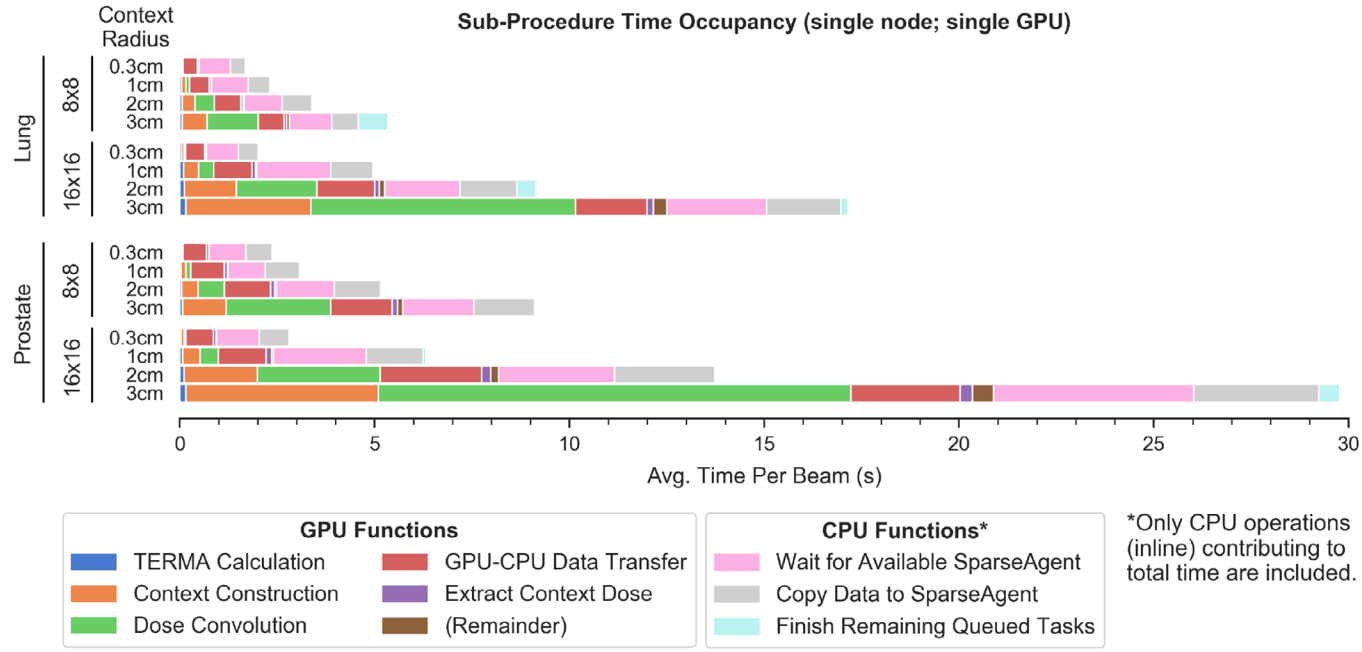

FIG. 7. Fractional execution time spent in each sub-procedure on one computational node with four threaded postprocessing "SparseAgents." Only time spent on the main processing thread is represented.

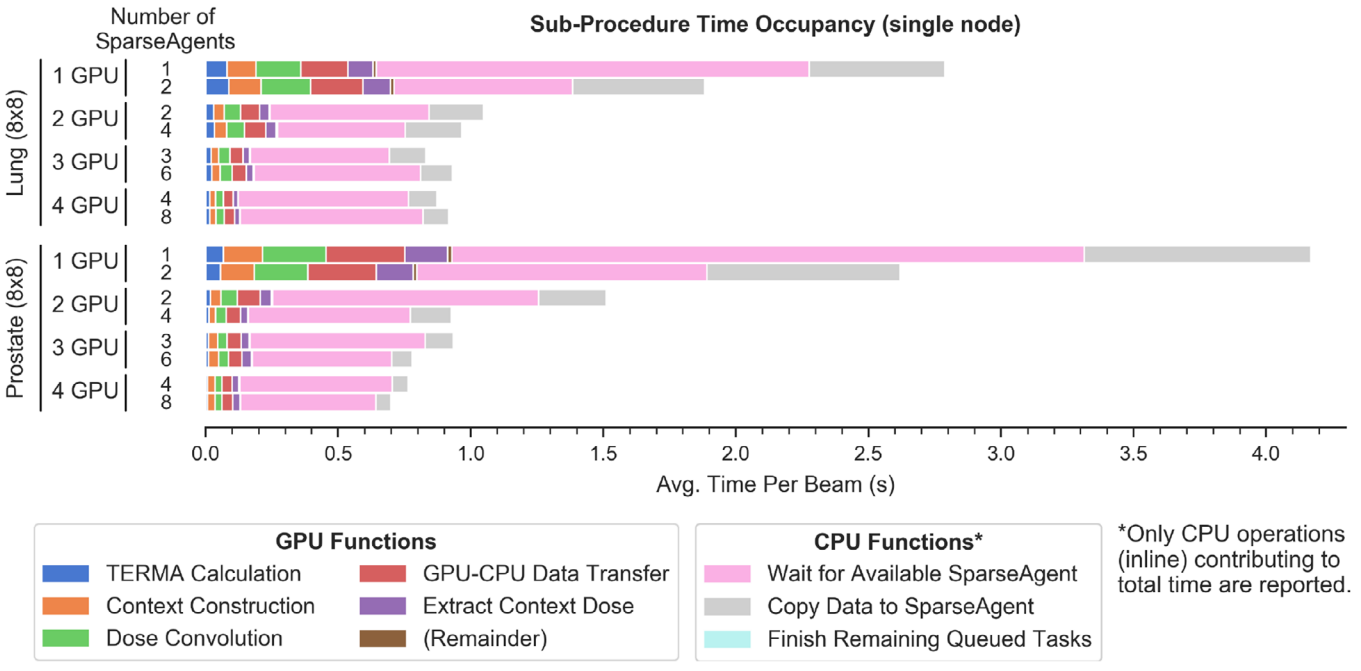

FIG. 8. Fractional execution time for 1-cm context radius with a variable number of background postprocessing (dose sparsification) threads. Only time spent on the main processing thread is represented.

phantom for the 5×5 mm^2 beamlet width by setting the context radius to just 3 mm. Targeting this beamlet dose error of <1%, we additionally timed our approach on the 4π dose calculation task using the reduced 3-mm context radius. In this test, our method demonstrated even greater single GPU acceleration rates of 95–292× and 28–77× compared to the beamlet-sequential GPU-CCCS baseline for the prostate and lung plans, respectively.

With its simplicity in scaling, our approach was also configured for multi-node calculation. When distributed across three workers employing two GPUs each, for a total of only six GPUs, we measured acceleration factors of 190–200× and 58–66× for the prostate and lung plans, respectively, using a 2-cm context radius. Applying the results of the single GPU experiments, we expect even greater accelerations for the dosimetrically similar 3 mm and 1 cm context radii.

TABLE II. Peak memory usage for graphical processing unit (GPU)-based CCCS methods (in Megabytes)

	Context-based GPU-CCCS (by context radius)				Sequential GPU-CCCS
	0.3 cm	1 cm	2 cm	3 cm	
50 beamlets	200.07	244.23	528.34	1053.16	959.03
100 beamlets	188.12	382.64	980.64	2136.38	973.84
150 beamlets	255.09	618.59	1774.54	4014.71	985.25
200 beamlets	323.60	864.74	2643.31	5634.15	1001.78

Graphical processing unit memory usage for sequential and context-based GPU-CCCS methods for $2 \times 2 \times 2$ mm^3 voxels and 16×16 convolution rays.

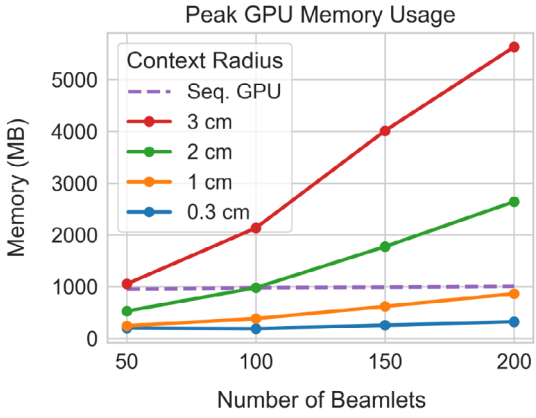

FIG. 9. Peak memory usage for various beamlet counts and context radii.

Figure 6 presents complete evidence of our framework's scalability, reaching nearly linear efficiency gains in the number of GPUs used in a multi-node configuration. Network latency did not contribute significant overhead in our testing. However, we believe multi-node scaling performance can be further enhanced by utilizing dedicated 10 Gigabit inter-node connections to increase the network communication bandwidth over the 1 Gigabit connections used during testing but will leave such confirmation for future work.

Looking more closely at Fig. 6, we observed that the single-node performance gains quickly plateaued beyond the use of three GPUs for the lung target. A performance bound is not observed for the prostate, since we did not have the resources to test the single-node configuration with more than four local GPUs. We hypothesized that as the number of GPUs increased on a node, the beamlet dose of multiple beams is computed much more quickly than it can be transferred to the host, converted to a sparse format, and stored to the hard drive (collectively referred to as *post-processing*). By limiting the number of GPUs on each device, and instead increasing the number of computational nodes, we spread the GPU computational resources across more CPUs and output disks, and better balance the computational speed with the postprocessing speeds. We tested this hypothesis for both treatment sites by distributing GPUs over more nodes and found that the multi-node configuration reduces the effects of the bottleneck, overcoming the undesired plateau of performance scaling seen in Fig. 6.

To confirm our hypothesis, moreover, we analyzed our algorithm using standard code-profiling techniques. Figure 7 indicates that the only sub-procedures with strongly dependent runtime contention as the context radius and quality of the dose increase are *Context Construction and Dose Convolution*, both implemented on GPU. This is expected since these functions are dependent on the size of the context array which is directly affected by manipulation of the context radius. Unlike the former two GPU operations, which are executed once for each convolution ray, the *Extract Context Dose* operation is executed once for every beamlet to transform each set of computed beamlet dose data from the context array (in the arbitrary *common orientation*, introduced in Section *Nonvoxel-based transformation*) to the original coordinate system. This is a simple transformation and is made efficient by coalesced GPU memory access from the context array.

Undesirable, however, is the observation that copying the dose data from the GPU to the host memory *(GPU-CPU Data Transfer)* follows a weakly scaling trend, indicating that even as the quality of the dose (context radius) is reduced, the total computation time approaches a lower bound, in part determined by the memory transfer bandwidth between the host and GPU device. The other, more dominant factor determining the efficiency bound is the speed of postprocessing (dose sparsification and storage). Since these tasks place postprocessing requests in a fixed-size queue, and are handled by a team of SparseAgents in separate CPU threads, we only see these operations contribute to the total runtime (*CPU Functions*) when the GPU outpaces the CPU. When this occurs, GPU computation is paused to limit the host memory usage while the postprocessing queue is sufficiently depleted. The combination of the *Wait for Available SparseAgent* and *Copy Data to SparseAgent* operations indicate the amount of time that the main processing thread must wait while the occupancy of the postprocessing queue is reduced. This type of delay is most pronounced when many GPUs are available on each node. This limit is demonstrated in Fig. 8 where we see that nearly every GPU operation shortens in aggregate as more GPUs are added to a node, while the inline postprocessing operations initially shorten as more threaded SparseAgents are provisioned but quickly exhibit diminishing returns; further supporting our hypothesis that the single-node algorithm performance is bounded by the postprocessing time consumed on each worker. This time is in turn dominated by CPU core availability, as well as hard drive write and network transfer speeds, that can potentially be alleviated by increasing network transfer bandwidth and distributing GPU resources over more computational nodes, as suggested.

Using the memory usage results, reported in Table II and Fig. 9, we show that the primary factors determining the GPU memory usage of the context-based GPU-CCCS method are the number of beamlets in each beam, and the selected context radius, in addition to the general considerations such as voxel size and patient size (determining the beamlet length) common to all CCCS methods. We observed that for some combinations of beamlet count and context

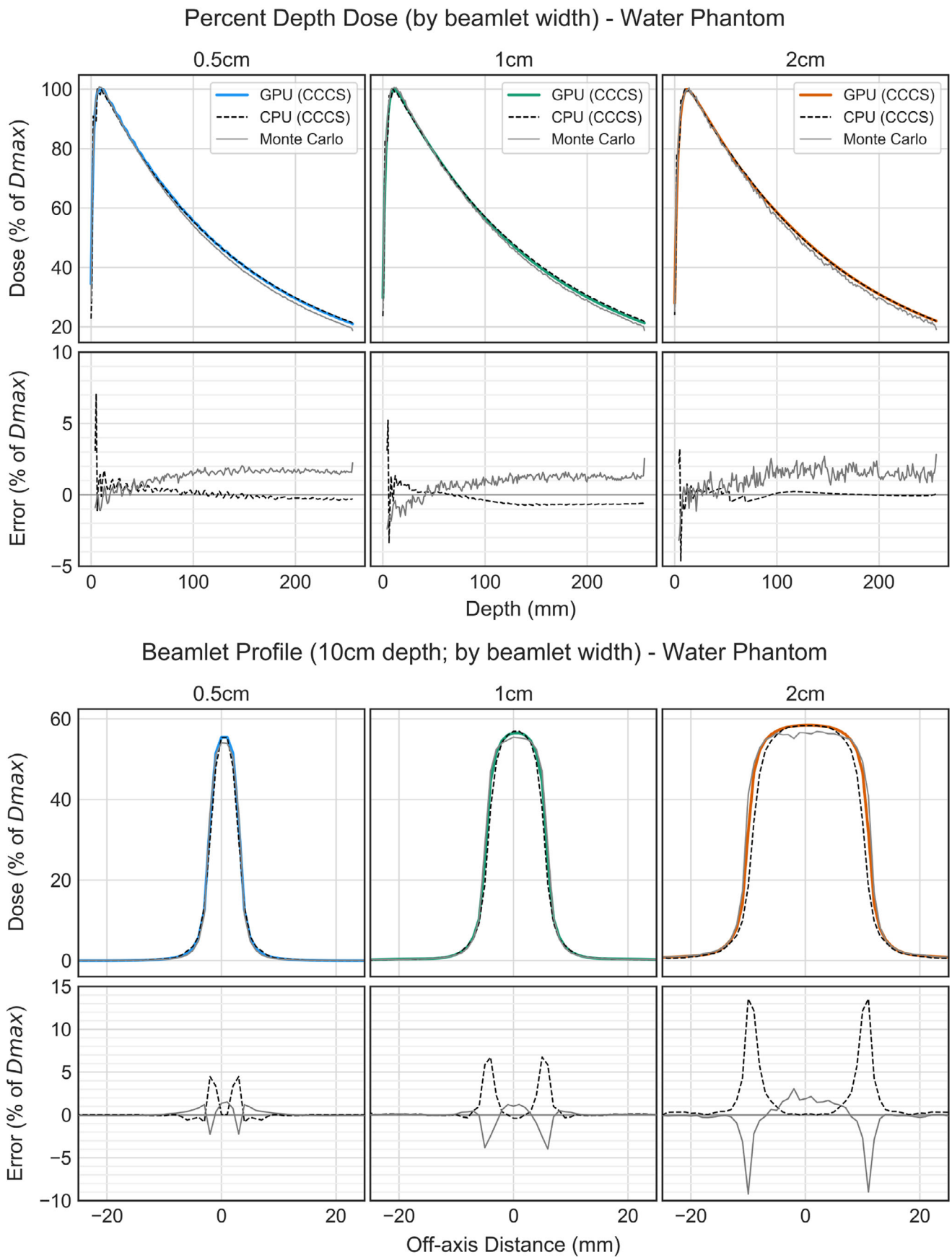

FIG. 10. Single-beamlet depth dose and lateral profiles in the water phantom for increasing beamlet widths. Error is calculated between our context-based GPU-CCCS method and each of CPU-CCCS and Monte Carlo.

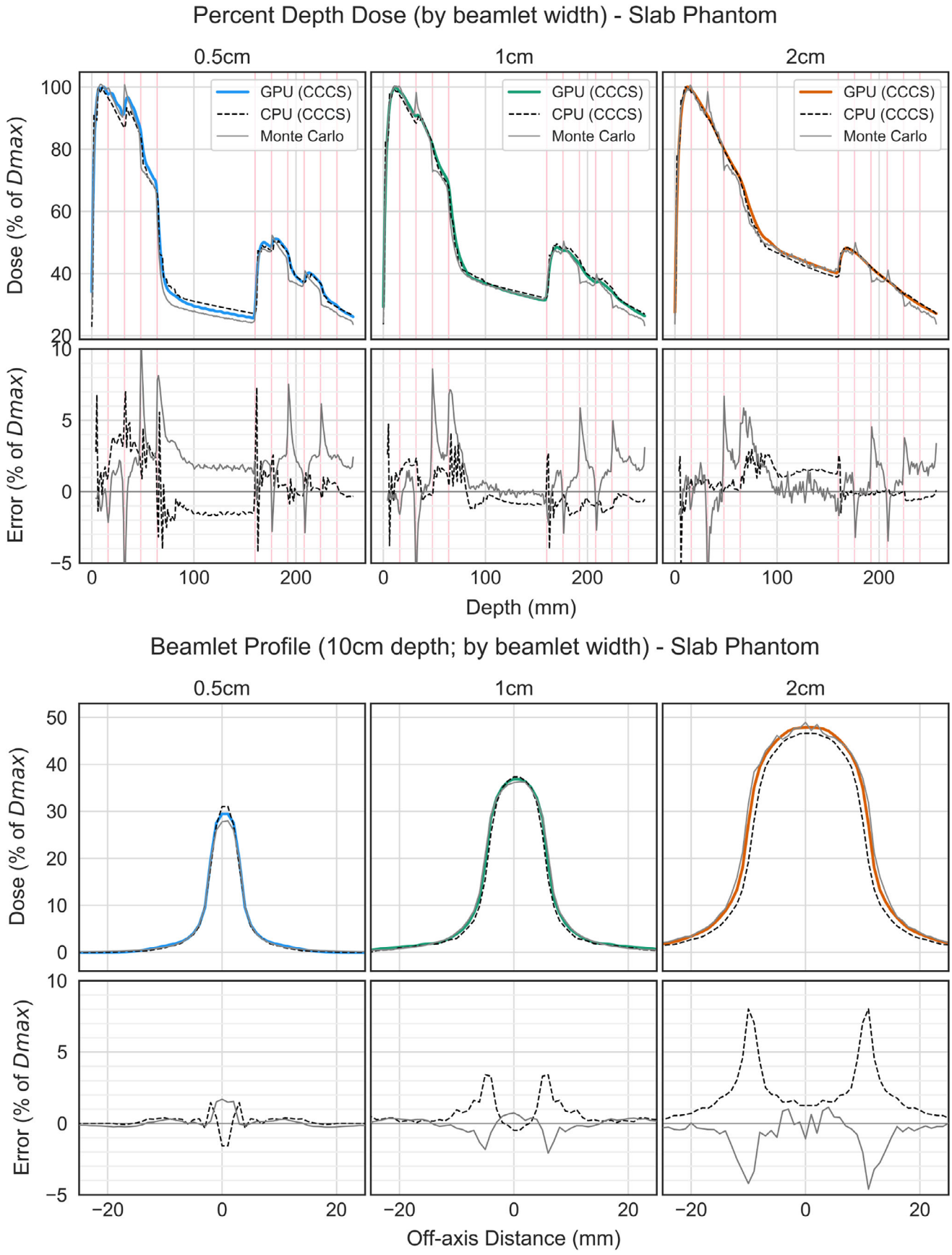

FIG. 11. Single-beamlet depth dose and lateral profiles in the stack of slabs phantom for increasing beamlet widths. Error is calculated between our context-based GPU-CCCS method and each of CPU-CCCS and Monte Carlo.

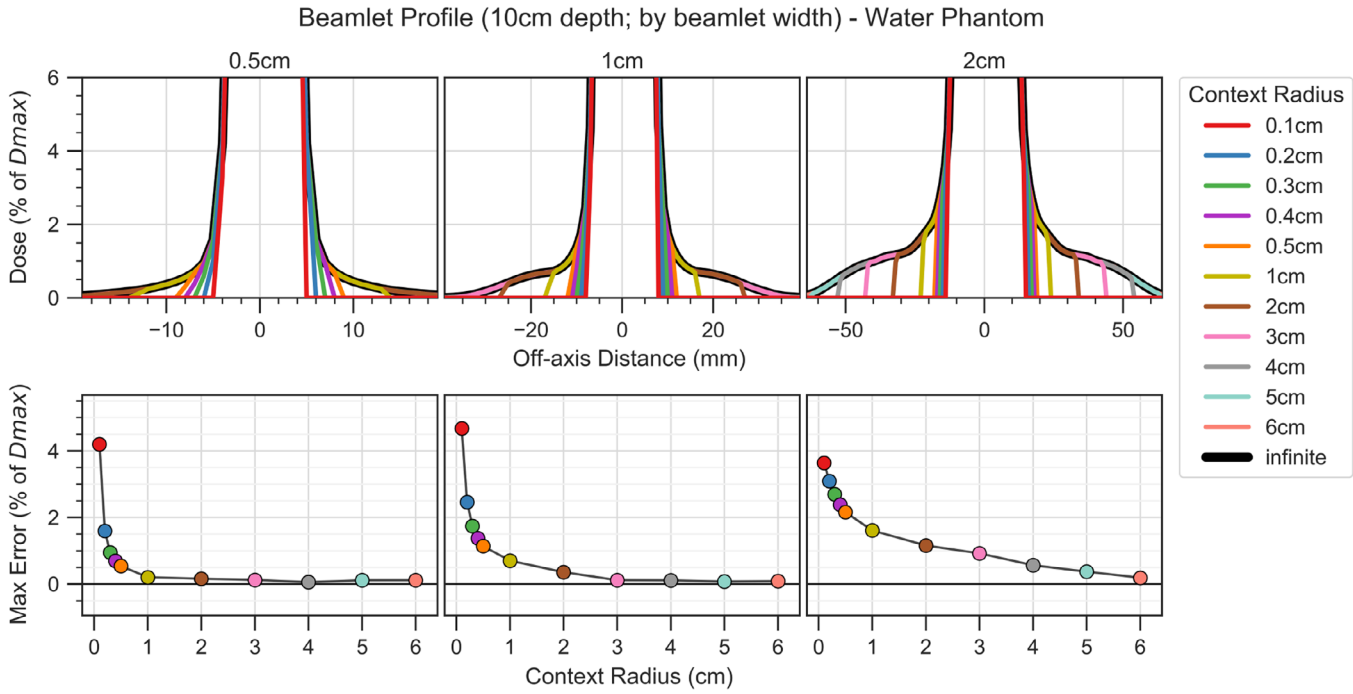

FIG. 12. Central lateral line profile in the water phantom at 10 cm depth for various beamlet widths and context radii pairs (top). Maximum errors (%) between noncontext-based (infinite radius) and context-based dose profiles are provided (bottom). The dose is normalized to the maximum dose in the volume. *Y*-axis range is limited to better depict low-dose beamlet penumbra region where context-based approximation is active.

radius, the peak GPU memory usage was similar. Further investigation of these cases confirms our intuition that for each, a context array [Fig. 3(a)] of similar size and shape was constructed. As expected, the peak memory usage of the sequential GPU-CCCS algorithm shows no impactful dependence on the number of beamlets in each beam, due to the sequential calculation of beamlet dose inherent to the technique. A weak correlation was observed but is insignificant and likely caused by changes in bookkeeping and the geometry of the calculated beamlets as the field size changes and intersects with different volumes of the CT. As described in Section 2.A.1, we have developed the context-based GPU-CCCS method with optional dynamic beamlet batching to alleviate high memory usage concerns for cases such as that with 200 beamlets and 3 cm contexts, showing peak usage of 5.6GB. With this feature, we hope to increase compatibility with budget-friendly GPUs providing less total memory.

4.B. Accuracy

Like the NVB algorithm on which we have based the core of our algorithm, calculated dose closely agrees with the CPU-CCCS calculated dose in the water phantom (Fig. 10), with maximum single-beamlet PDD errors of 2% beyond the high-dose gradient region found in the first few millimeters of the phantoms. Single-beamlet lateral dose profile errors in the water phantom are greatest at the beamlet edges where high-dose gradients are again observed. Inspection of the beamlet profile errors in Fig. 10 indicates that the context-based method consistently displays smaller error in these regions when compared to Monte Carlo dose than when compared to CPU-CCCS dose, likely due to the use of TERMA super-sampling that has been employed in the context-based method to this effect. Errors in profile dose in the primary portion of the beamlets are below 2% on average between context-based and Monte Carlo methods for all beamlets sizes. This small error results from slight depth-dependent difference seen in the beamlet PDDs in the water phantom geometry (Fig. 10), likely caused by the use of a continuous beam spectrum in Monte Carlo simulation rather than a discrete spectrum as in CCCS.

Single-beamlet dosimetric errors observed in the slab phantom (Fig. 11) are slightly larger overall than those found in the water phantom. The greatest deviations of the context-based method from Monte Carlo beamlet dose occurs after interfaces between media of substantially different densities (particularly at depths of 32, 64, and 160 mm), an effect attributable to the well-known shortcomings of the heterogeneity correction used in the CCCS method that have already been independently investigated.[40–42] Closer agreement of our context-based GPU-CCCS method with the reference CPU-CCCS implementation at these interfaces support this explanation. Despite these inherent shortcomings in the CCCS algorithm, the context-based method shows average single-beamlet PDD errors of magnitude <1.35% and 2.35% for all beamlet sizes in the water and slab phantoms, respectively.

The dose truncation effects of our context-based approach are evident in Fig. 12 where its resemblance to cylindrical kernel truncation[34] in the lateral direction is clear. In line with

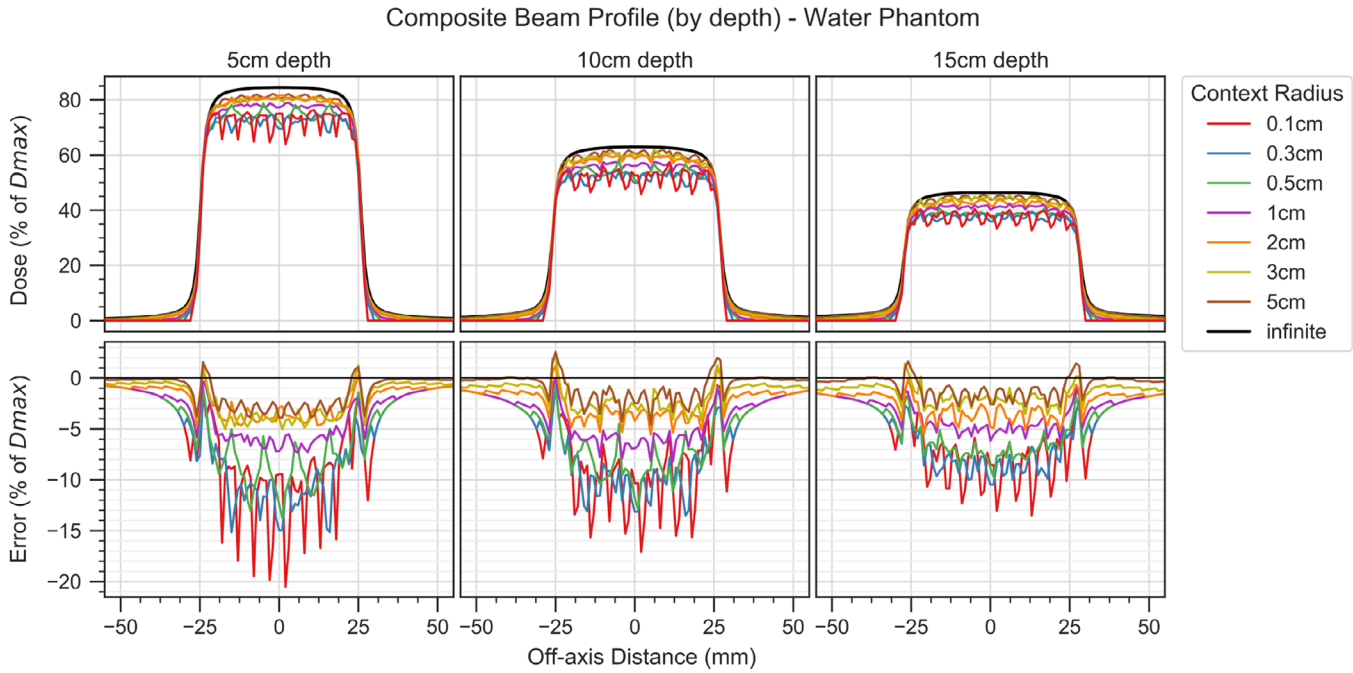

FIG. 13. Lateral line profile in the water phantom at various depths for a 5 × 5 cm^2 broad beam calculated as the sum of 5 × 5 mm^2 beamlets for various context radii at three depths. Dose for "infinite" radius was computed without context-based approximation.

our expectation, the maximum single-beamlet profile error decreases quickly as the context radius is increased. The rate of decrease in the error is smaller as the beamlet size is made larger, because more dose is physically scattered outside of the primary beam, as observed by the longer tails of the 2-cm wide beamlet compared to the 0.5 cm beamlet. The profile error for the 5 × 5 cm^2 wide broad beam, composed as a sum of 100 5 × 5 mm^2 wide beamlets, presented in Fig. 13, shows that full beam dose profile errors below 5% and 10% can be expected for context radii above 2 and 1 cm, respectively. The nature of the context-based method makes it difficult to directly truncate the polyenergetic dose kernel and renormalize its remaining weights to sum to 1, and thus, energy is not strictly conserved in the current implementation. We instead recommend the intuitive use of a small context radius when the beam candidate pool is large, such as in early stage of automatic beam orientation optimization which considers over 1000 beams. Approximate dose is often sufficient for ruling out trivially unsuitable beam orientations and the context radius can be increased to recompute more accurate beamlet dose once the beam candidate pool has been reduced.

Our current implementation relies on the user-defined context radius which corresponds to a physical path length from the edge of a beamlet. We have also considered dynamically setting the context radius of each beamlet in response to local density patterns. By doing so, beamlets in homogeneous high-density environments would be assigned low radii to match the short radiological path lengths, whereas those in low density or heterogeneous environments would be assigned higher radii. This adaptive approach would allow more optimal allocation of computational resources to beamlets where distant dose scatter is expected; this work, however, has been left for future investigation.

5. CONCLUSIONS

We developed and implemented a highly efficient GPU-CCCS algorithm for computing beamlet dose with customizable fidelity using an intuitive *context radius* setting for high beam counts in complex static-beam and dynamic arc IMRT planning problems. We have demonstrated that the use of our novel parallel beamlet-context based technique substantially outperforms the naive approach of computing beamlet dose in sequence as is done by existing CCCS algorithms in terms of efficiency, while maintaining similar levels of dosimetric accuracy. Additionally, we embedded this approach in a scalable high-performance computing architecture that allows the number of independent computing nodes, and the number of GPUs employed by each to be adapted to match the resources and demands of the user.

ACKNOWLEDGMENTS

The authors would like to thank Qihui Lyu, Kaley Woods, Angelia Landers, and Daili Shang for their assistance with testing the proposed framework and for the valuable feedback that was provided in turn. This research is supported by DOE Grants Nos. DE-SC0017057 and DE-SC0017687, NIH Grants Nos. R44CA183390, R43CA183390 and R01CA188300.

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

REFERENCES

1. Mackie TR, Scrimger JW, Battista JJ. A convolution method of calculating dose for 15-MV x rays. *Med Phys*. 1984;12:188–196.
2. Mackie TR, Bielajew AF, Rogers DWO, Battista JJ. Generation of photon energy deposition kernels using the Monte Carlo code. *Phys Med Biol*. 1988;33:1–20.
3. Sievinen J, Ulmer W, Kaissl W. AAA photon dose calculation model in Eclipse. *Varian Med Syst*. 2005; 1:1–23. http://www.rtsalon.cn/upload/RTsalon_p_3218_2.pdf
4. Hasenbalg F, Neuenschwander H, Mini R, Born EJ. Collapsed cone convolution and analytical anisotropic algorithm dose calculations compared to VMC++ Monte Carlo simulations in clinical cases. *Phys Med Biol*. 2007;52:3679–3691.
5. Lyu Q, Yu VY, Ruan D, Neph R, O'Connor D, Sheng K. A novel optimization framework for VMAT with dynamic gantry couch rotation. *Phys Med Biol*. 2018;63:125013.
6. Dong P, Lee P, Ruan D, et al. 4π non-coplanar liver SBRT: a novel delivery technique. *Int J Radiat Oncol Biol Phys*. 2013;85:1360–1366.
7. Yu VY, Landers A, Woods K, et al. A prospective 4π radiation therapy clinical study in recurrent high-grade glioma patients. *Int J Radiat Oncol Biol Phys*. 2018;101:144–151.
8. Tran A, Zhang J, Woods K, et al. Treatment planning comparison of IMPT, VMAT and 4π radiotherapy for prostate cases. *Radiat Oncol*. 2017;12:10.
9. Smyth G, Evans PM, Bamber JC, et al. Non-coplanar trajectories to improve organ at risk sparing in volumetric modulated arc therapy for primary brain tumors. *Radiother Oncol*. 2016;121:124–131.
10. Panet-Raymond V, Ansbacher W, Zavgorodni S, et al. Coplanar versus noncoplanar intensity-modulated radiation therapy (IMRT) and volumetric-modulated arc therapy (VMAT) treatment planning for fronto-temporal high-grade glioma. *J Appl Clin Med Phys*. 2012;13:44–53.
11. Hsieh C-H, Liu C-Y, Shueng P-W, et al. Comparison of coplanar and noncoplanar intensity-modulated radiation therapy and helical tomotherapy for hepatocellular carcinoma. *Radiat Oncol*. 2010;5:40.
12. Woods K, Lee P, Kaprealian T, Yang I, Sheng K. Cochlea-sparing acoustic neuroma treatment with 4π radiation therapy. *Adv Radiat Oncol*. 2018;3:100–107.
13. Amit G, Purdie TG, Levinshtein a, et al. Automatic learning-based selection of beam angles in radiation therapy of lung cancer. In: *2014 IEEE 11th International Symposium on Biomedical Imaging (ISBI)*. Beijing, China: IEEE; 2014: 230–233. https://doi.org/10.1109/isbi.2014.6867851
14. Bangert M, Unkelbach J. Accelerated iterative beam angle selection in IMRT. *Med Phys*. 2016;43:1073–1082.
15. Li Y, Yao J, Yao D. Automatic beam angle selection in IMRT planning using genetic algorithm. *Phys Med Biol*. 2004;49:1915–1932.
16. O'Connor D, Voronenko Y, Nguyen D, Yin W, Sheng K. Fast non-coplanar beam orientation optimization based on group sparsity. 2017; 1–11.
17. Smyth G, Bamber JC, Evans PM, Bedford JL. Trajectory optimization for dynamic couch rotation during volumetric modulated arc radiotherapy. *Phys Med Biol*. 2013;58:8163–8177.
18. Lyu Q, Connor DO, Ruan D, Yu V, Nguyen D, Sheng K. VMAT optimization with dynamic collimator rotation. *Med Phys*. 2018;45:2399–2410.
19. Chen Q, Chen M, Lu W. Ultrafast convolution/superposition using tabulated and exponential kernels on GPU. *Med Phys*. 2011;38:1150–1161.
20. Chen Q, Lu W, Chen Y, Chen M, Henderson D, Sterpin E. Validation of GPU based TomoTherapy dose calculation engine. *Med Phys*. 2012;39:1877–1886.
21. Neylon J, Sheng K, Yu V, et al. A nonvoxel-based dose convolution/superposition algorithm optimized for scalable GPU architectures. *Med Phys*. 2014;41:101711.
22. Tian Z, Shi F, Folkerts M, Qin N, Jiang SB, Jia X. A GPU OpenCL based cross-platform Monte Carlo dose calculation engine (goMC). *Phys Med Biol*. 2015;60:7419–7435.
23. Ziegenhein P, Kozin IN, Kamerling CP. Towards real-time photon Monte Carlo dose calculation in the cloud Towards real-time photon Monte Carlo dose calculation in the cloud. *Phys Med Biol*. 2017;62:4375–4389. (Iccr 2016).
24. Park JC, Li JG, Arhjoul L, et al. Adaptive beamlet-based finite-size pencil beam dose calculation for independent verification of IMRT and VMAT. *Med Phys*. 2015;42:1836–1850.
25. Cho N, Tsiamas P, Velarde E, et al. Validation of GPU-accelerated superposition-convolution dose computations for the Small Animal Radiation Research Platform. *Med Phys*. 2018;45:2252–2265.
26. Hissoiny S, Ozell B, Despŕs P. A convolution-superposition dose calculation engine for GPUs. *Med Phys*. 2010;37:1029–1037.
27. Hissoiny S, Ozell B, Després P. Fast convolution-superposition dose calculation on graphics hardware. *Med Phys*. 2009;36:1998–2005.
28. Jacques R, Taylor R, Wong J, McNutt T. Towards real-time radiation therapy: GPU accelerated superposition/convolution. *Comput Methods Programs Biomed*. 2010;98:285–292.
29. Lu W. A non-voxel-based broad-beam (NVBB) framework for IMRT treatment planning. *Phys Med Biol*. 2010;55:7175–7210.
30. Piotrowski T, Skośka M, Jodda A, et al. Tomotherapy - a different way of dose delivery in radiotherapy. *Wspolczesna Onkol*. 2012;16:16–25.
31. Jacobs F, Sunderman E, Sutter B, Christiaens M, Lemahieu I. A fast algorithm to calculate the exact radiological path through a pixel or voxel space. *J Comput Inf Technol*. 1998;6:89–94.
32. Siddon RL. Fast calculation of the exact radiological path for a three-dimensional CT array. *Med Phys*. 1984;12:252–255.
33. Ahnesjö A, Aspradakis MM, Ahnesj A, Aspradakis MM. Dose calculations for external photon beams in radiotherapy Dose calculations for external photon beams in radiotherapy. *Phys Med Biol*. 1999;44:99–155.
34. Zhong H, Chetty IJ. Generation of a novel phase-space-based cylindrical dose kernel for IMRT optimization. *Med Phys*. 2012;39:2518–2523.
35. Li Y, Tian Z, Shi F, et al. A new Monte Carlo-based treatment plan optimization approach for intensity modulated radiation therapy. *Phys Med Biol*. 2015;60:2903–2919.
36. Ahnesjö A. Collapsed cone convolution of radiant energy for photon dose calculation in heterogeneous media. *Med Phys*. 1989;16:577–592.
37. Kirk DB, Hwu WW. *Programming Massively Parallel Processors*, 2nd edn. Burlington, MA: Elsevier, Inc.; 2010.
38. Lu W, Olivera GH, Chen M, Reckwerdt PJ, Mackie TR. Accurate convolution/superposition for multi-resolution dose calculation using cumulative. *Phys Med Biol*. 2005;655:655–680.
39. Hoban PW, Murray DC, Round WH. Photon beam convolution using polyenergetic energy deposition kernels. *Phys Med Biol*. 1994;39:669–685.
40. Tillikainen L, Helminen H, Torsti T, et al. A 3D pencil-beam-based superposition algorithm for photon dose calculation in heterogeneous media. *Phys Med Biol*. 2008;53:3821–3839.
41. Arnfield MR, Siantar CH, Siebers J, Garmon P, Cox L, Mohan R. The impact of electron transport on the accuracy of computed dose. *Med Phys*. 2000;27:1266–1274.
42. Zhen H, Hrycushko B, Lee H, et al. Dosimetric comparison of Acuros XB with collapsed cone convolution/superposition and anisotropic analytic algorithm for stereotactic ablative radiotherapy of thoracic spinal metastases. *J Appl Clin Med Phys*. 2015;16:181–192.