GPU-accelerated Monte Carlo method for dose calculation of mesh-type computational phantoms

Authors: Yan, Shuchang, Qiu, Rui, Luo, Xiyu, Hu, Ankang, Wu, Zhen, Li, Junli, Qiu, Rui

Date: 2025-07-25T15:45:59+00:00

Abstract

Computational phantoms play an essential role in radiation dosimetry and health physics. Although mesh-type phantoms offer a high resolution and adjustability, their use in dose calculations is limited by their slow computational speed. Progress in heterogeneous computing has allowed for substantial acceleration in the computation of mesh-type phantoms by utilizing hardware accelerators. In this study, a GPU-accelerated Monte Carlo method was developed to expedite the dose calculation for mesh-type computational phantoms. This involved designing and implementing the entire procedural flow of a GPU-accelerated Monte Carlo program. We employed acceleration structures to process the mesh-type phantom, optimized the traversal methodology, and achieved a flattened structure to overcome the limitations of GPU stack depths. Particle transport methods were realized within the mesh-type phantom, encompassing particle location and intersection techniques. In response to typical external irradiation scenarios, we utilized Geant4 along with the GPU program and its CPU serial code for dose calculations, assessing both computational accuracy and efficiency. In comparison with the benchmark simulated using Geant4 on the CPU using one thread, the relative differences in the organ dose calculated by the GPU program predominantly lay within a margin of 5%, whereas the computational time was reduced by a factor ranging from 120 to 2700. To the best of our knowledge, this study achieved a GPU-accelerated dose calculation method for mesh-type phantoms for the first time, reducing the computational time from hours to seconds per simulation of ten million particles and offering a swift and precise Monte Carlo method for dose calculation in mesh-type computational phantoms.

Full Text

Preamble

GPU-Accelerated Monte Carlo Method for Dose Calculation of Mesh-Type Computational Phantoms

Shu-Chang Yan[12], Rui Qiu[12]†, Xi-Yu Luo[12], An-Kang Hu[12], Zhen Wu[123], and Jun-Li Li[12]

[1]Department of Engineering Physics, Tsinghua University, Beijing 100084, China
[2]Key Laboratory of Particle & Radiation Imaging, Tsinghua University, Ministry of Education, Beijing 100084, China
[3]Nuctech Company Limited, Beijing 100084, China

Computational phantoms play an essential role in radiation dosimetry and health physics. Although mesh-type phantoms offer high resolution and adjustability, their use in dose calculation is limited by slow computational speeds. Recent advances in heterogeneous computing enable substantial acceleration of mesh-type phantom computations through hardware accelerators. This study develops a GPU-accelerated Monte Carlo method to accelerate dose calculation for mesh-type computational phantoms, involving the design and implementation of a complete GPU-accelerated Monte Carlo program workflow. We employ acceleration structures for processing mesh-type phantoms, optimize traversal methodologies, and implement a flattened structure to overcome GPU stack depth limitations. Particle transport methods are realized within mesh-type phantoms, encompassing particle location and intersection techniques. For typical external irradiation scenarios, we utilize Geant4 alongside our GPU program and its CPU serial counterpart for dose calculations, assessing both computational accuracy and efficiency. Compared to the benchmark simulated through Geant4 on CPU using a single thread, the relative differences in organ doses calculated by the GPU program predominantly remain within a 5% margin, while computational time is reduced by factors ranging from 120 to 2700. To the best of our knowledge, this study achieves GPU-accelerated dose calculation for mesh-type phantoms for the first time, reducing computational time from hours to seconds per simulation of ten million particles, thereby offering a swift and precise Monte Carlo method for dose calculation in mesh-type computational phantoms.

Keywords: GPU Monte Carlo, Mesh-type phantom, External exposure, Heterogeneous computing

Introduction

Computational phantoms are crucial in radiation dosimetry and health physics, demonstrating significant value in radiation therapy, accidental radiation events, and occupational exposure assessment [1–7]. Voxel-type and mesh-type phantoms represent the primary models used in current applications [7–10]. Compared to voxel-type models constructed from uniformly sized cubic voxels, mesh-

type models—the latest generation of anatomical models—are represented by boundary surfaces or meshes, offering both pose adaptability and high resolution [8]. This enables precise definition of human tissue and organ contours with arbitrary precision while maintaining smooth surfaces [11, 12], thereby overcoming limitations commonly observed in voxel-type models such as stair-stepped surfaces, resolution restrictions, and adaptability issues [3]. Consequently, mesh-type phantoms exhibit substantial potential in radiation therapy, radiation protection, and individual dosimetry assessment, where more accurate anthropomorphic models are required for obtaining precise individual doses [13, 14].

The use of computational phantoms in Monte Carlo simulations represents a crucial method for assessing human dose [7, 9, 15]. In Monte Carlo simulations, determining the relationship between particles and surrounding geometry is critical [16, 17]. When mesh models are directly utilized in computational simulations, aligning particle positions with all facets significantly decreases computational speed. Research conducted at Hanyang University in South Korea demonstrated that mesh models required 70 to 150 times more computational time compared to voxel models [18]. In 2014, Professor Chan Hyeong Kim and colleagues developed a method to accelerate mesh phantom computations by employing tetrahedral decomposition [19]. This approach significantly reduces computation time by enabling particle transport to be determined by comparing the particle' s position with only the four faces of a tetrahedron rather than the entire mesh, thereby achieving speeds comparable to voxel model geometries. However, in fields with stringent time constraints, such as clinical radiation therapy, nuclear medicine, and accident dose reconstruction, the computational time for tetrahedron models still markedly exceeds acceptable limits, thus restricting their applicability [20, 21].

GPU acceleration is a method employed to expedite Monte Carlo simulations, enhancing computational efficiency [17, 22, 23]. In recent years, GPUs have experienced rapid development. Compared to CPUs, GPUs exhibit superior single-precision floating-point computational capabilities and memory bandwidth, while GPU hardware systems offer enhanced ease of maintenance at lower costs. Since 2010, various GPU-accelerated photon transport and photon-electron coupled transport programs, such as gDPM [24], gSMC [25], and FRED [26], have been developed either by building upon existing fast Monte Carlo frameworks or creating entirely new implementations, with extensive verification and efficiency assessments. Nonetheless, to the best of our knowledge, no existing GPU program has yet achieved GPU acceleration based on mesh-type models. The primary challenges stem from the complexity of implementing rapid particle transport within tetrahedral mesh-type models on GPUs, while the limited stack depth of GPUs complicates the execution of recursive algorithms necessary for navigating these intricate structures, further hindering performance [27].

To address the deficiencies in Monte Carlo simulations for mesh-type phantoms, this study developed a GPU-based photon-electron coupled accelerated dose

calculation program, realizing particle transport within tetrahedral models on GPU devices for the first time. Rigorous validation demonstrates precise computational outcomes and significant acceleration, successfully addressing the challenges associated with accelerated dose calculations for mesh-type phantoms.

II. Materials and Methods

A. Manipulation of Mesh-Type Phantoms

1. Constructing Acceleration Structure During Monte Carlo simulation, determining physical and geometric step lengths requires traversal of all geometries for particle positioning and intersection operations [16, 17, 28]. Voxel-type phantoms are represented as three-dimensional arrays consisting of cuboidal voxels organized into columns, rows, and slices, where each voxel acts as a volumetric unit and every array entry defines the specific organ or tissue corresponding to each voxel [3, 8]. Since voxel phantoms feature a uniform grid of identically sized voxels, determining the voxel containing a particle based on its position can be performed swiftly. In contrast, mesh-type phantoms employ unstructured grids composed of Non-Uniform Rational B-Spline (NURBS) surfaces or polygon surfaces, where NURBS-surface phantoms cannot be directly calculated and computations involving polygon-surface phantoms are time-intensive [8, 11, 12].

To address this issue, mesh-type phantoms are typically tetrahedralized, resulting in a tetrahedral mesh-type phantom that retains the external shape of the original model but internally comprises numerous stacked tetrahedra [19]. This configuration significantly reduces intersection operation time as it only requires comparing distances to the four faces of the tetrahedron surrounding the particle. Although tetrahedralization simplifies intersection operations, the resulting tetrahedral mesh-type phantom remains unstructured, with each tetrahedron differing in shape. Consequently, particle positioning still necessitates traversing all tetrahedra. Simultaneously, calculating the distance from an external particle to the surface of the tetrahedral model involves an exhaustive search across all tetrahedra to find the one intersected by the particle' s direction of motion at the shortest distance. These traversal processes in the tetrahedral model are notably time-consuming.

To optimize data traversal efficiency, implementing an acceleration structure becomes essential [29, 30]. Such structures systematically organize data into layers, significantly reducing search times and improving query efficiency, thus facilitating complex computational tasks. With this consideration, we propose constructing a tree-like acceleration structure for all tetrahedra within the tetrahedral model. This strategy is expected to notably decrease the temporal complexity of data traversal. Considering construction time complexity and memory resource utilization, we have chosen to adopt the Bounding Volume Hierarchy (BVH) tree [31].

Central to the BVH is the concept of encapsulating scene entities within bound-

ing volumes at various hierarchy levels, which effectively reduces unnecessary object testing and enhances algorithmic efficiency. In the context of tetrahedral mesh-type phantoms, each tetrahedron' s bounding box is initially calculated to form a parent node. To simplify visualization of our acceleration structure construction process, consider a tetrahedral mesh-type phantom composed of numerous stacked tetrahedra represented by eight discrete tetrahedra, functionally equivalent in principle. As illustrated in Fig. 1 [Figure 1: see original paper], the model is represented by eight discrete tetrahedra. Initially, the bounding box is calculated for all tetrahedra, forming the parent node. These tetrahedra are then divided into two groups based on a specific pattern, such as average division by length, with each group' s bounding box calculated subsequently to form child nodes. This division process continues for left and right child nodes until the number of tetrahedra in a subdivided child node falls below a specified threshold, at which point it becomes a leaf node. Consequently, only leaf nodes contain detailed tetrahedron information, while other nodes store only bounding box information and child node references. Fig. 1 indicates that the maximum number of tetrahedra in a leaf node is 2, signifying that division stops when the number of tetrahedra in a child node is less than or equal to 2, establishing it as a leaf node. Importantly, a smaller threshold is not always advantageous as it increases tree depth, which raises memory requirements and reduces access speed, whereas a larger number might diminish the acceleration effect. In practice, during division of the tetrahedral mesh-type phantom, the threshold number of tetrahedra in leaf nodes is set to eight.

2. Optimization and Treatment When constructing a BVH tree, selecting an appropriate division strategy for each node is crucial. Simple division methods, such as equal splits by axial length or tetrahedron count, might not always yield optimal results. For example, dividing a complex phantom into upper and lower body sections by equal axial length can result in a disproportionate number of tetrahedra in the upper body due to its denser organ composition. This disparity can significantly increase depth differences between branches, reducing tree access efficiency. Similarly, equal division by tetrahedron count may balance tetrahedron numbers, but varying sizes and uneven distributions can lead to substantial bounding box volume differences between branches, disrupting the structural balance of the entire tree.

To address these challenges, a more sophisticated partitioning strategy such as the Surface Area Heuristic (SAH) algorithm can be employed [31, 32]. The SAH algorithm optimizes acceleration structure construction, such as BVH trees, by selecting the most cost-effective node partitioning strategy. The core principle involves calculating the expected cost of different partitioning schemes during node division, aiming to select the scheme with the lowest cost to achieve the fastest traversal speed. Considering the necessity for similar traversal operations in our particle transport process, employing the SAH algorithm to optimize acceleration structure construction becomes a valuable strategy. Specifically, the SAH algorithm estimates child node traversal costs by assessing node surface

areas and the likely number of geometries they contain after splitting. The space is divided into two parts, each forming a new child node. For each possible split, the SAH algorithm evaluates two main costs: the probability of traversing a child node (usually proportional to the node' s surface area) and the cost of further traversal within that node (usually related to the number of tetrahedra in the node). Through this method, the SAH algorithm seeks the partitioning scheme with the lowest total cost, thus minimizing traversal time. This approach is particularly effective for complex geometries and uneven distributions, as it adapts to various geometry shapes and sizes, ensuring balanced and efficient acceleration structure construction. While the SAH method may demand more upfront effort during construction, the payoff in enhanced traversal efficiency and speed is worthwhile, particularly given the frequent traversal operations necessary during particle transport.

The substantial number of tetrahedra in a tetrahedral mesh-type phantom leads to a complex acceleration structure with deep nesting depth. This complexity presents significant challenges for GPU calculations, which are limited by finite stack depths [27]. These limitations constrain the number of recursive calls and data structure depth that can be processed efficiently in parallel. Consequently, GPUs may struggle to manage deeply nested tree-like structures effectively due to limited stack depth, adversely affecting the performance of recursive algorithms critical for traversing these structures. To overcome this limitation, flattening the BVH tree becomes essential, which involves transforming the tree structure into a linear format for efficient parallel processing on GPUs [33]. Our implementation of the "tree flattening" method is illustrated in Fig. 2 [Figure 2: see original paper]. Instead of directly creating new child nodes within a node each time, we choose to solely record the child node index in the node sequence. This strategy results in two sequences: a BVH nodes list that sequentially stores information for each BVH node, and a tetrahedra list that sequentially stores geometric and material information of tetrahedra corresponding to leaf nodes. Flattening the BVH tree structure, by avoiding nested configurations, circumvents GPU stack depth limitations. This architectural adjustment ensures that acceleration structures for more intricate phantoms can be efficiently handled on the GPU.

B. Procedural Implementation of the Transport Algorithm on GPU

1. Transport in the Acceleration Structure Composed of Numerous Tetrahedra Following the above study, we have constructed a tetrahedral mesh-type phantom into a flattened BVH acceleration structure to facilitate efficient particle transport, which is crucial for Monte Carlo simulations. This particle transport process within the acceleration structure is illustrated in Fig. 3 [Figure 3: see original paper]. Effective execution of particle transport depends critically on two pivotal assessments, distinctly marked in red in Fig. 3: first, determining whether particles are positioned inside the tetrahedral model; and second, evaluating whether external particles have trajectories that might

intersect with the model. Both scenarios necessitate iterating over all tetrahedra. Traversal of a flattened tree generally requires establishing a stack structure, where nodes are orderly placed in a stack, enabling depth-first traversal by accessing the most recently added nodes first. In our GPU program, this functionality is achieved using a predefined stack array with a capacity of 256.

When locating particles, traversal begins from the root node, which is pushed into the stack array. The top-level node is extracted from the stack array to determine if it is a leaf node. If it is, all tetrahedra within the leaf node are traversed for positioning. If not, the algorithm assesses whether the particle lies within the bounding boxes of the left and right child nodes. If it does, the corresponding child nodes are pushed into the stack array. The process continues in a loop until no nodes remain in the stack array. Intersection computation is similar, but differs in that positioning only requires finding the tetrahedron containing the particle, while intersection computation requires traversing all intersecting nodes to identify the closest distance.

Particle transport within the acceleration structure begins with source sampling to generate the primary particle. The sampling process can accommodate various external irradiation sources, including parallel beams (e.g., Anterior-Posterior (AP), Posterior-Anterior (PA)) and divergent beams (e.g., isotropic point sources), and can perform energy spectrum sampling for polyenergetic sources. If the primary particle is located within a specific tetrahedron, the tetrahedron' s identifier is retrieved. If not, the algorithm assesses whether the particle intersects with the tree along its trajectory. Absent an intersection, the particle is immediately terminated. Conversely, if an intersection occurs, the particle is directed to the intersecting tetrahedron, and the index of this tetrahedron is acquired. Based on the tetrahedra sequence and tetrahedron index, the geometry and material properties of the current tetrahedron are ascertained. By determining the particle' s relative position to the tetrahedron, we calculate the minimum distance to the tetrahedron surface, defined as the safety distance, as well as the geometric distance from the particle to the tetrahedron surface along its path of motion. The mean free path is determined based on the elemental composition and concentration of the tetrahedron' s material, from which the physical step length is sampled. Subsequently, the transport method is determined to be either geometric transport or particle interaction, with any generated secondary particles added to the particle stack array. After one transport step, the particle state is evaluated. The primary particle is iteratively processed until termination, at which point the top secondary particle from the stack is selected. These steps are repeated until the particle stack is empty, signifying the end of transport.

2. Physical Models of the GPU Program Given the scope of our photon-electron coupled transport Monte Carlo program, particularly for human radiation dose calculations, judicious selection of appropriate physical models is essential. Our methodology prioritizes the most impactful physical interactions

that significantly influence simulation outcomes within our research scope and applications.

In this context, we excluded coherent photon scattering from our simulations after careful consideration. While relevant in certain scenarios, coherent scattering was deemed to have negligible impact on our intended outcomes, allowing us to streamline computational efforts and focus on more critical interactions. Meanwhile, we address electron scattering processes using a multiple scattering approach [16, 21, 34]. This approach leverages the condensed history technique, which enhances simulation efficiency and accuracy by reducing the computational burden associated with tracking individual scattering events.

The program primarily investigates photon-electron coupled transport processes. For clarity and detailed reference, Table 1 outlines the specific physical models employed for each process along with the corresponding effective energy ranges.

Photon physical processes predominantly draw upon the Livermore low-energy model provided by Geant4. In contrast to parameterized calculations prevalent in the low-energy domain of Geant4' s other standard physics models, the Livermore low-energy model directly leverages nuclear shell reaction cross-section information, resulting in heightened precision and a lower energy threshold for photon simulation, with an energy floor as low as 100 eV for isotopes spanning Z from 1 to 99 [16]. Meanwhile, the multiple scattering model primarily incorporates the Urban physical model based on Lewis theory [35].

TABLE 1. Physical models of photons and electrons.

Physical process	Physical model	Energy range
Photoelectric effect	Livermore	100 eV 100 MeV
Compton effect	Livermore	100 eV 100 MeV
Electron pair production	Livermore	1.02 MeV 100 MeV
Bremsstrahlung	Seltzer-Berger	1 keV 100 MeV
Ionization	Moller-Bhabha	100 eV 100 MeV
Multiple scattering	Urban	100 eV 100 MeV
Positrons annihilation	Eplus2gg	100 eV 100 MeV

3. Program Implementation Flow Drawing upon the particle transport method and physical models discussed above, and utilizing the widely embraced Monte Carlo technique as a reference, photon and electron transport has been implemented within the GPU domain. The program implementation flow is delineated in Fig. 4 [Figure 4: see original paper], with detailed process further explained by pseudocode in the supplementary materials, openly available in Science Data Bank at https://doi.org/10.57760/sciencedb.28433.

The implementation process begins by allocating memory space on the GPU, a critical step ensuring sufficient resources for complex computations. Initially, data crucial for simulation, such as particle properties and initial conditions,

are prepared on the CPU. This data is then transferred to the GPU through Compute Unified Device Architecture (CUDA) data transfer functions, which efficiently move large datasets between CPU and GPU. Following this, an appropriate configuration for device kernel functions is selected, specifying the number of thread blocks and thread grids to be executed. Subsequently, kernel functions for transport of three particle types—photons, positrons, and negatrons—are executed. These device functions can be invoked based on particle types within the incident or secondary particle stack.

During particle transport, accurately determining each particle's location within the corresponding tetrahedron is essential. This involves calculating both physical and geometric step lengths. The physical step length is determined by particle energies and tetrahedron material, while the geometric step length is calculated relative to tetrahedron boundaries. These calculations are vital as they directly influence particle interactions and subsequent reactions within the simulation.

In addition to primary transport processes, a secondary particle stack is established to manage particles generated by initial interactions. This stack operates on a last-in, first-out principle and handles up to 500 secondary particle arrays per thread, maintaining an orderly and efficient simulation process. As each particle interaction can produce additional secondary particles, these are systematically processed and transported in sequence, ensuring all resultant particles are accounted for. Finally, once all particle interactions have been simulated and the particle stack is depleted, results are transferred back to the CPU for analytical processing.

C. Dose Calculations and Efficiency Assessments

To validate the accuracy of our GPU program and ascertain its acceleration efficiency, we simulated a standard external irradiation scenario. The MRCP phantom was exposed to unidirectional, parallel beams of photons and electrons originating from a planar source measuring 600 mm × 1800 mm, located 600 mm from the phantom center. These beams converged from the anterior direction, as illustrated in Fig. 5 [Figure 5: see original paper], with energies of 0.01 MeV, 0.05 MeV, 0.1 MeV, 0.5 MeV, 1 MeV, 5 MeV, and 10 MeV.

Geant4 simulations on the CPU platform serve as our calculation benchmark. The Geant4 project source code employed in our research is derived from supplemental material provided in ICRP Report 145 [8]. Building upon the Geant4 code for external exposure provided in this supplemental material, we adapted it for our specific source, physical models, and energy thresholds. The computational framework operates on Geant4 version 10.04, incorporating the Livermore physics model, with secondary electron and photon energy thresholds set at 0.2 MeV and 0.002 MeV, respectively. This careful selection balances dose accuracy with reasonable computation speed. The simulation scenario follows the standard AP external exposure described above. CPU hardware utilized is specified

as Intel(R) Xeon(R) CPU E5-2660 v4 @ 2.00 GHz with 64 GB RAM.

The GPU program developed in this study utilizes the physical models outlined in Section II.B.2, maintaining consistency with the benchmark in terms of secondary electron and photon energy thresholds, as well as irradiation scenarios. It is executed on an NVIDIA GeForce RTX 4090 GPU with 24 GB VRAM, running CUDA version 12.2. The objective is to compute organ doses under identical irradiation conditions, thereby assessing our GPU program' s acceleration performance.

To further explore acceleration effects achieved through our methodology independent of hardware influences [22, 36, 37], we developed a serial CPU program derived from our GPU parallel program. This serial program replicates the physical models and acceleration structures used in its GPU counterpart, employing identical secondary particle cutoff parameters. It is executed on the same CPU hardware platform as the benchmark, simulating the same standard AP irradiation scenario. Essentially, our CPU serial program mirrors the GPU program in nearly every aspect, with the primary difference being the hardware platform. This strategy enables distinct assessment of acceleration effects resulting exclusively from methodology itself versus benefits derived specifically from hardware capabilities.

Our simulation and data processing results yield dose conversion coefficients, which serve as pivotal factors for converting particle fluence into organ dose (in $\mathrm{Gy \cdot cm^2}$) [38]. These coefficients are precisely calculated based on the ratio of organ absorbed dose to particle fluence, facilitating direct correlation between measured particle fluence and resulting dose absorbed by specific organs. Our calculations focus on key organs that substantially impact effective dose, particularly those with higher tissue weighting factors, including extrathoracic regions, trachea, thyroid, blood, and skin. For electrons, due to their limited penetration ability, internal organ doses are generally low; therefore, we focus on comparing dose results for surface organs such as the cornea and sensitive skin areas.

We conducted dose calculations for specific irradiation scenarios using three software tools: the benchmark Geant4 program, the CPU serial program, and the GPU program. The objective of employing these varied approaches was to evaluate both dose calculation accuracy and GPU program acceleration efficiency. Although the programs can execute Monte Carlo simulations using single or double precision floating-point, we employed single-precision floating-point for calculations considering computational efficiency. Simulations employed 10 particles to achieve lower statistical uncertainty. Within our GPU program, statistical uncertainty was determined using the batch-based method [39], which involves conducting simulations over multiple particle batches and evaluating variance among batches. Although the batch-based approach has drawbacks [40], we opted for this method due to its heightened efficiency, which is particularly advantageous for GPU-based calculations.

III. Results and Discussion

A. Comparison of Calculated Dose Values

Figure 6 [Figure 6: see original paper] compares organ-averaged absorbed doses per photon and electron fluence using the benchmark, CPU serial program, and GPU program, employing tetrahedral mesh phantoms for six specific organs across seven photon and electron energies.

As illustrated in Fig. 6, CPU serial execution results closely align with GPU execution results. This similarity is expected, as the CPU serial program is a direct adaptation of the GPU parallel program framework, utilizing identical physical models and transport methodologies. This direct adaptation ensures fundamental physics and computation principles remain consistent across both platforms, providing a reliable basis for comparing simulation times.

Moreover, maximal dose relative differences between the GPU program and benchmark remain within 5% for most scenarios, demonstrating our GPU program' s accuracy. However, an exception is noted for 0.01 MeV electron AP irradiation, where GPU program results display notable deviation from the benchmark. The substantial dose difference observed specifically at 0.01 MeV for electron exposure in skin' s most sensitive regions can be attributed to limited penetration capabilities of such low-energy electrons. These electrons deposit minimal energy within critical 50 μm–100 μm sensitive zones, leading to lower absorbed doses and consequently larger statistical errors and more pronounced dose deviations.

As depicted in Fig. 6, relative dose differences for most scenarios are confined to within twice the statistical error range (indicated by black horizontal bars). This observation suggests that, apart from the noted exception, no substantial discrepancy exists between benchmark results and GPU program results. Overall, findings underscore the accuracy of our GPU-based Monte Carlo simulation program, demonstrating its ability to produce dose calculations within acceptable statistical variances compared to conventional CPU-based programs such as Geant4.

B. Comparison of Computation Speed

Figure 7 [Figure 7: see original paper] summarizes computation time results, indicating that our GPU program achieves considerable acceleration employing tetrahedral models. For photons and electrons with energies ranging from 0.01 MeV to 10 MeV, the GPU program exhibits acceleration ratios (defined as benchmark calculation time divided by GPU program calculation time) between 120 and 2700 compared to the benchmark simulated using Geant4 on CPU with a single thread, reducing computational duration from hours to seconds for simulating 10 particles.

Comparing CPU serial and benchmark results reveals that acceleration achieved through acceleration structures and physical models is of single-digit magni-

tude. This implies that the enhancement in computational speed attributed to implementing acceleration structures and physical models on the CPU side is marginal, with the predominant contribution to overall acceleration emanating from GPU utilization.

Moreover, acceleration improvement is more significant for electrons than photons, which is associated with thread divergence [41, 42]. In GPU-based Monte Carlo simulations, when one particle finishes its path but others within the same thread group remain moving, the completed thread waits idly due to thread divergence, leading to resource use inefficiencies. Given the intrinsic characteristics of charged particles, electrons exhibit higher interaction propensity within material media compared to photons. This results in shorter electron ranges and transport distances, with relatively smaller variation in transport times between different electrons. Consequently, reduced thread divergence amplifies the acceleration effect.

Similarly, as particle energy increases, acceleration efficiency decreases. This occurs because transport times for low-energy particles are quite similar, resulting in less thread divergence among these particles and thereby enhancing acceleration effects.

IV. Conclusions

This study successfully accomplishes GPU-accelerated dose calculation of mesh-type models for the first time, providing an expeditious and accurate tool for dose calculation in mesh-type computational phantoms. These swift calculations are particularly beneficial in scenarios such as radiation therapy, accidental radiation events, occupational exposures, and individual dosimetry assessment, where finely detailed and personalized phantoms along with faster computational methods are critical. This advancement facilitates more precise radiation dose assessments, which are crucial for effective and safe medical practices.

The GPU-accelerated Monte Carlo program employs an acceleration structure coupled with optimized traversal processes, realizing particle transport within tetrahedral models. For typical external photon and electron AP irradiation scenarios, most organ dose relative differences remain within 5%. Compared to the benchmark simulated by Geant4 on CPU with a single thread, computational speed has increased by factors ranging from 120 to 2700, diminishing the time required to simulate 10 particles from hours to seconds. A minor portion of the acceleration effect arises from acceleration structures and varied physical models, while the majority stems from GPU utilization.

Future research will focus on enhancing particle transport simulation efficiency to achieve real-time dose calculations for mesh-type phantoms. Key improvements will involve individualizing phantoms based on specific exposure scenarios and optimizing GPU Monte Carlo programs to further reduce GPU thread divergence.

References

[1] H. Liu, J. Li, R. Qiu et al., Dose conversion coefficients for Chinese reference adult male and female voxel phantoms from idealized neutron exposures. J. Nucl. Sci. Tech. 54, 921–932 (2017). doi:10.1080/00223131.2017.1323685

[2] Y. Wu, M. Cheng, W. Wang et al., Development of Chinese Female Computational Phantom Rad-Human and Its Application in Radiation Dosimetry Assessment. Nucl. Tech. 201, 155–164 (2018). doi:10.1080/00295450.2017.1411717

[3] H.G. Menzel, C. Clement, P. DeLuca, ICRP Publication 110. Realistic reference phantoms: an ICRP/ICRU joint effort. A report of adult reference computational phantoms. Ann. ICRP 39, 1-164 (2009). doi:10.1016/j.icrp.2009.09.001

[4] A. Alfuraih, O. Kadri, K. Alzimami, Investigation of SPECT/CT cardiac imaging using Geant4. Nucl. Sci. Tech. 29, 105 (2018). doi:10.1007/s41365-018-0435-8

[5] R. Qiu, J.L. Li, S.M. Huang et al., Organ dose conversion coefficients for external neutron irradiation based on the Chinese mathematical phantom (CMP). J. Nucl. Sci. Technol. 49, 263–271 (2012). doi:10.1080/00223131.2011.649081

[6] M.E. Miyombo, Y.K. Liu, A. Ayodeji, An analytical approach for photon attenuation coefficient estimation based on material composition. Prog. Nucl. Energy (2021). doi:https://doi.org/10.1016/j.pnucene.2021.103955

[7] X.Y. Luo, R. Qiu, Z. Wu et al., THUDosePD: a three-dimensional Monte Carlo platform for phantom dose assessment. Nucl. Sci. Tech. 34, 164 (2023). doi:10.1007/s41365-023-01315-y

[8] C.H. Kim, Y.S. Yeom, N. Petoussi-Henss et al., ICRP Publication 145: Adult Mesh-Type Reference Computational Phantoms. Ann. ICRP 49, 13–201 (2020). doi:10.1177/0146645319893605

[9] Y. Liu, T.W. Xie, Q. Liu, Monte Carlo simulation for internal radiation dosimetry based on the high resolution Visible Chinese Human. Nucl. Sci. Tech. 22, 165–173 (2011).

[10] K. Zhao, M.Y. Cheng, P.C. Long et al., A hybrid voxel sampling method for constructing Rad-HUMAN phantom. Nucl. Sci. Tech. 25, 020503 (2014). doi:10.13538/j.1001-8042/nst.25.020503

[11] W.P. Segars, B.M. Tsui, D.S. Lalush et al., Development and application of the new dynamic Nurbs-based Cardiac-Torso (NCAT) phantom. J. Nucl. Med. 42, 7P-7P (2001).

[12] Y.S. Yeom, M.C. Han, C.H. Kim et al., Conversion of ICRP male reference phantom to polygon-surface phantom. Phys. Med. Biol. 58, 6985–7007 (2013). doi:10.1088/0031-9155/58/19/6985

[13] C. Choi, T.T. Nguyen, Y.S. Yeom et al., Mesh-type reference Korean phantoms (MRKPs) for adult male and female for use in radiation protection dosime-

try. Phys. Med. Biol. 64, 085020 (2019). doi:10.1088/1361-6560/ab0b59

[14] C. Cumur, T. Fujibuchi, K. Hamada, Dose estimation for cone-beam computed tomography in image-guided radiation therapy using mesh-type reference computational phantoms and assuming head and neck cancer. J. Radiol. Prot. 42, 021533 (2022). doi:10.1088/1361-6498/ac7914

[15] M. Bhar, O. Kadri, K. Manai, Assessment of self- and cross-absorbed SAF values for HDRK-man using Geant4 code: internal photon and electron emitters. Nucl. Sci. Tech. 30, 149 (2019). doi:10.1007/s41365-019-0675-2

[16] S. Agostinelli, J. Allison, K. Amako et al., GEANT4—a simulation toolkit. Nucl. Instrum. Methods Phys. Res. Sect. A-Accel. Spectrom. Dect. Assoc. Equip. 506, 250–303 (2003). doi:10.1016/s0168-9002(03)01368-8

[17] Z.F. Luo, R. Qiu, M. Li et al., Study of a GPU-based parallel computing method for the Monte Carlo program. Nucl. Sci. Tech. 25, S010501 (2014). doi:10.13538/j.1001-8042/nst.25.S010501

[18] C.H. Kim, J.H. Jeong, W.E. Bolch et al., A polygon-surface reference Korean male phantom (PSRK-Man) and its direct implementation in Geant4 Monte Carlo simulation. Phys. Med. Biol. 56, 3137–3161 (2011). doi:10.1088/0031-9155/56/10/016

[19] Y.S. Yeom, J.H. Jeong, M.C. Han et al., Tetrahedral-mesh-based computational human phantom for fast Monte Carlo dose calculations. Phys. Med. Biol. 59, 3173–3185 (2014). doi:10.1088/0031-9155/59/12/3173

[20] Z. Peng, Y. Lu, Y. Xu et al., Development of a GPU-accelerated Monte Carlo dose calculation module for nuclear medicine, ARCHER-NM: demonstration for a PET/CT imaging procedure. Phys. Med. Biol. 67, 06NT02 (2022). doi:10.1088/1361-6560/ac58dd

[21] L. Su, Y.M. Yang, B. Bednarz et al., ARCHERRT—A GPU-based and photon-electron coupled Monte Carlo dose computing engine for radiation therapy: Software development and application to helical tomotherapy. Med. Phys. 41, 071709 (2014). doi:10.1118/1.4884229

[22] A.K. Hu, R. Qiu, H. Liu et al., THUBrachy: fast Monte Carlo dose calculation tool accelerated by heterogeneous hardware for high-dose-rate brachytherapy. Nucl. Sci. Tech. 32, 32 (2021). doi:10.1007/s41365-021-00866-2

[23] W.G. Li, C. Chang, Y. Qin et al., GPU-based cross-platform Monte Carlo proton dose calculation engine in the framework of Taichi. Nucl. Sci. Tech. 34, 77 (2023). doi:10.1007/s41365-023-01218-y

[24] X. Jia, X.J. Gu, J. Sempau et al., Development of a GPU-based Monte Carlo dose calculation code for coupled electron-photon transport. Phys. Med. Biol. 55, 3077–3086 (2010). doi:10.1088/0031-9155/55/11/006

[25] Y. Li, W. Sun, H. Liu et al., Development of a GPU-superposition Monte Carlo code for fast dose calculation in magnetic fields. Phys. Med. Biol. 67,

125002 (2022). doi:10.1088/1361-6560/ac7194

[26] G. Franciosini, G. Battistoni, A. Cerqua et al., GPU-accelerated Monte Carlo simulation of electron and photon interactions for radiotherapy applications. Phys. Med. Biol. 68, 044001 (2023). doi:10.1088/1361-6560/aca1f2

[27] K. Yang, B.S. He, Q.O. Luo et al., Stack-Based Parallel Recursion on Graphics Processors. ACM SIGPLAN Not. 44, 299–300 (2009). doi:10.1145/1594835.1504224

[28] L. Deng, G. Li, T. Ye et al., MCDB Monte Carlo Code with Fast Track Technique and Mesh Tally Matrix for BNCT. J. Nucl. Sci. Technol. 44, 1518–1525 (2007). doi:10.1080/18811248.2007.9711401

[29] A. Aman, S. Demirci, U. Güdükbay et al., Multi-level tetrahedralization-based accelerator for ray-tracing animated scenes. Comput. Anim. Virtual Worlds 32, e2024 (2021). doi:10.1002/cav.2024

[30] X. Wang, Y. Yu, X. Li et al., Development of a hybrid parallelism Monte Carlo transport middleware on mesh geometry. Ann. Nucl. Energy 190, 109872 (2023). doi:https://doi.org/10.1016/j.anucene.2023.109872

[31] M. Vinkler, V. Havran, J. Sochor, Visibility driven BVH build up algorithm for ray tracing. Comput. Graph. 36, 283–296 (2012). doi:10.1016/j.cag.2012.02.013

[32] J.D. MacDonald, K.S. Booth, Heuristics for ray tracing using space subdivision. Vis. Comput. 6, 153–166 (1990). doi:10.1007/bf01911006

[33] W. Boukaram, G. Turkiyyah, D. Keyes, Hierarchical Matrix Operations on GPUs: Matrix-Vector Multiplication and Compression. ACM Trans. Math. Softw. 45, 3 (2019). doi:10.1145/3232850

[34] Z. Tian, F. Shi, M. Folkerts et al., A GPU OpenCL based cross-platform Monte Carlo dose calculation engine (goMC). Phys. Med. Biol. 60, 7419–7435 (2015). doi:10.1088/0031-9155/60/19/7419

[35] V.N. Ivanchenko, O. Kadri, M. Maire et al., Geant4 models for simulation of multiple scattering. J. Phys.: Conf. Ser. 219, 032045 (2010). doi:10.1088/1742-6596/219/3/032045

[36] T. Liu, X.G. Xu, C.D. Carothers, Comparison of accelerators for Monte Carlo radiation transport calculations, Nvidia Tesla M2090 GPU and Intel Xeon Phi 5110p coprocessor: A case study for X-ray CT imaging dose calculation. Ann. Nucl. Energy 82, 230–239 (2015). doi:https://doi.org/10.1016/j.anucene.2014.08.061

[37] Y. Wang, J. Liang, Q. Zhang et al., Development and verification of Geant4-based parallel computing Monte Carlo simulations for nuclear logging applications. Ann. Nucl. Energy 172, 109079 (2022). doi:https://doi.org/10.1016/j.anucene.2022.109079

[38] N. Petoussi-Henss, W.E. Bolch, K.F. Eckerman et al., ICRP Publication 116. Conversion coefficients for radiological protection quantities for external radiation exposures. Ann. ICRP 40, 1–257 (2010). doi:10.1016/j.icrp.2011.10.001

[39] H. Lee, J. Shin, J.M. Verburg et al., MOQUI: an open-source GPU-based Monte Carlo code for proton dose calculation with efficient data structure. Phys. Med. Biol. 67, 174001 (2022). doi:10.1088/1361-6560/ac8716

[40] B.R.B. Walters, I. Kawrakow, D.W.O. Rogers, History by history statistical estimators in the BEAM code system. Med. Phys. 29, 2745–2752 (2002). doi:10.1118/1.1517611

[41] S.P. Hamilton, T.M. Evans, Continuous-energy Monte Carlo neutron transport on GPUs in the Shift code. Ann. Nucl. Energy 128, 236–247 (2019). doi:10.1016/j.anucene.2019.01.012

[42] S.P. Hamilton, S.R. Slattery, T.M. Evans, Multigroup Monte Carlo on GPUs: Comparison of history- and event-based algorithms. Ann. Nucl. Energy 113, 506–518 (2018). doi:10.1016/j.anucene.2017.11.032

Note: Figure translations are in progress. See original paper for figures.