Journal of Biomedical Optics

BiomedicalOptics.SPIEDigitalLibrary.org

Dual-grid mesh-based Monte Carlo algorithm for efficient photon transport simulations in complex three-dimensional media

Shijie Yan
Anh Phong Tran
Qianqian Fang

SPIE.

Shijie Yan, Anh Phong Tran, Qianqian Fang, "Dual-grid mesh-based Monte Carlo algorithm for efficient photon transport simulations in complex three-dimensional media," *J. Biomed. Opt.* **24**(2), 020503 (2019), doi: 10.1117/1.JBO.24.2.020503.

Dual-grid mesh-based Monte Carlo algorithm for efficient photon transport simulations in complex three-dimensional media

Shijie Yan,[a] Anh Phong Tran,[b] and Qianqian Fang[c,*]
[a]Northeastern University, Department of Electrical and Computer Engineering, Boston, Massachusetts, United States
[b]Northeastern University, Department of Chemical Engineering, Boston, Massachusetts, United States
[c]Northeastern University, Department of Bioengineering, Boston, Massachusetts, United States

Abstract. The mesh-based Monte Carlo (MMC) method is an efficient algorithm to model light propagation inside tissues with complex boundaries, but choosing appropriate mesh density can be challenging. A fine mesh improves the spatial resolution of the output but requires more computation. We propose an improved MMC—dual-grid mesh-based Monte Carlo (DMMC)—to accelerate photon simulations using a coarsely tessellated tetrahedral mesh for ray-tracing computation and an independent voxelated grid for output data storage. The decoupling between ray-tracing and data storage grids allows us to simultaneously achieve faster simulations and improved output spatial accuracy. Furthermore, we developed an optimized ray-tracing technique to eliminate unnecessary ray–tetrahedron intersection tests in optically thick mesh elements. We validate the proposed algorithms using a complex heterogeneous domain and compare the solutions with those from MMC and voxel-based Monte Carlo. We found that DMMC with an unrefined constrained Delaunay tessellation of the boundary nodes yielded the highest speedup, ranging from 1.3× to 2.9× for various scattering settings, with nearly no loss in accuracy. In addition, the optimized ray-tracing technique offers excellent acceleration in high-scattering media, reducing the ray–tetrahedron test count by over 100-fold. Our DMMC software can be downloaded at http://mcx.space/mmc. [DOI: 10.1117/1.JBO.24.2.020503]

Keywords: mesh-based Monte Carlo; photon transport; mesh generation; optical imaging.

Paper 180493LRR received Aug. 12, 2018; accepted for publication Jan. 22, 2019; published online Feb. 20, 2019.

1 Introduction

With the rapidly evolving computational capabilities of modern central processing units (CPUs) and graphics processing units (GPUs), the Monte Carlo (MC) method[1] has taken an increasingly important role in understanding complex photon–tissue interactions in biomedical optics applications. The increasing popularity of MC is largely due to its superior generality and high scalability in computation.[2] As a stochastic solver to the radiative transfer equation, MC is capable of dealing with general complex media, such as low-scattering tissues, e.g., synovial fluids in the joint, air pockets in the lung, and cerebrospinal fluid (CSF) in the brain, or highly absorbing organs, such as livers, without compromising accuracy due to the use of approximations. The intuitive simulation strategies also allow easy adaptation for a wide-range of applications.[2]

Despite the advantages of the MC method, its wide adoption was limited by slow computation. However, the rise of general-purpose GPU has enabled massively parallel simulations and resulted in hundred- to thousand-fold acceleration. Notable GPU MC algorithms include MCX,[3] CUDAMCML,[4] GPU-MOSE,[5] MCX-CL,[6] and O^{3}MC.[7] Extensive cross validations between these algorithms have been performed.[8–10]

On the other hand, the proposals of the mesh-based Monte Carlo (MMC) approach[11,12] reflect the desires of higher flexibility and accuracy when modeling complex heterogeneous domains, such as human anatomy. The use of a tetrahedral mesh model overcomes the staircase pattern that often characterizes voxel-based tissue representations, while also allowing notable improvement in memory efficiency.[12] Compared to the surface-based MC methods,[5,13] the use of tetrahedral elements drastically reduces ray–triangle intersection tests.[10]

In nearly all published MMC implementations, a single tetrahedral mesh is used for three main purposes: (1) representing complex three-dimensional (3-D) tissue boundaries, (2) ray-tracing acceleration by restricting the intersection testing to a single tetrahedron, and (3) storing the output quantities at the nodes[12] or elements.[11,14] To capture accurate spatial distributions of fluence, a dense tetrahedral mesh is often desirable,[12] but results in longer simulation time. In most cases, refined mesh elements do not improve tissue boundary accuracy as they fall under the same tissue type. Because a photon's trajectory is only dependent on boundaries delineating different tissues, a coarsely tessellated mesh containing only boundary nodes is more computationally advantageous. The conflict on the desirable mesh densities between accuracy and speed inspires us to propose a "dual-grid" mesh-based Monte Carlo (DMMC) featuring a coarsely tessellated mesh for ray-tracing calculations, referred to as the "forward mesh," and an independent voxelated space for output storage, referred to as the "output grid." Although coarsely tessellated meshes have been found previously in MMC studies,[15,16] the use of a dual-mesh for output data storage has not been reported.

As we coarsen the mesh to only preserve the boundary nodes, the dimensions of each tetrahedron may grow and become significantly larger compared to photon's mean-free-path (MFP). In such case, a photon may experience a large number of scattering events before migrating into the adjacent element, making the per-step ray–tetrahedron intersection tests increasingly redundant. To further optimize MMC speeds on coarse meshes, a method to effectively eliminate such redundant intersection tests may yield a significant performance improvement.

In the rest of this paper, we first describe the method to create a coarse tetrahedral mesh that accurately preserves complex boundaries without unnecessary mesh refinement and then discuss the procedure to deposit fluence into the output grid in Sec. 2. Furthermore, we describe our optimized MMC

*Address all correspondence to Qianqian Fang, E-mail: q.fang@neu.edu

ray-tracing technique to avoid unnecessary ray–tetrahedron intersection tests. In Sec. 3, we test both the accuracy of the DMMC approach using a heterogeneous domain, as well as the speed improvement compared to conventional single-mesh MMC. The acceleration is evaluated at various scattering coefficient and mesh density settings. Finally, we summarize our findings and discuss the next steps of this research in Sec. 4.

2 Materials and Methods

As we demonstrated previously,[16] MMC simulation accuracy, unlike that of the finite-element method, is insensitive to the presence of poorly shaped tetrahedral elements or "slivers."[17] The mesh that can preserve the tissue boundaries with minimum number of total nodes is simply a 3-D tessellation of the boundary nodes. A tessellation algorithm usually outputs either a Delaunay or non-Delaunay mesh.[17] When one requires a set of predefined nodes and triangles to present in the output mesh, such as the pregenerated tissue boundaries, the output mesh is referred to as a constrained Delaunay triangulation (CDT) of the input nodes. A CDT mesh is typically not a Delaunay mesh but can be converted into a Delaunay-conforming mesh following a series of triangle-flipping and refinement steps.[17]

Creating CDTs from a set of predefined tissue surfaces is supported by numerous 3-D mesh generators. Here, we use an open-source 3-D mesh generator, TetGen[17] in combination with Iso2Mesh,[18] to obtain a coarse tessellation of the boundary nodes and triangles. To minimize the output mesh size, the TetGen flag "-Y" is used to suppress the insertion of Steiner points as part of the Delaunay refinement.[17] In some rare cases, a CDT may not exist for a given set of input boundaries. In such cases, Delaunay refinement must be permitted using the "-Yq" flag with TetGen, but excessive refinement is prevented.

Once a photon starts to propagate inside the above generated coarse forward mesh, the next step is to store the photon energy deposition in an independent "dual-mesh." For simplicity and minimization of memory demands, a voxelated grid coincident to the bounding box of the tetrahedral mesh space is dynamically created, similar to the grid used to store outputs in the voxel-based MC (MCX).[3] The voxel size of the output grid is user-defined and is independent to the tetrahedral mesh. In our implementation, the default size of each output grid voxel is $1 \times 1 \times 1$ mm^3, although anisotropic voxels can also be supported.

Similar to MMC, every scattering path in DMMC is divided into path segments by tetrahedron/tissue boundaries. Unlike MMC, the energy loss in each path segment is not accumulated to the enclosing element, but within an independent voxelated grid. Each path segment, such as $\overrightarrow{P_k - P_{k+1}}$ in Fig. 1(b), is further divided into equal-length subsegments, indicated by the black division marks in Fig. 1(b), with a step size smaller than half of the output grid's voxel edge length l_E. The energy loss of the photon packet along each path subdivision is then calculated using the Beer–Lambert law and accumulated to the voxel that encloses the midpoint of the segment. In our current implementation, a single-instruction multiple-data algorithm is used for both the ray-tracing calculations—using a "branchless Badouel" algorithm[14]—and the energy depositions to the output grid.

To further reduce the redundant ray–tetrahedron intersection tests in a coarse mesh, here, we introduce a ray-tracing technique specifically optimized for MMC simulations on meshes with optically thick elements. In this technique, we calculate and maintain the signed perpendicular distances of the current photon position to the four triangles of the tetrahedron using a lightweight update strategy, and only perform the ray–tetrahedron intersection computation when one or more distance values change the sign. Illustrations explaining the algorithm can be found in Figs. 1(a) and 1(b).

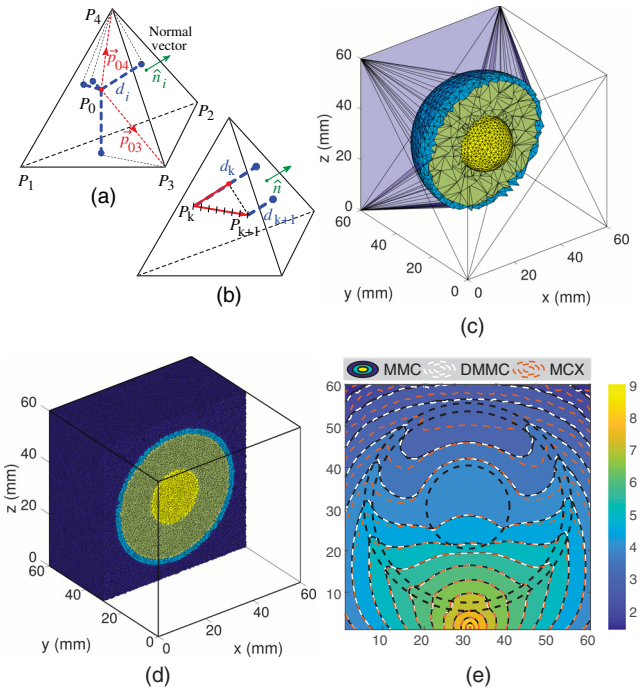

Fig. 1 Illustrations explaining (a) the calculations, (b) updates of distance-based ray-tracing algorithm, and (c)–(e) validations of DMMC. Particularly, we show the cross-cut views ($y = 30.5$ mm) of (c) the coarse forward mesh for DMMC, (d) fine mesh for MMC, and (e) contour plots of the fluence, in log-scale, for DMMC (white dashed lines), MMC (black solid lines), and MCX (orange dashed lines).

Specifically, the signed distances between the photon and the four facets of the enclosing element can be calculated using the inner products between the precomputed outward-pointing surface normal vectors ($\hat{n}_i$) and the displacement vectors ($\vec{p}_{03}$ and $\vec{p}_{04}$) of the current position, as depicted in Fig. 1(a). When a photon packet moves from one scattering site to the next, the four signed distances can be incrementally adjusted by adding the inner products between the displacement vector and the surface normal vectors ($\overrightarrow{P_k - P_{k+1}} \cdot \hat{n}_i$) [see Fig. 1(b)]. When a photon attempts to escape from the current element, at least one of the signed distances should become negative. One can then perform the ray–tetrahedron intersection testing to determine the exact exit position. Otherwise, photons traverse to the next scattering site without extraneous computations. As the dimensions of the element become much larger than the MFP, such saving in computation can become significant.

3 Results and Discussions

We validate the proposed DMMC algorithm, including the optimized algorithm for optically thick meshes, by comparing its solutions with single-mesh MMC and MCX[3] in a complex heterogeneous domain. We are particularly interested in measuring the overall performance improvement, in terms of simulation speed in photon/ms, over a range of mesh refinement levels and scattering coefficients, as well as reduction of ray–tetrahedron intersection test counts. Finally, we perform a profiling analysis to identify the changes in the runtimes in both

ray-tracing and data storage operations in the optimized ray-tracing technique. All tests are performed on a desktop running Ubuntu Linux 18.04 with an Intel i7-8700K CPU. A Henyey–Greenstein phase function[1] is assumed in all simulations.

In Figs. 1(c)–1(e), we show a comparison between DMMC, MMC and the MCX simulations using a heterogeneous numerical phantom. The numerical phantom contains a $60 \times 60 \times 60$ mm^3 cubic domain filled with a tissue-like medium with absorption coefficient $\mu_a = 0.02$ mm^{-1}, scattering coefficient $\mu_s = 7.0$ mm^{-1}, anisotropy $g = 0.89$, and refractive index $n = 1.37$. A 10-mm radius nonscattering spherical inclusion with $\mu_a = 0.05$ mm^{-1}, $\mu_s = 0.0$ mm^{-1}, $g = 1.0$, and $n = 1.37$ is centered in the domain and is further enclosed by two concentric spherical shells (inner shell has a radius between 10 and 23 mm with $\mu_a = 0.02$ mm^{-1}, $\mu_s = 9.0$ mm^{-1}, $g = 0.89$, and $n = 1.37$; outer shell has a radius between 23 and 25 mm with $\mu_a = 0.004$ mm^{-1}, $\mu_s = 0.009$ mm^{-1}, $g = 0.89$, and $n = 1.37$ to mimic CSF). This structure is selected to highlight the advantages of a mesh-based MC over MCX when modeling complex shapes. For DMMC, a coarse mesh containing only 3733 nodes and 21,256 elements is created using the "-Yq" flag with TetGen, shown in Fig. 1(c). This mesh is a coarse CDT generated from 4 triangular surfaces, including 793, 1029, and 1217 nodes from the 3 spherical surfaces at $r = 10$, 23, and 25 mm, respectively, and 8 nodes from the bounding box surface. As shown in Fig. 1(d), the mesh used for MMC, containing 604,297 nodes and 3,733,387 elements, is a Delaunay refinement derived from the same input surfaces. For MCX, a $60 \times 60 \times 60$ grid with 1 mm^3 isotropic voxels is used to discretize the domain and store the output. A pencil beam source is positioned at (30.5, 30.5, 0) mm pointing at the $+z$ axis; a total of 10^8 photons are simulated in all cases. The DMMC simulation uses a $60 \times 60 \times 60$ output grid, coincident to that of MCX. Contour plots of the fluence along the source plane ($y = 30.5$ mm) for DMMC, MMC, and MCX are compared in Fig. 1(e).

Despite the fractional numbers of nodes and elements used, the DMMC simulation produced a solution that matches excellently with that of MMC in most areas in Fig. 1(e). As we showed previously[19] but not here, the DMMC and MCX solutions match excellently near the source while the single-mesh MMC shows a degradation in accuracy due to limited mesh resolution. If we compare the MCX fluence with those from DMMC and MMC in the regions distal to the source, a large discrepancy can be observed within the two spherical shells. We believe this is a result of the staircase approximation of the spherical boundaries in MCX. Overall, the use of a coarse forward mesh allows DMMC to achieve a 2.4-fold speed improvement over MMC (74.64 versus 31.43 photon/ms using 12 threads on the Intel CPU). The MCX simulation reported a speed of 13,760 photon/ms using an NVIDIA Titan V GPU.

Next, we perform a systematic study to characterize the impacts of varying mesh densities and scattering coefficients on DMMC's speed and accuracy. A uniform cubic domain, with $\mu_a = 0.005$ mm^{-1} and $n = 1$, is set to 3 scattering settings: (S1) $\mu_s = 0.5$ mm^{-1}, $g = 0.01$; (S2) $\mu_s = 1$ mm^{-1}, $g = 0.01$; (S3) $\mu_s = 10$ mm^{-1}, $g = 0.9$. A pencil beam similar to the above test is used. A total of 19 tetrahedral meshes are created using TetGen, by incrementally reducing the maximum element volume size by a factor of 2. The finest mesh contains 433,020 nodes and 2,673,099 elements and the coarsest one has only 8 nodes and 6 tetrahedra. DMMC simulations with and without the distance-based ray-tracing optimization are performed. All simulations output fluence in a $60 \times 60 \times 60$ grid with 1 mm^3 voxels. In Fig. 2(a), we plot the node and element counts (blue) of the 19 tested meshes on the left axis, and the DMMC simulation speed (in photon/ms, red) at two scattering settings (S1 and S2) on the right axis. For the speed plot, we report both the simulations with (solid) and without (dotted) the distance-based ray-tracing optimization. In addition, we run the simulations with the S3 configuration and plot in Fig. 2(b), the counts of ray–tetrahedron tests in the left axis and speed in the right axis. The runtimes of the simulation steps before and after the optimization are profiled and reported as a stacked-bar plot in Fig. 2(c). Furthermore, the dB root-mean-square (RMS) errors between DMMC and MCX on a matching output grid ($60 \times 60 \times 60$ voxels) are calculated at various mesh densities and plotted in Fig. 2(d).

A nearly monotonic increase in speed is observed from Fig. 2(a) as one coarsens the mesh. Using the mesh with an average element volume $V_a = 1$ mm^3 (size of a voxel in the output grid) as reference, the speedup ratios for DMMC with the coarsest mesh ($V_a = 36{,}000$ mm^3) for S1, S2, and S3 are 2.68×, 1.88×, and 1.14×, respectively, without the ray-tracing optimization, and 2.94×, 2.19×, and 1.33×, respectively, with the optimization. Applying the optimized ray-tracing results

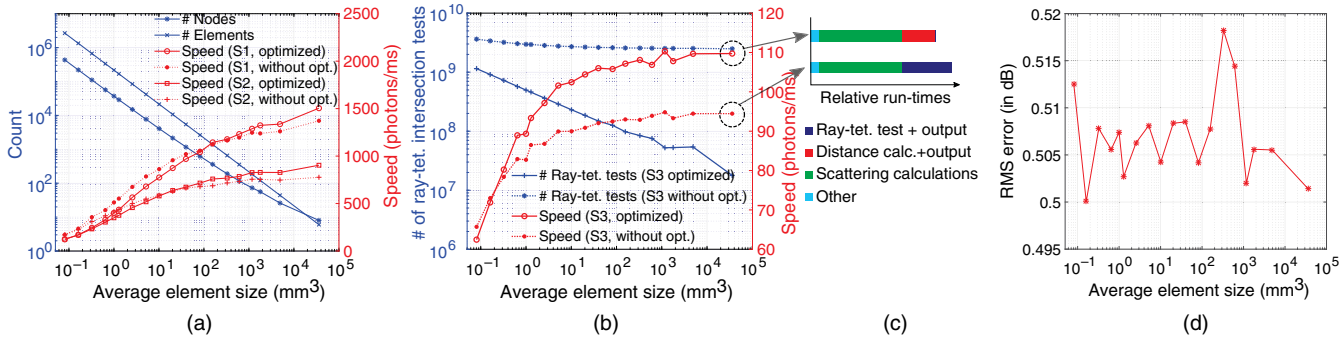

Fig. 2 Speed benchmark and error assessment for DMMC. We show the simulation speeds (red, right axes) at 19 different mesh sizes with (a) low-μ_s (S1: $\mu_s = 0.5$/mm, $g = 0.01$; S2: $\mu_s = 1$/mm, $g = 0.01$) and (b) high-μ_s (S3: $\mu_s = 10$/mm, $g = 0.9$) settings. The mesh node/element and ray–tetrahedron testing counts are shown in the left axes of (a)/(b), respectively. In (c), we show the relative runtimes of the simulation for a selected case; in (d), we plot the fluence RMS error in dB, i.e., $|20\log_{10}(\text{DMMC}) - 20\log_{10}(\text{MCX})|$, at different mesh densities.

in 1.10×, 1.16×, and 1.16× speedups for S1, S2, and S3, respectively, when using the coarsest mesh.

The significantly improved simulation speed obtained at the coarsest mesh is a result of reduced ray–tetrahedron intersection tests. It is apparent that such speedup is dependent upon the scattering coefficient settings—the lower the scattering coefficient, the higher the acceleration—as expected. When the medium's scattering coefficient is low, the ray–tetrahedron intersection tests dominate the runtime. Thus, reducing mesh density can lead to a significant reduction in the overall runtime. However, as shown in Fig. 2(b), when the scattering coefficient is relatively high, the scattering-related calculations become dominant; coarsened meshes affected lesser speed enhancement than in the low-μ_s cases.

On the other hand, the distance-tracking-based ray-tracing optimization achieves the highest acceleration at the highest μ_s setting (S3), cutting the ray–tetrahedron intersection testing counts by an impressive 100-fold while reporting a 16% overall speed improvement over the standard MMC algorithm at the coarsest mesh. This is again expected as the optimized ray-tracing algorithm skips unnecessary ray–tetrahedron intersection tests, which occur most frequently at the coarsest mesh and the highest scattering. The moderate overall speed improvement results from scattering and data storage costs dominating runtimes in high-μ_s cases, as indicated by the stacked-bar chart in Fig. 2(c).

The error plot in Fig. 2(d) further assures us that the large reduction in internal elements within uniform tissue regions shows no sign of accuracy loss as long as the output data are stored on a fine grid. This is because the photon packet's trajectory is only dependent upon the boundaries between heterogeneities. Mesh refinements without altering the tissue boundaries would not change the photon paths (within the limit of numerical precision); thus, the output of DMMC stays the same. This result suggests that the coarsest CDT of the boundary nodes is the most desirable forward mesh for DMMC, as it provides the highest computational speed with little loss in accuracy.

4 Conclusion

In summary, we reported a DMMC algorithm to significantly improve 3-D mesh-based photon transport simulations. Compared to conventional MMC, DMMC achieved a 1.3× to 2.9× speed improvement over a range of scattering coefficient settings while simultaneously improving the spatial accuracy of the output fluence. We want to particularly highlight our finding that photon trajectories in an MC simulation are insensitive to interior element boundaries within a given tissue region. This allows us to decouple the storage grid from the ray-tracing grid and utilize the most suited discretization in respective steps. In addition, we developed a distance-based ray-tracing technique to reduce unnecessary ray–tetrahedron intersection tests by over 100-fold in high-μ_s media, resulting in an overall 16% speed increase in a benchmark. The next steps of this research include anisotropic output grid support and GPU-accelerated DMMC. The DMMC algorithm has been incorporated into our MMC software and can be freely downloaded from http://mcx.space/mmc.

Disclosures

No conflicts of interest, financial or otherwise, are declared by the authors.

Acknowledgments

This research was supported by the National Institutes of Health (NIH) Grant Nos. R01-GM114365, R01-CA204443, and R01-EB026998.

References

1. L. V. Wang, S. L. Jacques, and L. Zheng, "MCML-Monte Carlo modeling of light transport in multi-layered tissues," *Comput. Methods Progr. Biomed.* **47**(2), 131–146 (1995).
2. C. Zhu and Q. Liu, "Review of Monte Carlo modeling of light transport in tissues," *J. Biomed. Opt.* **18**(5), 050902 (2013).
3. Q. Fang and D. A. Boas, "Monte Carlo simulation of photon migration in 3D turbid media accelerated by graphics processing units," *Opt. Express* **17**(22), 20178–20190 (2009).
4. E. Alerstam et al., "Next-generation acceleration and code optimization for light transport in turbid media using GPUs," *Biomed. Opt. Express* **1**(2), 658–675 (2010).
5. N. Ren et al., "GPU-based Monte Carlo simulation for light propagation in complex heterogeneous tissues," *Opt. Express* **18**(7), 6811–6823 (2010).
6. L. Yu et al., "Scalable and massively parallel Monte Carlo photon transport simulations for heterogeneous computing platforms," *J. Biomed. Opt.* **23**(1), 010504 (2018).
7. A. Doronin and I. Meglinski, "Online object oriented Monte Carlo computational tool for the needs of biomedical optics," *Biomed. Opt. Express* **2**(9), 2461–2469 (2011).
8. L. Wang, S. Ren, and X. Chen, "Comparative evaluations of the Monte Carlo-based light propagation simulation packages for optical imaging," *J. Innovative Opt. Health Sci.* **11**(01), 1750017 (2018).
9. H. Shen and G. Wang, "A study on tetrahedron-based inhomogeneous Monte Carlo optical simulation," *Biomed. Opt. Express* **2**(1), 44–57 (2011).
10. Q. Fang, "Comment on 'A study on tetrahedron-based inhomogeneous Monte-Carlo optical simulation'," *Biomed. Opt. Express* **2**(5), 1258–1264 (2011).
11. H. Shen and G. Wang, "A tetrahedron-based inhomogeneous Monte Carlo optical simulator," *Phys. Med. Biol.* **55**(4), 947–962 (2010).
12. Q. Fang, "Mesh-based Monte Carlo method using fast ray-tracing in Plücker coordinates," *Biomed. Opt. Express* **1**(1), 165–175 (2010).
13. E. Margallo-Balbás and P. J. French, "Shape based Monte Carlo code for light transport in complex heterogeneous tissues," *Opt. Express* **15**(21), 14086–14098 (2007).
14. Q. Fang and D. Kaeli, "Accelerating mesh-based Monte Carlo method on modern CPU architectures," *Biomed. Opt. Express* **3**(12), 3223–3230 (2012).
15. Y. S. Yeom et al., "Tetrahedral-mesh-based computational human phantom for fast Monte Carlo dose calculations," *Phys. Med. Biol.* **59**(12), 3173–3185 (2014).
16. R. Yao, X. Intes, and Q. Fang, "Generalized mesh-based Monte Carlo for wide-field illumination and detection via mesh retessellation," *Biomed. Opt. Express* **7**(1), 171–184 (2016).
17. H. Si, "TetGen, a Delaunay-based quality tetrahedral mesh generator," *AMC Trans. Math. Software* **41**(2), 11 (2015).
18. Q. Fang and D. A. Boas, "Tetrahedral mesh generation from volumetric binary and gray scale images," in *Proc. IEEE Int. Symp. Biomed. Imaging*, pp. 1142–1145 (2009).
19. S. Yan, A. P. Tran, and Q. Fang, "A dual-mesh Monte Carlo algorithm using a coarse tetrahedral mesh and voxel output," in *OSA Biomed. Opt. Congress*, Paper JTh3A.17 (2018).