Mesh-based Monte Carlo method using fast ray-tracing in Plücker coordinates

Qianqian Fang*
Martinos Center for Biomedical Imaging, Massachusetts General Hospital, 149 13th St, Charlestown, Massachusetts, 02129, USA
*fangq@nmr.mgh.harvard.edu

Abstract: We describe a fast mesh-based Monte Carlo (MC) photon migration algorithm for static and time-resolved imaging in 3D complex media. Compared with previous works using voxel-based media discretization, a mesh-based approach can be more accurate in modeling targets with curved boundaries or locally refined structures. We implement an efficient ray-tracing technique using Plücker Coordinates. The Barycentric coordinates computed from Plücker-formed ray-tracing enables us to use linear Lagrange basis functions to model both media properties and fluence distribution, leading to further improvement in accuracy. The Plücker-coordinate ray-polygon intersection test can be extended to hexahedral or high-order elements. Excellent agreement is found when comparing mesh-based MC with the analytical diffusion model and 3D voxel-based MC code in both homogeneous and heterogeneous cases. Realistic time-resolved imaging results are observed for a complex human brain anatomy using mesh-based MC. We also include multi-threading support in the software and will port it to a graphics processing unit platform in the near future.

©2010 Optical Society of America

OCIS codes: (170.3660) Light propagation in tissues; (170.5280) Photon migration; (170.7050) Turbid media.

References and links
1. B. C. Wilson, and G. Adam, “A Monte Carlo model for the absorption and flux distributions of light in tissue,” Med. Phys. 10(6), 824–830 (1983).
2. P. van der Zee, and D. T. Delpy, “Simulation of the point spread function for light in tissue by a Monte Carlo method,” in Oxygen Transport to Tissue I. A. Silver and A. Silver, eds. (Plenum, New York.), Vol. IX. (1987)
3. Y. Hasegawa, Y. Yamada, M. Tamura, and Y. Nomura, “Monte Carlo simulation of light transmission through living tissues,” Appl. Opt. 30(31), 4515–4520 (1991).
4. M. Hiraoka, M. Firbank, M. Essenpreis, M. Cope, S. R. Arridge, P. van der Zee, and D. T. Delpy, “A Monte Carlo investigation of optical pathlength in inhomogeneous tissue and its application to near-infrared spectroscopy,” Phys. Med. Biol. 38(12), 1859–1876 (1993).
5. L. H. Wang, S. L. Jacques, and L. Q. Zheng, “MCML - Monte Carlo modeling of light transport in multilayered tissues,” Comput. Meth. Prog. Bio. 47(2), 131–146 (1995).
6. J. Ripoll, M. Nieto-Vesperinas, S. R. Arridge, and H. Dehghani, “Boundary conditions for light propagation in diffusive media with nonscattering regions,” J. Opt. Soc. Am. A 17(9), 1671–1681 (2000).
7. S. R. Arridge, H. Dehghani, M. Schweiger, and E. Okada, “The finite element model for the propagation of light in scattering media: a direct method for domains with nonscattering regions,” Med. Phys. 27(1), 252–264 (2000).
8. F. Martelli, D. Contini, A. Taddeucci, and G. Zaccanti, “Photon migration through a turbid slab described by a model based on diffusion approximation. II. Comparison with Monte Carlo results,” Appl. Opt. 36(19), 4600–4612 (1997).
9. Y. Xu, Q. Zhang, and H. Jiang, “Optical image reconstruction of non-scattering and low scattering heterogeneities in turbid media based on the diffusion approximation model,” J. Opt. A, Pure Appl. Opt. 6(1), 29–35 (2004).
10. A. H. Hielscher, R. E. Alcouffe, and R. L. Barbour, “Comparison of finite-difference transport and diffusion calculations for photon migration in homogeneous and heterogeneous tissues,” Phys. Med. Biol. 43(5), 1285–1302 (1998).
11. A. Custo, W. M. Wells 3rd, A. H. Barnett, E. M. Hillman, and D. A. Boas, “Effective scattering coefficient of the cerebral spinal fluid in adult head models for diffuse optical imaging,” Appl. Opt. 45(19), 4747–4755 (2006).
12. G. Ma, J. F. Delorme, P. Gallant, and D. A. Boas, “Comparison of simplified Monte Carlo simulation and diffusion approximation for the fluorescence signal from phantoms with typical mouse tissue optical properties,” Appl. Opt. 46(10), 1686–1692 (2007).

13. D. R. Kirby, and D. T. Delpy, “Parallel operation of Monte Carlo simulations on a diverse network of computers,” Phys. Med. Biol. 42(6), 1203–1208 (1997).

14. A. Colasanti, G. Guida, A. Kisslinger, R. Liuzzi, M. Quarto, P. Riccio, G. Roberti, and F. Villani, “Multiple processor version of a Monte Carlo code for photon transport in turbid media,” Comput. Phys. Commun. 132(1–2), 84–93 (2000).

15. S. R. Arridge, M. Schweiger, M. Hiraoka, and D. T. Delpy, “A finite element approach for modeling photon transport in tissue,” Med. Phys. 20(2 Pt 1), 299–309 (1993).

16. K. Ren, G. S. Abdoulaev, G. Bal, and A. H. Hielscher, “Algorithm for solving the equation of radiative transfer in the frequency domain,” Opt. Lett. 29(6), 578–580 (2004).

17. CGAL, Computational Geometry Algorithms Library, URL: http://www.cgal.org

18. Q. Fang, and D. Bess, “Tetrahedral mesh generation from volumetric binary and gray-scale images,” Proceedings of IEEE International Symposium on Biomedical Imaging 2009, 1142–1145 (2009).

19. T. Binzioni, T. S. Leung, R. Giust, D. Rüfenacht, and A. H. Gandjbakhche, “Light transport in tissue by 3D Monte Carlo: influence of boundary voxelization,” Comput. Methods Programs Biomed. 89(1), 14–23 (2008).

20. D. A. Boas, J. P. Culver, J. J. Stott, and A. K. Dunn, “Three dimensional Monte Carlo code for photon migration through complex heterogeneous media including the adult human head,” Opt. Express 10(3), 159–170 (2002).

21. E. Alerstam, T. Svensson, and S. Andersson-Engels, “Parallel computing with graphics processing units for high-speed Monte Carlo simulation of photon migration,” J. Biomed. Opt. 13(6), 060504 (2008).

22. 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).

23. Y. Fukui, Y. Ajichi, and E. Okada, “Monte Carlo prediction of near-infrared light propagation in realistic adult and neonatal head models,” Appl. Opt. 42(16), 2881–2887 (2003).

24. 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).

25. N. Ren, J. Liang, X. Qu, J. Li, B. Lu, and J. Tian, “GPU-based Monte Carlo simulation for light propagation in complex heterogeneous tissues,” Opt. Express 18(7), 6811–6823 (2010).

26. N. Platis, and T. Theoharis, “Fast ray-tetrahedron intersection using Pliker coordinates,” Journal of Graphics Tools 8(4), 37–48 (2003).

27. H.S.M. Coxeter, Non-Euclidean geometry, Univ. Toronto Press pp. 88–90 (1965)

28. Q. Fang, “Mesh-based Monte Carlo – the software,” URL: http://ncx.sourceforge.net/mmc/

29. H. Shen, and G. Wang, “A tetrahedron-based inhomogeneous Monte Carlo optical simulator,” Phys. Med. Biol. 55(4), 947–962 (2010).

30. M. Meyer, H. Lee, A. Barr, and M. Desbrun, “Generalized Barycentric Coordinates on Irregular Polygons,” Journal of Graphics Tools 7(1), 12–22 (2002).

31. C. A. Felippa, Advanced Finite Element Methods, URL: http://www.colorado.edu/engineering/cas/courses.d/AFEM.d/ProgrammingGuide

32. OpenMP Architecture Review Board, OpenMP Application Program Interface, Version 3.0, URL: http://www.openmp.org/mp-documents/spec30.pdf (2008)

33. NVIDIA, CUDA Compute Unified Device Architecture - Programming Guide, Version 3.0 (2010)

34. Khronos Group, OpenCL 1.0 Specification, URL: http://www.khronos.org/opencl/

35. H. Si, and K. Gaertner, “Meshing piecewise linear complexes by constrained Delaunay tetrahedralizations,” In Proc. of the 14th Int. Meshing Roundtable, pp. 147–163, (2005)

36. P. Fleischmann, B. Haindl, R. Kosik, and S. Seltbergh, “Investigation of a mesh criterion for three-dimensional finite element diffusion simulation,” Proc. Int. Conf. Simulation Semiconductor Processes Devices (SISPAD 99), 71–74 (1999).

37. D. A. Boas, M. A. O’Leary, B. Chance, and A. G. Yodh, “Scattering of diffuse photon density waves by spherical inhomogeneities within turbid media: analytic solution and applications,” Proc. Natl. Acad. Sci. U.S.A. 91(11), 4887–4891 (1994).

38. D. L. Collins, A. P. Zijdenbos, V. Kollokian, J. G. Sled, N. J. Kabani, C. J. Holmes, and A. C. Evans, “Design and construction of a realistic digital brain phantom,” IEEE Trans. Med. Imaging 17(5), 463–468 (1998).

39. J. Vollmer, R. Mencel, and H. Müller, “Improved Laplacian smoothing of noisy surface meshes;” In Proc,” EuroGraphics 99, 131–138 (1999).

40. A. N. Yaroslavsky, P. C. Schulze, I. V. Yaroslavsky, R. Schober, F. Ulrich, and H.-J. Schwarzmair, “Optical properties of selected native and coagulated human brain tissues in vitro in the visible and near infrared spectral range,” Phys. Med. Biol. 47(12), 2059–2073 (2002).

1. Introductions

In bio-optics applications, the Monte Carlo (MC) method is often used to model photon migration inside human tissue [1–5]. Because MC effectively solves the Radiative Transport Equation (RTE) via random sampling, it offers excellent accuracy when simulating photon propagation inside general complex media. In many cases, it was chosen as the gold standard when validating a new algorithm [6–9] or approximation [10–12]. MC has several additional
merits such as being easy-to-program and that it’s straightforward to parallelize [13,14]. Compared with finite-element (FE) [15] diffusion solvers used in many diffuse optical imaging applications, MC produces more accurate solutions, especially when simulating low-scattering media where the diffusion approximation becomes invalid. In many other cases, MC attracts users by avoiding the complex pre-/post-processing steps (such as mesh-generation) required by FE or finite-volume (FV) [16] methods. In particular, the expense of generating high quality 3D meshes for complex media had been a non-trivial task due to the lack of appropriate software tools. Only in recent years, the development of image-based 3D mesh generation software [17,18] had started to make mesh generation from volumetric data practical and easily accessible for general data processing.

The major drawbacks of MC are the low computational efficiency and the lack of ability to accurately model curved boundaries [19]. A traditional MC simulation can easily take several hours or even over a day to obtain a solution with the desired accuracy [11,20]. In the last couple of years, the fast development of multi-core and many-core processors, especially the graphics processing units (GPU), has opened new possibilities for highly efficient MC simulations by exploring massively parallel computing. As a technology preview, Alerstam et al. [21] had shown a 1000x acceleration using GPU in a simple homogeneous domain. More realistic GPU-accelerated MC code for 3D heterogeneous media and time-resolved imaging was recently reported by Fang and Boas [22], obtaining a better than 300x acceleration on a low-cost graphics card compared with single-threaded CPU-based MC simulations. With a modern graphics card, these GPU-based MC codes are capable of producing 3D solutions in less than a minute.

In addition to the dramatic acceleration in speed, significant effort has been made to improve the capability of modeling complex heterogeneous structures. The most widely used MC photon migration code developed by Wang and Jacques, MCML [5], can handle layered media with a circular symmetry. Boas et al. [20] and Fang et al. [22] had developed methods to use a voxelated space to represent arbitrarily complex media, showing great flexibilities in a range of applications [23,11,12]. However, to model regions with curved boundaries with these tools, one has to increase grid density which requires more memory and computation. Margallo-Balbás and French [24] as well as Ren et al. [25] have explored triangular-surface-based MC approaches which allow an improved approximation for domain interfaces. However, to test for ray-surface intersection, one has to scan a range of triangles based on a partitioning scheme, which introduces redundant calculations. In addition, the surface representation is unable to model continuously varying complex media. An algorithm that combines the strengths of MC and FE methods, i.e both accuracy for general media and flexibility for representing arbitrary domain shapes, is highly desired.

Inspired by the works from computer graphics [26], we have developed a mesh-based Monte Carlo method (referred to as MMCM hereafter) by making use of a fast ray-tracing algorithm with Plücker coordinate representation [27]. With this approach, one cannot only model domains with smooth boundaries, but also simulate continuously varying random media, or meshes with complex or mixed element types (tetrahedral, hexahedral, etc). We implemented MMCM in C and multi-threaded programming and developed a software package [28], including the core MMCM simulation code, validation suite, mesh processing scripts and a toolbox to compute analytical diffusion solutions for a heterogeneous sphere. Combined with the open-source 3D mesh-generation toolbox developed previously [18], we expect this software to be an accurate and efficient option for MC simulations in biological tissues.

We also noticed an independent study and software (TIM-OS) recently published by Shen and Wang [29] exploring a similar idea, i.e using tetrahedral meshes for MC simulations. The key differences between TIM-OS and MMCM are 1) MMCM uses Plücker coordinates while TIM-OS uses a Cartesian representation, 2) MMCM allows linear Lagrange basis functions to represent optical properties and fluence continuum while TIM-OS only supports piece-wise constant basis, 3) MMCM can handle more complicated element shapes such as high-order tetrahedral or hexahedral elements, and 4) MMCM can perform both static and time-resolved
simulations. Additionally, we provide rigorous validations using analytical solutions and voxel-based MC for both homogeneous and heterogeneous cases to show the accuracy improvement using the mesh-based approach.

In the remaining portion of the paper, we first present the basic ray-polygon intersection test using Plücker coordinates and describe the algorithm work-flow. Then we discuss extensions of the algorithm such as using linear basis functions, supporting complex mesh elements and parallelization. In the Results section, we validate the algorithm using a homogeneous domain with and without a spherical inclusion, and compare the results with analytical diffusion solutions as well as those from a voxel-based MC simulation; then we show an example of time-resolved simulation from a complex human brain atlas. In the last section, we summarize our findings and discuss further expansion of the algorithm, particularly the massively parallel version running on GPU processors.

2. Methods

2.1 Ray-polygon intersection test using Plücker coordinates

A 3D ray (with a direction) can be uniquely determined by specifying two distinct 3D points, \( p = (x_1, y_1, z_1) \) and \( q = (x_2, y_2, z_2) \). As a result, the Cartesian form of a 3D ray can be typically written as

\[
x = p + (q - p)t
\]

where \( t \) is a scalar. In the Plücker coordinates [27], a 3D ray, \( R \), can be expressed by a vector-pair \( R(d:m) \) where \( d \) represents displacement and \( m \) represents moment, defined as

\[
\begin{align*}
d &= q - p \\
m &= p \times q
\end{align*}
\]

respectively. One of the attractive properties of the Plücker coordinates is its simplicity in a ray-polygon intersection test. In Fig. 1(a), we show a diagram where a ray enters \( (R_1) \), exits \( (R_2) \) or misses \( (R_3) \) a 3D triangle \( (ABC) \). We assume the edges of a 3D triangle \( e_1(U_{e_1}; V_{e_1}), e_2(U_{e_2}; V_{e_2}) \) and \( e_3(U_{e_3}; V_{e_3}) \) are oriented in counter-clock-wise order. A ray \( r(U_r; V_r) \) can be tested for intersection with the triangle by computing the permuted inner products \( w_i \) as

\[
w_i = U_r \cdot V_{e_i} + V_r \cdot U_{e_i}
\]

from which we can assert that

\[
w_i \geq 0 \quad \text{for all } i \rightarrow r \text{ enters the triangle}
\]

\[
w_i \leq 0 \quad \text{for all } i \rightarrow r \text{ exits the triangle (4)}
\]

\[
w_i = 0 \quad \text{for all } i \rightarrow r \text{ is coplanar with the triangle}
\]

else \( r \) misses the triangle

The test in (4) only involves several vector dot-products and tests for signs, thus, it can be very fast. As a by-product from the above test, the Barycentric coordinates [30], \( u_i \), at the intersection point are directly related to the permuted inner products by

\[
u_i = w_i / \sum_i w_i
\]

and the coordinates of the intersection point can be computed by

\[
p = \sum_i u_i p_i,
\]

For a tetrahedron that is known to intersect with a ray, the above test can quickly identify the indices of the faces where the ray enters and exits the tetrahedron; with a few additional
floating-point operations, we can get the coordinates of the entry/exit points. Note that the above test is not limited to tetrahedrons. It can be easily extended to any convex polyhedral elements [31]. A few examples of other commonly used polyhedral elements are shown in Fig. 1(b).

2.2 Ray-tracing based MC algorithm

In MMCM, the heterogeneous media is represented by a 3D volumetric mesh, which can be either a uniform grid (with cubic or cuboid elements) such as the one for voxel-based MC, or an unstructured mesh with polyhedral elements (tetrahedra, hexahedra etc). For a given element in the mesh, the indices of the neighboring elements through each shared face are pre-computed as the “face-neighbor list” and stored in the input files. A source position vector ($s_0$) and an incident directional vector ($c_0$) are specified by the user.

After initializing the mesh information, including nodes, elements and face-neighbor list, an initial intersection test is performed for all elements to identify the initial element ($e_0$) that encloses the source ($e_0$ can also be pre-computed). Starting from $e_0$, we simulate the photon propagation using the well-established MC simulation procedures [5,20,22]. Briefly, we produce a random scattering length and scattering (azimuth and zenith) angles based on the exponential distribution and Henyey-Greenstein phase function [5], respectively, and move each photon along the scattering trajectory recursively. For each element along the path, we perform the above ray-tracing test and calculate the distance ($d$) between the current photon position ($p$) to the exit point ($p_x$) of the enclosing element along the propagation vector ($c$). If $d$ is greater than the remaining scattering length, we will move the photon to the end of the scattering path, and generate a new set of scattering length and angles and repeat the above process. If $d$ is less than the remaining scattering path, we move the photon to the intersection point, $p$, and update the enclosing element ID ($e$) by looking up the face-neighbor list. Then we repeat the above ray-element test for the new element to further propagate the photon. Encountering an empty face-neighbor (denoted by a zero value) indicates a photon passing through a boundary face. We can either reflect the photon at the exiting face of the element based on the Snell’s law, or terminate the photon and launch a new one from the source. The escaped photon energy is stored to an array associated with surface nodes. Because the domain boundary is represented by a surface mesh, MMCM can be more accurate than the boundary reflection scheme developed for the voxelated-space [22].

In order to compute the volumetric fluence inside the domain, we assign an initial photon weight of 1 for each simulated photon and attenuate this weight by its propagation distance using the Beer-Lambert law based on local absorption coefficients [5]. For each ray-polygon test and the subsequent photon movement, we compute the weight loss of the photon and distribute it to the nodes of the interacting element. Because we have access to the Barycentric coordinates of the entry and exit points, we can use the more accurate linear basis functions to store energy deposit/fluence. In our current implementation, we calculate the Barycentric
coordinates at the mid-point between $\mathbf{p}$ and $\mathbf{p}$, and accumulate the energy deposit to the nodes of the enclosing element. Because photon weight decays exponentially with distance, we also provide an option to set a maximum step-size along the path for the accumulation. This can potentially reduce the error due to non-linearity without noticeably increasing the computation time. When the optical properties are defined on the vertices rather than elements, we can also use the computed Barycentric coordinates to quickly interpolate the local optical properties at the current photon position based on the nodal values of the enclosing element. This enables us to simulate continuously varying media which was impossible with previous MC simulations. The final stored energy within the mesh is converted to volumetric fluence by dividing the Veronoi volume at each node ($V_i = \sum_{k,i \in T_k} V_{T_k} / 4$ where $T_k$ is the $k$-th tetrahedron sharing the $i$-th node and $V_{T_k}$ is its volume) and the local absorption coefficient at each node.

2.3. Time-resolved photon migration

MMCM supports time-resolved photon migration using a similar approach as in [20] and [22]. Briefly, we create multiple copies of photon weight array at all nodes for each time gate of the simulation. For each photon, we keep track of the elapsed time of the propagation and accumulate the photon weight loss to the array of the matched time gate. MMCM can produce static solutions when specifying a single time gate with sufficient length. To produce the volumetric fluence distribution, we apply a normalization scheme as in [22] to get the temporal point-spread function, or TPSF.

2.4. Parallelization and optimization

The parallelization of MMCM is surprisingly straightforward. A simple multi-threaded implementation using OpenMP [32] is included in our software. Because the essential part of the calculation in MMCM involves inner and cross products of short vectors, for modern CPUs, this calculation is well suited for Streaming SIMD Extensions (SSE) instructions (inner product is supported in SSE4). Therefore in our implementation, vector operations are written in SSE assembly and can be enabled by compiling switches. These operations are expected to be significantly more efficient on graphics hardware [33]. Further discussion about the massively parallel version of MMCM for the GPU platform using CUDA [33] and OpenCL [34] can be found in the last section.

3. Results

A multi-threaded implementation of MMCM algorithm for multi-core CPU was developed and released under an open-source license [28]. In the current version of the software, we provide support for tetrahedral elements. The extensions to more general elements or mixed types will be added in the future. Two multi-threaded random number generators (RNG) are included: a thread-safe 48bit linear congruential RNG and a Logistic-lattice RNG [22]. In this software package, we also provide a Matlab toolbox to compute the analytical solution for a sphere inside a semi-infinite medium or infinite slab based on diffusion theory.

In this section, we first assess the accuracy improvement for MMCM by comparing its output with the analytical solution from diffusion theory and the output from a voxel-based MC software, Monte Carlo eXtreme (MCX) [22]. We also explore memory utilization and simulation speed with respect to different mesh configurations. Lastly, we show a realistic simulation using complex meshes generated from a human brain atlas. All the computations were performed on a computer running Ubuntu 9.10 (64bit) with an Intel Xeon E5520 (2.3GHz) Quad-core processor. The voxel-based MC simulator, MCX, ran on the same computer with an nVidia GTX 470 GPU. For all the cases, the FE meshes were produced using our previously published 3D mesh generation software “iso2mesh” [18] where part of the meshing tasks were performed by the Computational Geometry Algorithms Library (CGAL) [17] and tetgen [35].

#129767 - $15.00 USD Received 8 Jun 2010; revised 11 Jul 2010; accepted 13 Jul 2010; published 15 Jul 2010
(C) 2010 OSA 2 August 2010 / Vol. 1, No. 1 / BIOMEDICAL OPTICS EXPRESS 170
In the first example, we validate the algorithm using a homogeneous medium in a 60x60x60mm cubic domain. The configuration is similar to that in [20] and [22]: the point source is located at (30.1,30.2,0) mm (the non-integer coordinates are used to avoid simulating photons near the edges/faces of this particular mesh) with an incident direction vector of (0,0,1); the medium has an absorption coefficient $\mu_a = 0.005$/mm, scattering coefficient $\mu_s = 1$/mm, refractive index $n = 1$ and anisotropy $g = 0.01$. We simulate $3 \times 10^7$ photons for a time window of 0 to 5 ns and a resolution of 0.1 ns. For MMCM, we generate a tetrahedral mesh by splitting each 2x2x2 cube in a 60x60x60 grid into 5 tetrahedra (i.e. a T5 mesh [36]). The resulting mesh contains 29791 nodes and 135000 elements.

The temporal response (TPSF) recorded at location (30,14,10) mm is shown in Fig. 2(a). The continuous wave (CW) fluence profiles [20] along plane $y = 30$ mm, represented by contour lines with 10db spacing, are shown in Fig. 2(b). For comparisons, we also plot the analytical solution from the diffusion model [20] and the simulation results from MCX (running for $3 \times 10^8$ photons) in Fig. 2(a).
Fig. 3. Validations of MMCM in heterogeneous media. We show the mesh cross-cut-views and the fluence contour plots (with 10db spacing) of two mesh configurations: (a)-(b) a high-density uniform mesh and (c)-(d) a mesh with higher density at the surface of the spherical inclusion and near source.

In the second example, we changed the background \( \mu_a \) to 0.002/mm and embed a sphere centered at (30,30,30) mm with a radius of 10 mm inside the domain. The optical properties for the sphere are \( \mu_a = 0.05/mm, \mu_s = 5/mm, n = 1.37 \) and \( g = 0.9 \). The source position is set to (30,30,0) mm with an incident vector (0,0,1). We also set \( n = 1.37 \) in the background medium to ignore reflection. Two sets of meshes are generated for this case: 1) a high-density uniform mesh and 2) a mesh with higher density around the sphere boundary and source. The node and element numbers are summarized in Table 1. The cross-cut views of the 2 meshes are shown in Figs. 3(a) and (c). We simulate \( 3 \times 10^8 \) photons with MCX using a 60x60x60 voxel grid, and \( 3 \times 10^7 \) photons for MMCM using the above meshes. By interpolating the MC solutions to a grid using a piecewise-linear basis functions, the contour plots at plane \( y = 30 \) mm are shown in Figs. 3(b) and (d) for the two mesh configurations, respectively. Meanwhile, we compute the analytical solution for a sphere inside an infinite-slab extended from the diffusion models in [37] and overlap it in both figures. Because of the presence of the vertical boundaries, we only show the analytical solution near the source and inside the sphere for the comparison. The speed and memory utilities for MCX and MMCM using various number of CPU cores/threads are summarized in Table 1.

| Table 1. Memory utility and speed comparisons for MMCM and MCX for a heterogeneous simulation. |
|---------------------------------------------------------------|
| Nodes | Elements | Memory for  | Memory for each  | Speed (in photon/ms) vs.  |
|       |         | mesh data (MB) | time-gate (MB) | threads*                |
|---------------------------------------------------------------|
| MMCM Mesh 1 | 63820 | 375753 | 41.3 | 0.240 | 5.6 (1), 10.6 (2), 21.5(4) |
| MMCM Mesh 2 | 20717 | 124992 | 13.7 | 0.079 | 5.6 (1), 10.4 (2), 21.2(4) |
| MCX      | 60x60x60 | -   | 0.82 | 0.820 | ~6700 (thread#>40000) |

*The numbers in the parenthesis are the thread numbers. For MMCM, this equals to the utilized CPU cores. For MCX, we used a thread-block size between 64 and 512.
In the last example, we use MMCM to simulate photon migration in a 3D complex FE mesh generated from a segmented human brain atlas [38,22]. Using the “iso2mesh” toolbox, we first extract four triangular surfaces for scalp, cerebro-spinal fluid (CSF), gray matter and white matter, respectively. Then we apply 4 iterations of the Laplacian + HC smoothing algorithm [39] to each surface, from which we generate a 3D volumetric FE mesh. We configure the mesh algorithm to generate a denser mesh near the top of the head or close to the complex cortex surface. This helps greatly in reducing overall memory demands of the computation. The total number of nodes and elements of the mesh are 69865 and 425224, respectively. The brain tissue optical properties are summarized in Table 2 and are chosen to be identical to those used in [22]. A point source is positioned at (75.7,67.0,168.2) mm pointing toward the center of the head.

In Figs. 4(a) and (b), we show the 3D-cut and the sagittal-cut views of the generated mesh. Again, a total of $3 \times 10^7$ photons are simulated for a time-window of 0-3 ns. The CW and the animated time-resolved solutions (in log-scale) extracted at plane $x = 76$ mm are shown in Figs. 4 (c) and (d), respectively. The computation for mesh generation is under 2 minutes and the MMCM simulation time is 40.5 minutes using 4 CPU cores.

Table 2. Optical parameters in various brain tissue types for the human brain atlas simulation. The properties for scalp/skull and cerebro-spinal fluid (CSF) are based on [11] and those for gray and white matters are based on [40] at 630 nm.

| Tissue types | $\mu_a$ (mm$^{-1}$) | $\mu_s$ (mm$^{-1}$) | Anisotropy (g) | Refract. Index (n) |
|--------------|---------------------|---------------------|----------------|-------------------|
| Scalp & skull| 0.019               | 7.8                 | 0.89           | 1.37              |
| CSF          | 0.004               | 0.009               | 0.89           | 1.37              |
| Gray-matter  | 0.02                | 9.0                 | 0.89           | 1.37              |
| White-matter | 0.08                | 40.9                | 0.84           | 1.37              |
4. Discussion

For simulations in the homogeneous medium, the results from MMCM match well with voxel-based MCX output in both temporal and spatial profiles. In Fig. 2(a), both MCX and MMCM solutions show lower fluence in the later temporal windows compared with the diffusion model. This artifact was caused by the domain truncation due to a finite boundary in the two simulations [22] and disappeared when using a larger volume (not shown). In the heterogeneous case, the MMCM solutions inside the sphere and near the source for both meshes match accurately with the analytical diffusion model. In comparison, the solution from the voxel-space simulation deviates from the analytical/MMCM solutions inside the sphere, confirming that an inaccurate geometric representation of the target may negatively impact the accuracy of the solution [19].

Comparing Figs. 3(b) and 3(d), we find that the two MMCM solutions using different meshes are quite similar. From Table 1, their run-times are also similar. This is expected because the computational expense in MMCM is roughly proportional to the mesh density multiplied by the fluence distribution, not to the absolute numbers of elements or nodes. However, the essential advantage of the second mesh is its memory efficiency. From Table 1, we can see that the total memory required by the second mesh is only 1/3 of the first one. This may have a significant implication when running simulations on a memory-limited device, such as a GPU. Although MCX does not need extra memory to store the basic mesh information, for each additional time-gate, it needs to duplicate the entire grid. By comparison, the memory cost for each additional time-gate for MMCM is very small. For a device with 1GB memory (a typical size for a high-end graphics card), the above difference can translate to over 12000 time-gates using the coarser MMCM mesh compared to only 1200 time-gates using MCX.

Comparing speeds with different CPU cores, we find that MMCM offers excellent parallel acceleration, which is almost proportional to the number of CPU cores. It is not surprising that MCX is a few hundred times faster than MMCM, as we are comparing parallel implementations of different scales and architectures. However, it is reasonable to believe that by porting the MMCM ray-tracing code to CUDA or OpenCL, we can expect to see a comparable acceleration. In addition to running simulations with many parallel threads, the built-in short-vector operations in GPU hardware are highly efficient; this may further improve the speed of MMCM. On the other hand, each ray-tracing step in MMCM requires accessing mesh data (such as node coordinates and elements) from global memory, which may introduce overhead. Fortunately, by running a large number of threads, this latency can be efficiently “hidden” [33]. To further reduce memory access latency, we can also exploit the constant memory in the GPU and optimal ordering of the mesh nodes to improve data caching efficiency.

In the last example, the calculated CW and time-domain solutions generally agree with what we have observed in [22]: photons travel a longer distance in the low-scattering, low-absorption CSF layer, while attenuating rapidly toward the center of the brain. This indicates that both iso2mesh and MMCM are capable of handling real-world complex heterogeneous media.

In summary, we have described a mesh-based Monte Carlo algorithm and developed a free software package that combines the accuracy of MC and the flexibility of FE method for 3D photon migration modeling. We validate this code for both homogeneous and heterogeneous media and show that the algorithm is capable of producing more accurate solutions when simulating objects with curved boundaries.

In the next stage of this research, we will work on a massively parallel implementation of MMCM algorithm using CUDA and OpenCL. This may lead to a dramatic acceleration in speed. We will also extend the pencil beam source model to support more general source and detector forms, such as Gaussian beam or plane illumination. Additionally, we will add supports for hexahedral elements and high-order elements and take further advantage of the flexibility enabled by the mesh-based representations.
Acknowledgement

QF would like to thank the funding support from Harvard Catalyst Pilot Grant. He also would like to thank David Boas for the helpful discussions on the analytical diffusion solutions of a sphere.