Published in final edited form as:

Proc SPIE Int Soc Opt Eng. ; 9412: . doi:10.1117/12.2082349.

NVIDIA OptiX ray-tracing engine as a new tool for modelling medical imaging systems

Jakub Pietrzak^{1,2}, Krzysztof Kacperski¹, and Marek Ciełar^{1,3}

¹The Maria Skłodowska - Curie Memorial Cancer Centre and Institute of Oncology, Warsaw, Poland ²University of Warsaw, Faculty of Physics, Warsaw, Poland ³University of Warsaw, Astronomical Observatory, Warsaw Poland

Abstract

The most accurate technique to model the X- and gamma radiation path through a numerically defined object is the Monte Carlo simulation which follows single photons according to their interaction probabilities. A simplified and much faster approach, which just integrates total interaction probabilities along selected paths, is known as ray tracing. Both techniques are used in medical imaging for simulating real imaging systems and as projectors required in iterative tomographic reconstruction algorithms. These approaches are ready for massive parallel implementation e.g. on Graphics Processing Units (GPU), which can greatly accelerate the computation time at a relatively low cost.

In this paper we describe the application of the NVIDIA OptiX ray-tracing engine, popular in professional graphics and rendering applications, as a new powerful tool for X- and gamma ray-tracing in medical imaging. It allows the implementation of a variety of physical interactions of rays with pixel-, mesh- or nurbs-based objects, and recording any required quantities, like path integrals, interaction sites, deposited energies, and others. Using the OptiX engine we have implemented a code for rapid Monte Carlo simulations of Single Photon Emission Computed Tomography (SPECT) imaging, as well as the ray-tracing projector, which can be used in reconstruction algorithms. The engine generates efficient, scalable and optimized GPU code, ready to run on multi GPU heterogeneous systems. We have compared the results our simulations with the GATE package. With the OptiX engine the computation time of a Monte Carlo simulation can be reduced from days to minutes.

Keywords

SPECT; GPU; Monte Carlo; Medical Imaging

1 Introduction

Graphics Processing Units (GPUs) have been increasingly popular in recent years as an efficient tool for computationally intensive problems [1]. Medical imaging is one of the domains where the need for high computing power often arises as a result of complex image processing algorithms, which potentially lead to higher image quality and diagnostic efficiency [2]. In particular, tomographic image reconstruction using statistical algorithms,

e.g. the maximum likelihood expectation maximisation (MLEM), involves multiple computation of estimated image projections and subsequent backprojection of correction factors obtained by comparison with the measured data [3]. The quality of the reconstructed image improves with the accuracy of the physical model assumed in the numerical projector, however, at the price of increased computing time. Many implementations of iterative reconstruction algorithms in SPECT use projectors based on rotation and convolution. Such approach usually involve some approximations of the physical process of propagation of radiation. More accurate modelling can be obtained using ray tracing, or Monte Carlo simulations of the system. However, the implementation of a gamma ray tracer, which can scale computations for multiple GPU systems and work with different types of objects and images (voxel/blob based, 3D triangle mesh, polynomial representation) is a complicated and time consuming task. Moreover, a code optimized for a particular type of graphic cards may not be easily transferable to another hardware, which can be a nuisance as the GPU technology is advancing rapidly. These difficulties can be to a large extent alleviated by using the NVIDIA OptiX ray-tracing engine, which provides a flexible and versatile platform for developing algorithms involving ray tracing and using the computing potential of the GPUs.

In this paper we present the application of the NVIDIA OptiX ray-tracing engine for accurate modelling of gamma ray propagation in the SPECT imaging system. It can be used as a rapid Monte Carlo simulator as well as accurate ray-tracing projector in iterative reconstruction algorithms.

2 NVIDIA OptiX ray-tracing engine and its application in medical imaging

The NVIDIA OptiX ray-racing engine [4] is a programmable system designed for highly parallel architectures build on the key observation, that all ray-racing based algorithms may be represented as a set of similar components. OptiX library is equipped with a low level engine focused on the fundamental computations required for the ray-tracing. The engine provides a mechanism for coding interactions of rays with a predefined set of geometry objects without referring to concepts specific to computer graphics, like lights, shadows, reflections etc. The whole physics of the ray-object interactions is left to be implemented by the user. OptiX also provides an abstract ray execution model as a sequence of small user-defined programs. The execution model, combined with data stored in buffers (e.g. object geometry and properties, interaction coefficients, etc.) and additional data stored in per-ray data structures (e.g. photon energy, interaction types and sites), can be used for implementation of various types of ray-tracing based algorithms from the standard rendering algorithms to more sophisticated applications like aerodynamics, thermodynamics or X- and gamma-ray transmission. OptiX engine is fully compatible with CUDA, OpenGL and DirectX technologies and OptiX buffers may be shared for example with OpenGL for real time display or with CUDA for parallel post processing like filtering.

There are seven types of programs (user defined functions), each of which operates on a single ray, or on geometric objects. The combination of user defined functions and OptiX kernel code creates the ray tracing pipeline. The description of all program types can be found in [1]. From the medical imaging point of view the most important programs are:

1. The ray generation program. This is the entry point to the ray-tracing pipeline. Ray generation program determines the starting point and the direction of the ray and the initial value of the per-ray data. Single invocation of Optix engine launch function (`rtContextLaunch`) creates many instances of ray generators. In medical imaging the most natural approach would be to generate the rays from sources and track them to the detector. This would be the way to run full Monte Carlo simulations. However, in the case of "reversible" rays, when no branching is involved (e.g. modelling of attenuation only, without scatter) generating rays from the detector and following them to the sources, can be much more efficient, particularly in the case of spatially extended activity distributions in emission tomography.

2. Intersection program implements the ray-geometry object intersection tests and can provide the position of collision point, both absolute and relative to a particular object, the vector normal to surfaces at intersection point, or other attributes basing on the hit point position. Intersection program gives the user the possibility to implement intersections with various types of objects like planes, triangle meshes, surfaces, spheres, high order surfaces, or even fractal geometries. The intersection program contains the intersection test fully implemented by OptiX user, so for example there is a possibility to use different types of phantoms represented by simple geometric primitives (e.g. voxels), triangle-mesh, or polynomials. Moreover, there is also a possibility to include complex objects created in e.g. CAD software. This may be useful to describe collimators used in nuclear medicine.

3. Material programs are the functions associated with the geometric objects called when an intersection is reported by the intersection program. These programs, called Closest Hit Programs, implement the physics of ray interactions. The call of material program can stop the ray (absorption), recursively execute new ray with the set of new parameters like ray start position and direction (scatter), as well as compute the values of per ray data (interaction site, energy change, etc.). In standard rendering algorithms material programs are responsible for computing ray parameters depending on material, like color, attenuation, reflection and transmission (for example by implementing Snell's law). The physics of X- and gamma ray propagation in materials is different from that of light around the visible spectrum in that most of the physical phenomena occur in the bulk of the material rather than on surfaces. The material functions allow the implementation of phenomena like photoelectric absorption, Compton and elastic scatter using the appropriate material coefficients stored in structures associated with particular geometry objects.

4. Bounding box programs are functions for determining rectangular boxes containing geometry primitives. These user-defined functions enable OptiX engine to build the volume box hierarchy to accelerate the execution of intersection programs: they are only called on objects included in boxes representing a particular branch in the hierarchy tree. The user can change between several types of acceleration structures like variations of Bounding Volume Hierarchy (BVH) trees or Triangle Kd-Tree (dedicated for a mesh-based object). Recomputed acceleration structure allows OptiX to select the objects (e.g. triangles in triangle mesh) on which the intersection test should be performed.

All the data and functions used by OptiX engine are stored in a tree-hierarchical container object called Context, which provides the mechanism for binding programs, objects and data in a hierarchy graph describing the simulated scene. Each object in the scene is usually represented as a Geometry Instance node. Geometry Instances which share some properties (e.g. intersection function, material, acceleration structure) can be grouped in Geometry Groups. The OptiX programs described above are ascribed to Geometry Instances or Groups, defining their geometric properties and interactions with rays. Moreover, geometric transformations of translation and rotation can be performed on any object, or group of objects to model their relative motion. Any element of the Context can have access to buffer data storing the information about the geometry and physical properties of objects

3 Monte Carlo simulation of a parallel hole collimator point spread function (PSF)

We have developed an OptiX based code for simulating projections measured by gamma cameras with collimators of any type shape and size. In particular, with point sources we can obtain precise and detailed point response functions for any chosen collimator parameters. The collimator has been defined in form of triangular mesh, which can be prepared in any 3D modelling software and saved into Wavefront OBJ format. Photoelectric absorption, inelastic and elastic scatter is simulated, X-ray fluorescence is ignored since it can only contribute to low energy windows, e.g. as for ^{201}Tl . We show examples of simulated PSFs obtained with $^{99\text{m}}\text{Tc}$ point sources for four low energy parallel hole collimators, as well as PSFs obtained for ^{131}I for high energy collimator. Collimator parameters are listed in Table 1.

2×10^9 photons emitted into a solid angle subtended by the detectors have been simulated. We assumed a perfect detector without any photon position or energy blurring. For the random number generation we use cuRand library, which generates random numbers on GPU into the buffers accessible by OptiX. The code has been run on a GTX680 graphics card. For comparison similar simulations have been done using the GATE software running on Intel Core i7 930 processor.

4 Results

Table 2 shows examples of the point spread functions obtained for the specified collimators with a point source at 16 cm from the center of the detector surface. The particular types of interactions can be easily switched on and off to balance the accuracy of the obtained result against the computing time as well as to illustrate the effect of different physical processes.

Table 3 shows the results of simulations for HE collimator for three energies. The images agree very well with the corresponding results obtained with the GATE software, used here as a reference. Table 4 compares the simulation times. Acceleration factor over 500 has been obtained for the pure Monte Carlo simulation, without using any variance reduction techniques. This is increased by one more order of magnitude if collimator scatter is not modelled, which corresponds to the ray-tracing projector. Note, however, that GATE is not designed for such purpose.

5 Discussion

Massive computing power provided by the GPU technology is making an impact on the medical imaging. Complex algorithms known in research domain are becoming feasible for regular clinical use. However, the difficulty of code implementation remains an issue. Significant effort is needed to find the way to keep all available GPUs busy most of the time, to minimize memory transfers (or to hide them behind the streams), to use properly cached data structures to avoid non-coalescent memory read and write, and finally to adapt software to optimal use of multi-GPU hardware. By using the OptiX engine most of these problems are sorted out automatically. The engine determines the number of using parallel processors, schedules the tasks in the optimal way to keep all available processors busy. The user can just focus on the particular algorithm or simulation setup.

OptiX can be used for a number of applications in medical imaging: accurate ray-tracing based projectors for iterative tomographic reconstruction algorithms, both for emission (SPECT, PET) and transmission (CT) tomography, fast detailed Monte Carlo simulators of X- and gamma ray propagation in patients, collimators, detectors, etc., simulating propagation of light in scintillation crystals, dosimetry, and many others.

In contrast to low level CUDA programming, the OptiX code is clear, flexible, and easy to modify and develop on.

References

- [1]. Nickolls J, Dally WJ. The GPU computing era. *IEEE Micro*. 2010; 30(2):56–69.
- [2]. Pratz G, Xing L. GPU computing in medical physics: A review. *Medical Physics*. 2011; 38(5): 2685. [PubMed: 21776805]
- [3]. Kinouchi S, et al. GPU-Based PET Image Reconstruction Using an Accurate Geometrical System Model. *IEEE Trans Nucl Sci*. 2012; 59(5):1977.
- [4]. Parker, Steven G. OptiX: A General Purpose Ray Tracing Engine. *ACM Transactions on Graphics*, SIGGRAPH 2010;
- [5]. Jan S, Santin G. GATE: a simulation toolkit for PET and SPECT. *Physics in Medicine and Biology*. 2004; 49(19)

Table 1

Parameters of the simulated parallel hole collimators

Collimator	Hole pattern	Hole diameter [mm]	Hole length [mm]	Septal thickness [mm]	Size
High Resolution (HR)	square	1.11	24.05	0.16	30 × 30 cm
High Resolution (HR)	hexagonal	1.11	24.05	0.16	30 × 30 cm
High Sensitivity (HS)	square	2.54	24.05	0.36	29 × 29 cm
High Sensitivity (HS)	hexagonal	2.54	24.05	0.36	29 × 29 cm
High Energy (HE)	square	4.50	65.0	1.9	32 × 32 cm

Table 2

Point spread functions at 16 cm from detector surface simulated for 2×10^9 140 keV photons, for HR, HR hexagonal, HS, HS hexagonal collimators. First row shows Point Spread Function as summarized effect of all simulated interactions. Next rows represent: geometric photons only, septal penetration, coherent scattering and incoherent scattering.

Table 3

Point Spread Function simulated for the HE collimator for 2×10^9 photons of energy: 284 keV, 364.5 keV and 637 keV. Pictures show the summarized result for all kinds of simulated interactions.

Table 4

Simulation times for the HR collimator.

Simulation	Duration time		Speed-up factor
	OptiX	GATE	
Full Monte Carlo	5.5 min.	49.86 hrs	544
Geometric and penetration only	0.32 min.	44.19 hrs	8286