Accelerating Monte Carlo Photon Transport Using Massively Parallel GPUs

Jonas Lippuner [1, 2] and Idris A. Elbakri [2, 3]

[1)] Department of Mathematics, University of Manitoba [2)] Department of Physics and Astronomy, University of Manitoba
[3)] Division of Medical Physics, CancerCare Manitoba

Introduction

Background: Monte Carlo simulations are widely used to simulate the effects of ionizing radiation in medical applications, such as radiation therapy and x-ray imaging. They usually produce highly accurate results at the cost of extremely long computation times (several days are not uncommon).

Goal: Implementing an existing Monte Carlo simulation (EGSnrc) in a parallel fashion for GPUs, in order to reduce the computation time while completely maintaining the accuracy.

Method: Nvidia's CUDA is used for the implementation and the simulation itself is divided into different steps that will be executed in parallel.

Monte Carlo Method

The Monte Carlo method is a powerful method using repeated random sampling to simulate complex systems with many degrees of freedom and uncertainties in the initial conditions. Monte Carlo simulations produce very accurate approximations and are often used when an exact solution is difficult or impossible to calculate with a deterministic algorithm.

Monte Carlo simulations usually include the following steps:

1. Randomly generate a set of input parameters based on known (empirically determined) probability distributions.
2. Perform deterministic computations with the random input. The result of one set of inputs is called a *history*.
3. Repeat the above steps to calculate many histories and aggregate them to get the final result.

CUDA

CUDA is a parallel computing architecture developed by Nvidia to run programs on Graphics Processing Units (GPUs). GPUs consist of hundreds of cores and can execute thousands of instances of a function, called *threads*, simultaneously. A caveat is that groups of 32 threads, called a *warp*, must always perform exactly the same calculation, otherwise the performance may be drastically reduced, which is known as *warp divergence*.

Simulation

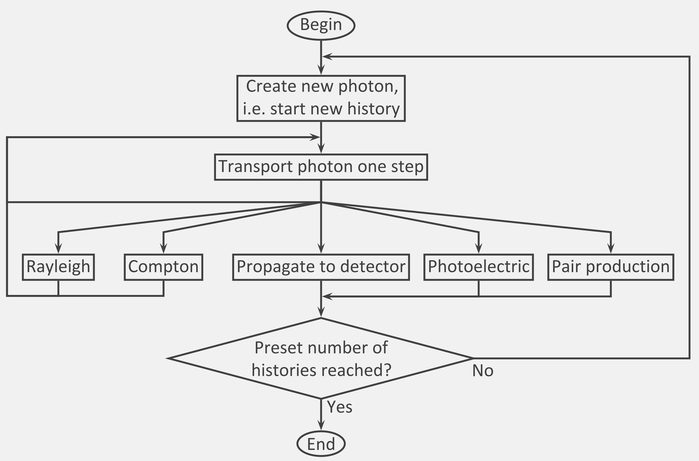
EGSnrc is a Monte Carlo simulation package for coupled electron-photon transport through any material in an arbitrary geometry. It is widely used for dose estimation in radiation therapy or to simulate x-ray scattering arising in imaging applications.

The simulation presented here is a reimplementation of the photon transport mechanism of EGSnrc, electrons are completely ignored. The simulation consists of an x-ray point source, a phantom (voxelized volume) and a detector. The result is an image of the energy fluence in each pixel of the detector. Separate images for primary and scattered photons can be created.

Implementation

Since the development of individual histories is determined by random numbers, the threads would diverge very quickly if each thread ran one complete history. Therefore the simulation was divided into different steps. The flow chart below shows the different steps and how they relate to each other.

The execution paths of the steps diverge only minimally, but the order of the steps varies for each history. To avoid warp divergence, the threads in a warp go through the possible steps together and only those threads that need to perform the current step perform it while the others wait. This is repeated so that each thread performs many steps thus simulating many histories.



Flow chart of the simulation with the different steps.

Speed Up

The same simulation was run with the original EGSnrc and the CUDA implementation for different number of voxels N_{vox}.

$N_{\text{vox}}^{1/3}$	hist/sec EGSnrc	hist/sec CUDA	Speed up
64	$161{,}000 \pm 2{,}000$	$3{,}265{,}000 \pm 6{,}000$	$\mathbf{20.2 \pm 0.2}$
128	$79{,}000 \pm 3{,}000$	$2{,}138{,}700 \pm 700$	$\mathbf{27 \pm 1}$
192	$44{,}100 \pm 600$	$1{,}608{,}000 \pm 4{,}000$	$\mathbf{36.4 \pm 0.5}$
256	$31{,}600 \pm 400$	$1{,}291{,}000 \pm 2{,}000$	$\mathbf{40.8 \pm 0.5}$

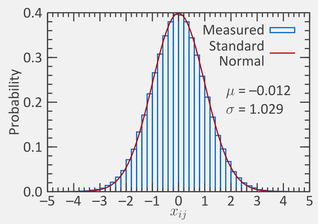
The errors represent one standard deviation.

Accuracy

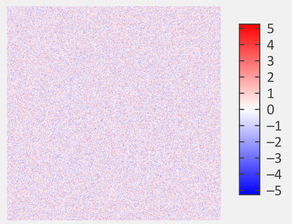
The normalized difference between the results in pixel (i, j) is

$$x_{ij} = \frac{E_{ij}^{\text{CUDA}} - E_{ij}^{\text{EGSnrc}}}{\sqrt{\left(\sigma_{ij}^{\text{CUDA}}\right)^2 + \left(\sigma_{ij}^{\text{EGSnrc}}\right)^2}}$$

Where E_{ij} is the value and σ_{ij} the standard deviation. If the differences are purely statistical, x_{ij} should be normally distributed.



Normalized histogram of the measured distribution of x_{ij}.



The differences x_{ij} between the CUDA and EGSnrc results.

Conclusion

Typical resolutions in clinical applications are between 64^3 and 128^3 voxels. Hence a speed up between 20 and 27 times can be expected. The distribution of x_{ij} is very close to normal and the different values of x_{ij} seem to be uniformly distributed across the detector. Therefore, there are no or only insignificant systematic differences between the results of CUDA and EGSnrc.