Stony Brook University

GPU-Acceleration of X-ray Photon Scattering Simulation

A Thesis Presented

by

Jaewoo Pi

to

The Graduate School

in Partial Fulfillment of the

Requirements

for the Degree of

Master of Science

in

Computer Science

Stony Brook University

December 2014

Stony Brook University

The Graduate School

Jaewoo Pi

We, the thesis committee for the above candidate for the

Master of Science degree, hereby recommend

acceptance of this thesis.

Klaus Mueller
Professor, Department of Computer Science

Pradipta De
Assistant Professor, Department of Computer Science

Ilchul Yoon
Assistant Professor, Department of Computer Science

This thesis is accepted by the Graduate School

Charles Taber
Dean of the Graduate School

GPU-Acceleration of X-ray Photon Scattering Simulation

by

Jaewoo Pi

Master of Science

in

Computer Science

Stony Brook University

2014

Computed Tomography is widely used in numerous fields. To facilitate the development of an algorithm while avoiding additional radiation dose to the patient, the simulation of the CT imaging process is required. The scattering effect of X-ray photons impact on quality of output images. Thus, modeling the scattering is an important task to predict and eliminate the effects on energy. However, the simulation has been challenging task. Two approaches, deterministic and Monte Carlo method has been widely used, but both methods have to undergo trade-off between speed and sampling rate. In this research, the new algorithm which exploits Bi-directional ray tracing technique, is suggested and implemented. The experimental program is focused specifically on parallelism on general-purpose GPUs. With the optimization, the result shows promising mark.

“All forward motion counts.”

- Hollie Baylor

Table of Contents

Pages

List of Figures/Tables/Illustrations

Pages

List of Abbreviations

CT - Computated Tomography

GPU - Graphics Processing Unit

GPGPU - General-Purpose Graphics Processing Unit

MRI - Magnetic Resonance Imaging

PET - Position Emission Tomography

RoI - Region of Interest

CUDA - Computer Unified Device Architecture

AMD – Advanced Micro Devices

Acknowledgments

I do appreciate Dr. Klaus Mueller, for his invaluable guidance. Thanks to my family for infinite support. Thanks to colleagues from department of Computer Science at SUNY Korea for professional discussions, especially Dr. Pradipta De and Dr. Ilchul Yoon, Arani Bhattacharya, Darius Coelho, Tan Le, and Quoc Duy Vo. Thank you colleagues at Visual Analytics and Imaging lab at Stony Brook University for all their input and ideas.

Chapter 1

Introduction

In Computed Tomography (CT), also known as Computed-Aided Tomography (CAT) is used for patient diagnosis in hospital, and also used for airport baggage security and for nondestructive evaluation in manufacturing [1]. In recent years scientists in the field of x-ray tomography rely on the simulation, and the sophistication of tomographic reconstruction algorithm has improved dramatically.

The effort of reducing radiation exposure is another important issue in medical imaging. To facilitate the development of new algorithms while avoiding additional radiation dose to the patient, the simulation of the imaging process is required. The scatter effect of X-ray photons significantly impact to quality of output images. More specifically, it impacts CT number inaccuracy, contrast difference, and more artifacts that hinders from getting an accurate output images.

In early 70's, it has been realized that the Compton Effect may give rise to new challenging imaging modalities. Compton scattered radiation behaves as noise hindering image quality. Modeling this scattering effect for the simulation has been challenging task [2]. Much sophisticated model is needed to model and simulate such scattering effects, and eventually minimize the effect of scattering

on output images. However, more sophisticated model led high computational cost.

The rise of General-Purpose GPU shed a light on reducing computation time, by adopting parallelism. It is relatively lower cost than numerous multi-core systems, still enables running program in parallel for computer graphics, and various scientific simulations. The GPUs, however, has unique hardware architecture, and a developer have to have understandings on internal structure.

The main goal of this research is to achieve fast, yet considering scattering effects from all samples. The remainder of thesis is organized as following: chapter 2 shows related research that have been done to achieve same or similar goals, chapter 3 explains background knowledge and theories about CT image formation, X-ray photon scattering effect, bi-directional ray tracing algorithm, followed by the architectural properties of GPU, and methodology of GPU utilization for bi-directional ray tracing. Chapter 4 shows system configuration for the simulation, and chapter 5 depicts implementation details with the result of experiments. Finally, chapter 6 sums up and discuss about the result and possible future research.

Chapter 2

Related Research

There are numerous computer codes for simulation based on ray tracing technique and on the X-ray attenuation, developed by a number of research groups. In general, two methods are widely used. Conventionally, the ray tracing algorithm is deterministic which accounts for attenuation from direct ray. This method is fast, but does not take into account of indirect photons. The other is Monte Carlo method, which is used at one of the well-developed simulations Duvauchelle et al. [3]. Their software includes test chain to reduce the number of experimental tests. However, a major drawback of Monte Carlo simulation is the extensive expenditure of computation time.

Several research groups have proposed hybrid approaches by adding stochastic properties into the deterministic approaches, to achieve physical accuracy [4][5]. Parallel implementation of scattering simulations have also been introduced to overcome tremendous amount of computation cost [6][7]. However, both of these methods have to confront speed and accuracy trade-off.

In this research, we developed an algorithm that simulates first-order scattering of X-ray photons. The idea shares common property with the global illumination problem in Computer Graphics; simulating photons which collide and are reflected, absolved, refracted. Traditional ray tracing algorithm build the

ray around the importance of the viewing point. Progressive radiosity method, in contrast, emphasize more contributions of the light sources. Bi-directional ray tracing method is the idea of shooting and gathering power to create photorealistic images [8]. This method performs better than single-directional ray tracing, however still utilizes Monte Carlo method and thus has trade-offs between over-solving (more accuracy, less speed), and less-solving (less accuracy, more speed).

In general, transporting a CPU-based algorithm to the GPU is not straightforward. It requires a new set of algorithms that involves a deep understanding of the GPU architecture and its programming model, and fitting the physical phenomenon into the General-Purpose GPU parallelism. The new algorithm that suggested in the rest of the contents has achieved higher performance in terms of both speed, yet not sacrificing the sample rate.

Chapter 3

Theories and Background

3.1. CT Image Formation

Almost all diagnostic imaging techniques, such as X-ray computed tomography (CT), magnetic resonance imaging (MRI), and position emission tomography (PET) rely on tomographic reconstruction. The formation of a digitized image utilizes image reconstruction algorithm from the raw data acquired from CT detectors. Figure 3-1 depicts the procedure of generating cross-sectional image from CT machine. The detector and the x-ray cone beam projector rotates around the target object, and casted photons travel through the object. The photons loses energy as they penetrate through the matters, called attenuation coefficient, and denoted as **μ**.

In reality, the attenuation is measured by CT number, which is normalize value compared to attenuation of water. The following formula calculates CT number, which is referred to as Hounsfield Unit (HU):

$$HU_x = \frac{\mu_x - \mu_{water}}{\mu_{water}} \text{ x } 1000. \qquad \text{Eq. (1)}$$

Table 1 gives a section of tissues with their corresponding CT numbers [9]. Note that the bigger number indicates more attenuation and less number means that the matter allows more photons to penetrate.

Tissue	CT Number, approx.
Dense Bone	1000+
Muscle	10-40
Liver	40-60
Blood	40
Kidney	30
Water	0
Fat	-50 to -100
Air	-1000+

Table 1. Tissues with their corresponding CT numbers

In 2-D case, Radon transform $\boldsymbol{R\mu(\theta, s)}$ is defined as line integral of function $\boldsymbol{\mu(x, y)}$ along a line $\boldsymbol{L}$, inclined at angle $\boldsymbol{\theta}$, at distance $\boldsymbol{s}$ from the origin. This is formally defined as:

$$\boldsymbol{R\mu(\theta, s)} = \int_{\boldsymbol{L}} \boldsymbol{\mu(x, y)du}$$

$$= \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} \boldsymbol{\mu(x, y)\delta(x\cos\theta + y\sin\theta - s)dxdy}, \qquad \textbf{Eq. (2)}$$

where $\boldsymbol{\delta(x)}$ denotes Dirac's delta function [10].

A function $\boldsymbol{\mu(x, y)}$ denotes the distribution of the X-ray attenuation coefficient within the object.

The photon Intensities $\mathbf{I_j}$ at the detector bin positions **j** are preprocessed to projection data $\boldsymbol{p_j}$, which is:

$$\boldsymbol{p_j} = -\ln\frac{\boldsymbol{I_j}}{\boldsymbol{I_0}} = \int_{\boldsymbol{L}} \boldsymbol{\mu}(\boldsymbol{x},\boldsymbol{y})\boldsymbol{dxdy}, \qquad \textbf{Eq. (3)}$$

and the X-ray photons traveling along line $\boldsymbol{L}$ is attenuated by the object according to Beer's law for the photon intensities $\boldsymbol{I_j}$ can be formally defined as:

$$\boldsymbol{I_j} = \boldsymbol{I_0}\exp\left(-\int_{\boldsymbol{L}} \boldsymbol{\mu}(\boldsymbol{x},\boldsymbol{y})\,\boldsymbol{dxdy}\right), \qquad \textbf{Eq. (4)}$$

where $\boldsymbol{I_0}$ is the initial photon intensity.

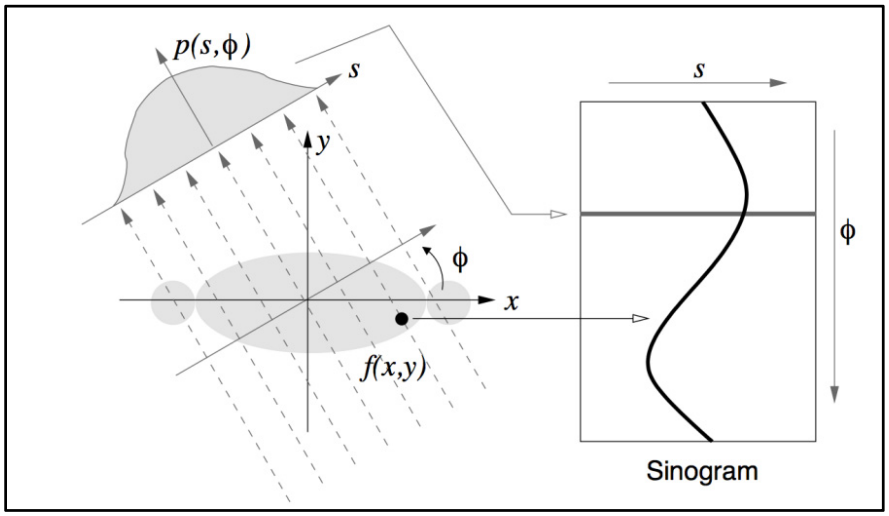

Figure 3.1. Formation of Sinogram

Reconstruction in tomography means a recovery from samples of Radon transform of an unknown object density distribution. The goal of CT imaging is to reconstruct $\boldsymbol{\mu}(\boldsymbol{x},\boldsymbol{y})$ from the projections of the object. Inverse radon transform is

the basic tool to perform the reconstruction, however, there needs much more sophisticated method to reconstruct.

In summary, Figure 3.2 depicts the overall process of the CT image acquisition. Section 3.2 covers the scattering effect in progress of getting sinogram.

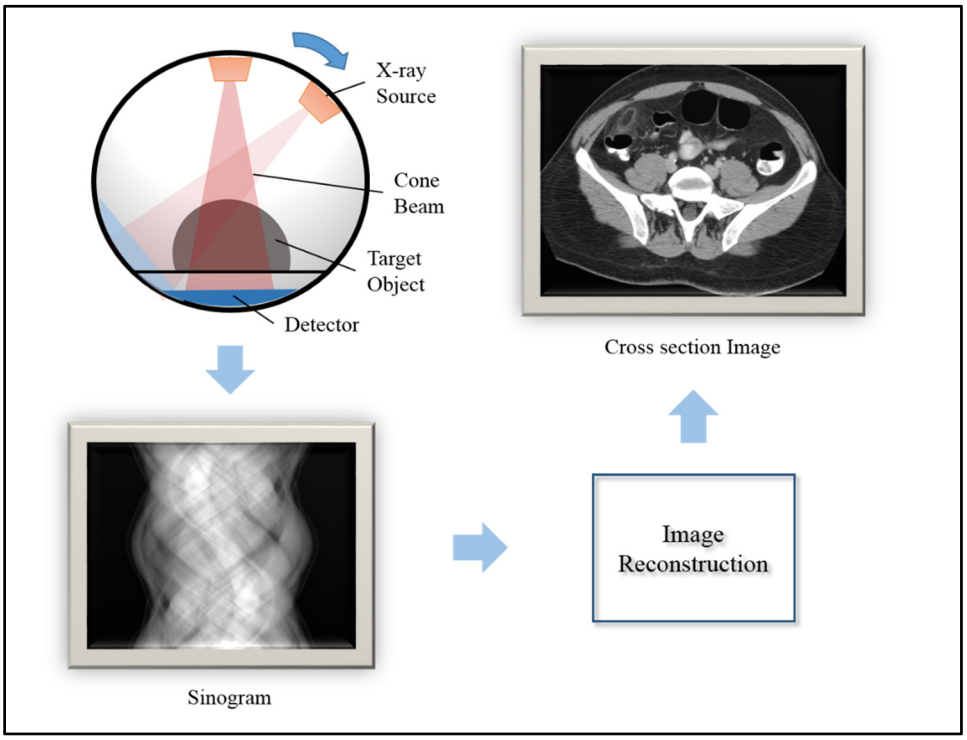

Figure 3.2. Overall CT Image acquisition process

3.2. The Scattering Effect

In early 70's it has been realized that the Compton Effect may give rise to new challenging imaging modalities. Compton scattered radiation behaves as noise hindering image quality. Compton Effect is the scattering of X-photons with electric charges. The energy of a scattered photon is related to the scattering angle ω can be formally defined as:

$$E_w = \frac{E_0}{1 + \frac{E_0}{mc^2}(1 - \cos\omega)} \qquad \textbf{Eq.}\ (5)$$

where E_0 is the emitted photon energy and mc^2 denotes the energy of an electron at rest (0.511MeV). Figure 3.3 below depicts the principle of Compton Effects [1]. As a result, a scattered photon steers its direction with lowered energy.

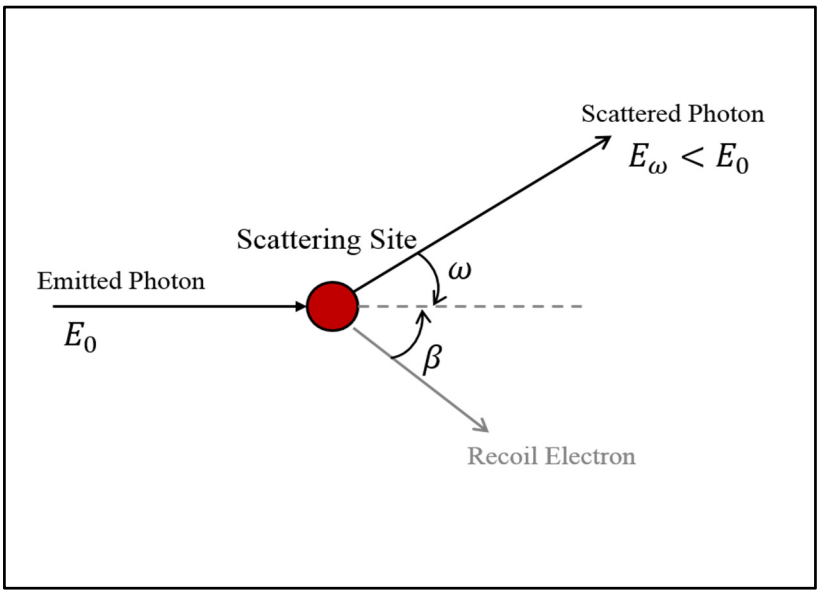

Figure 3.3. Principle of Compton Scattering

The Compton scattering equation is correlated to the mass of electron, initial and final wavelength, and scattering angle **ω**. In other words, different materials and the intensity of initial energy from the X-ray source decides the scattering angle and final energy intensity. This relation is formally given by Klein-Nishina formula. Figure 3.4 shows the Klein-Nishina distribution of scattering-angle cross sections over a range of commonly encountered energies. Note that the diameter of the circular plot indicates ratio of photon energy after and before the collision [11].

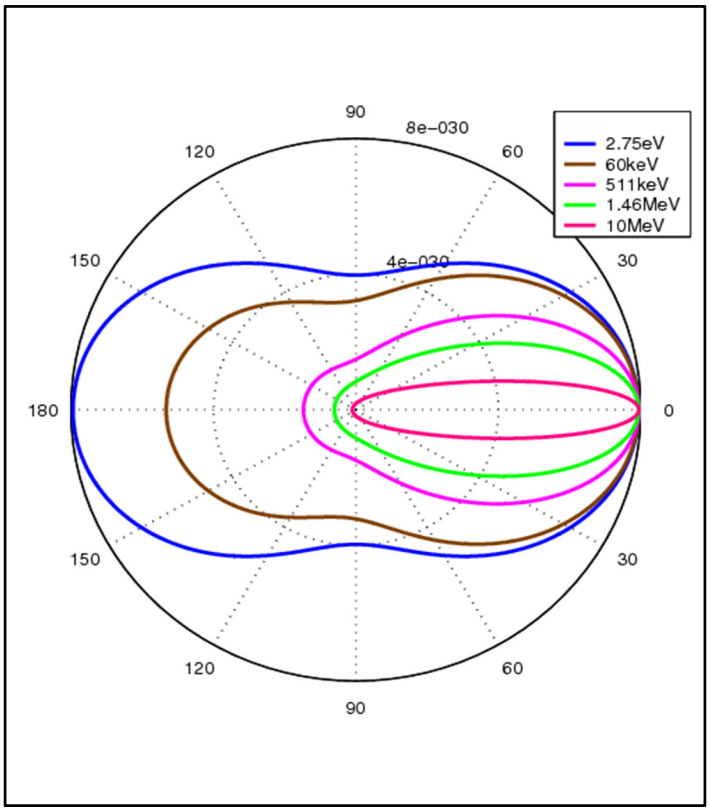

Figure 3.4. Klein-Nishina Distribution

3.3. Bi-direction Ray Tracing

Originally, ray tracing has been widely used technique for global illumination in computer graphics. The goal of global illumination is to find a relevant set of virtual point light sources to illuminate the object seen by camera [12]. Formally, solving the global illumination problem leads to evaluate the following equation:

$$\boldsymbol{L}(\mathbf{x}, \boldsymbol{\omega}_{\mathbf{e}}) = \boldsymbol{L}_{\boldsymbol{p}}(\boldsymbol{x}, \boldsymbol{\omega}_{\boldsymbol{e}}) + \int_{\Omega} \boldsymbol{L}(\boldsymbol{x}, \boldsymbol{\omega}_{\boldsymbol{i}}) \boldsymbol{f}_{\boldsymbol{r}}(\boldsymbol{x}, \boldsymbol{w}_{\boldsymbol{i}}, \boldsymbol{w}_{\boldsymbol{e}}) \cos(\boldsymbol{\theta}_{\boldsymbol{i}})\, \boldsymbol{d\omega}_{\boldsymbol{i}} \qquad \textbf{Eq. (6)}$$

where $\boldsymbol{L}(\mathbf{x}, \boldsymbol{\omega}_{\mathbf{e}})$ is the incoming radiance from point x in direction $\boldsymbol{\omega}_{\mathbf{e}}$, $\boldsymbol{L}_{\boldsymbol{p}}(\boldsymbol{x}, \boldsymbol{\omega}_{\boldsymbol{e}})$ is the emission from point x in direction $\boldsymbol{\omega}_{\mathbf{e}}$, $\boldsymbol{L}(\boldsymbol{x}, \boldsymbol{\omega}_{\boldsymbol{i}})$ is the outgoing radiance at point x in direction $\boldsymbol{\omega}_{\mathbf{e}}$, and $\boldsymbol{f}_{\boldsymbol{r}}(\boldsymbol{x}, \boldsymbol{w}_{\boldsymbol{i}}, \boldsymbol{w}_{\boldsymbol{e}})$ is the bi-directional reflectance distribution function (BRDF) of the material at point x.

Bi-direction ray tracing technique, also called 'two-pass' ray tracing has been originated from the concept of caching illumination. Two paths use cached illumination maps of each other's so that the computation cost can be dramatically reduced. Bidirectional ray tracing technique combined with Monte Carlo method has been developed for faster computation cost and consideration of all kind of indirect illumination [13]. The fundamental idea is that rays or photons are cast simultaneously at selected light source and viewing points. More specifically, figure 3.5 depicts that the virtual ray creates the sub-paths from both light source

and detector (or camera) at random points. The source *S* cast ray to the random point p_1, and it continuously cast to the random point to reach p_2. At the same time, *D* cast a ray in same way as light source does, and reach to p_3. A complete path is created by concatenating these two paths. From the randomness of the technique, not all casted rays generate full-paths from the light source to the detector. For example, a sub-path connecting p_2 and p_3 may not be generated [14], and will not be contributed for the image. Suppose there is connected path between the light source and the detector, the tracing ray algorithm accumulates the effects of reflection, refraction, and absorption. As a result, the detector (eye) knows what brightness value on the object surface.

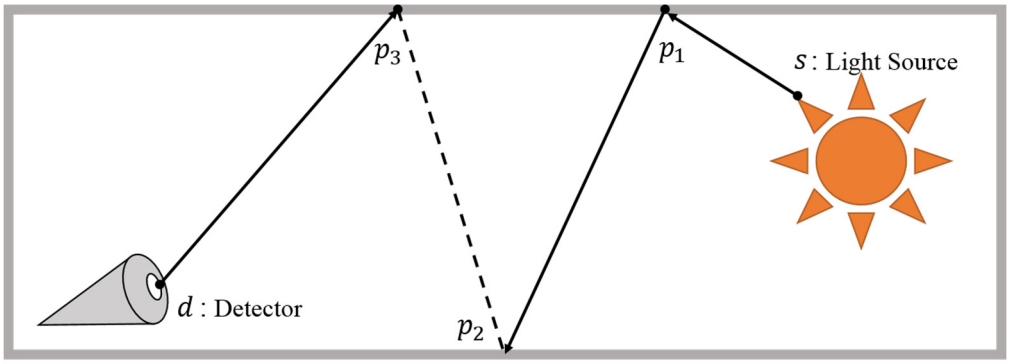

Figure 3.5. Bi-directional ray tracing

With this technique, the system can go into diverse types of geometrical objects. Furthermore, diffuse lighting effects, soft shadows, specular and glossy reflection and refraction can be simulated with much more accuracy [8].

3.4. General-Purpose GPU Utilization

Conventionally improving performance of computers is attained by CPU clock rate increment. However, the physical components of processors are reaching physical limitation of circuit density and power consumption. Thus Moore's Law, which states the speed of processors doubles every eighteen months, now implies not only clock rate increment, but also increasing concurrent execution with efficient communication among parallel cores.

Typical GPUs contains hundreds of multiprocessing core. The throughput of a high-end graphics card is on the order of four teraflops per second. They are designed to perform calculations on large amounts of independent data. In recent trends, General-purpose Graphical Processing Unit (GPGPU) has been widely used not only for computer graphics, but also mathematical and scientific computing. GPGPU code executes massively parallel on personal computers.

The largest graphics card manufacturers provide software development kits for programming on their GPUs. OpenCL framework is for writing program that execute across heterogeneous platforms, such as GPUs, DSPs, or FPGAs. OpenCL has been adopted by various chip vendors, and their use of GPU for general purpose is currently in stable release. However NVIDIA's CUDA is currently dominant in the market [15].

CUDA (Computer Unified Device Architecture) is a C-like API used to program the NVIDIA GPUs. It supports single program multiple data (SIMD), which means one set of instruction is executed by many threads. A CUDA program begins with initialization of variables and memory allocation from host (CPU) to device (GPU). Typically, the GPU cannot access the memory of the CPU, and special operation is needed to copy data between host and device.

Execution of a task by a CUDA kernel is organized into thread blocks. Thread blocks are organized into grids. The GPU used for our experiment supports CUDA 2.1, supports up to 1024 threads per thread block. Once a thread block is assigned to a streaming multi-processor, it is further divided into 32-thread units called warps, and each warp is following same instruction.

For our simulation, we accelerated the first-order scattering simulation on NVIDIA GTX 560. This GPU, like all modern GPUs has off-chip memory and on-chip caching mechanisms. Off-chip memory includes global, texture and constant memory which incurs hundreds of cycles of memory latency. This device, unlike 1.x version, supports maximum three dimensional grid of thread blocks, has 32K of 32-bit registers per multiprocessor, and has 32 shared memory banks.

3.5. Combining Tomographic Image Reconstruction with Bi-directional Ray Tracing

Bi-directional ray tracing technique has been applied on the backscattering tomographic image reconstruction algorithm. In the case of CT, the light source is regarded as an X-ray source, and the camera is an X-ray detector.

Suppose photons travel from the X-ray source through ray. At each small portion of the object (a pixel in cross-sectional image), major number of photons travel straight towards the detector, while little portion of photons will scatter with reduced amount of energy. The scattered photons may reach to another detector bucket with lowered energy. From the view at the detector bin in terms of bi-directional ray tracing, each bin has limited view of detecting scattered photons.

First step of the algorithm is to compute a layer of photon-scattered cross-sectional image, assuming a cross-sectional image with no scattering is acquired first. As stated in section 3.2., the layer caches source to pixel attenuation. To achieve the attenuation, each thread on CUDA block accounts for each pixel at the cross-sectional image, and compute the intensity of the energy and traveling angle of photons shot from the X-ray source just before ray hits the corresponding pixel. Then, the threads multiply by the attenuation at its pixel itself and store.

Second step is to generate the region of interest (ROI) from detector side. The detector consist of a number of buckets, which collects X-ray photons that are not absorbed or scattered within their ROI. The system traces a ray from the detector. In reality each detector bin in CT machine has filter so that it reduces the amount of artifacts [16], thus their sight angle to catch the photon is limited. I define a 'fishnet', literally implies that a detector bin catches coming photon within its fishnet to simulate filtering property. Each pixel of the object has a phase function that simulates Compton scattering. The phase function steers photons with lowered energy. Since this energy shift depends on the angle of scattering and not on the nature of the scattering medium, it is reasonable to simulate that all pixel have common phase function.

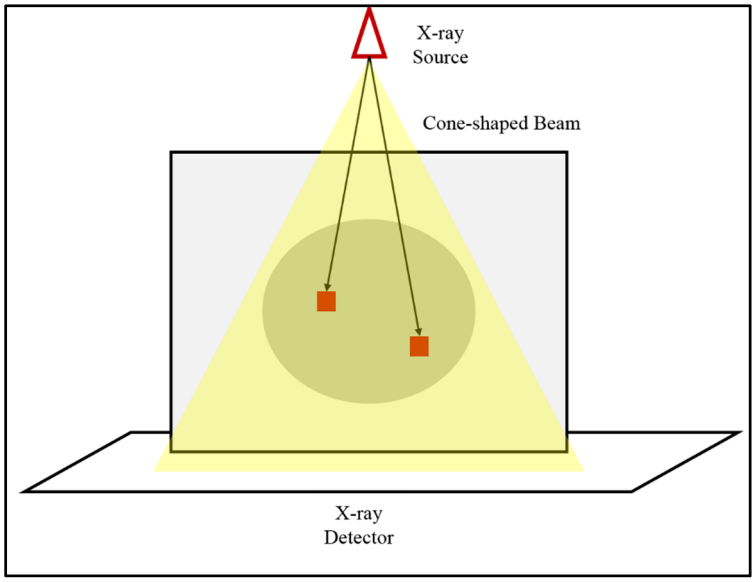

Figure 3.6. A 2-D Geometry of Formation of Caching Layer

Chapter 4

System Configuration

The major goal of the experiment is to get rid of the randomness, yet does not sacrifice the computing speed. As stated section 3.3, the deterministic method sacrifices computation cost, while Monte Carlo method sacrifices the accuracy as it has random characteristic.

For the experiment, the test machine has AMD Fx-8350 4.0GHz CPU with 16GB of RAM. For GPU side, the system utilized NVIDIA GTX560 with 1GB of graphics memory. The CPU has 8 cores, but the application only uses single core for the fair comparison with parallelism. This system configuration is reasonable setting to compare performance between CPU with high clock speed versus non-professional GPU.

In the experimental simulation, many properties has been simplified. For example, attenuation coefficient is proportional to its CT-number, which means the system the grayscale pixel value on pre-computed cross-sectional image. Furthermore, the fathom images is used instead of realistic image, to see the clear effect of the algorithm. The intensity of X-ray beam can be controlled by phase functions at pixels, and separate parameter for controlling X-ray photon intensity is not set. The phase function for the simple experiment is shown in figure 4.1.

The cosine function, although it is non-identical to real phase function, is used to see how well the experimental simulation code reveals the effect of scattering.

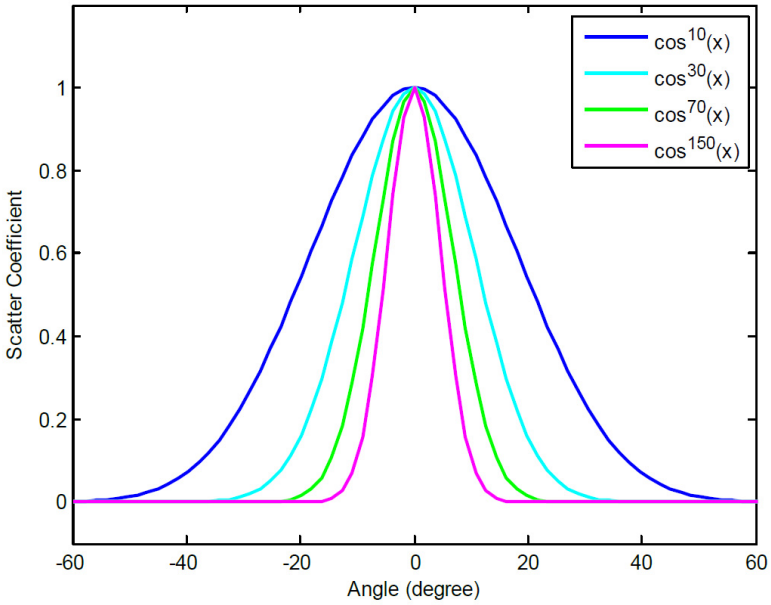

Figure 4.1. Simple Phase Functions for Experiments

Measurement metrics is the time comparison between CPU and GPU version of code, with experiment of various image size. Although a GPU has multiprocessor with a number of cores, global memory operations are bottleneck of GPU codes. On the experiment, performance of improved version of parallelism is also measured.

Chapter 5

System Implementation and Performance

5.1. Attenuation Caching Layer Generation

In first phase of the simulation the system builds Source-to-Pixel attenuation layer. Each 2-D GPU thread take the pixel, thus tracing of a ray from the source to each pixel runs concurrently on the GPU.

Algorithm 1. Generate Attenuation Caching Layer in Parallel

```
for all threads (thread_x, thread_y) do in parallel
    pixel_x ← thread_x, pixel_y ← thread_y
    Attenuation ←1.0
    for (ray_x, ray_y) = (source_x, source_y) → (pixel_x, pixel_y) do
        Attenuation ← Attenuation * Object(ray_x, ray_y) * μ(ray_x, ray_y)
    end
    AttenuationLayer(thread_x, thread_y) ← Attenuation
end
```

Figure 5.1. Pseudo code of Attenuation Caching Layer (Map) generation algorithm

The attenuation layer examples are shown in figure 5.2. Right hand side shows the attenuation layer of original cross sectional image on the left. The underlying assumption is that all pixels in the object contribute to scattering. Also, the experimental implementation has one point source at the top of the image, and it has enough distance to the top of the object, so that the X-ray source can cover all pixels of the object.

In the attenuation map layer from the right side of figure 5.2, white area indicates that the intensity of energy is high, while dark area has low energy intensity. In general CT system, the most bottom line of the attenuation layer is the energy intensity that the detector acquires.

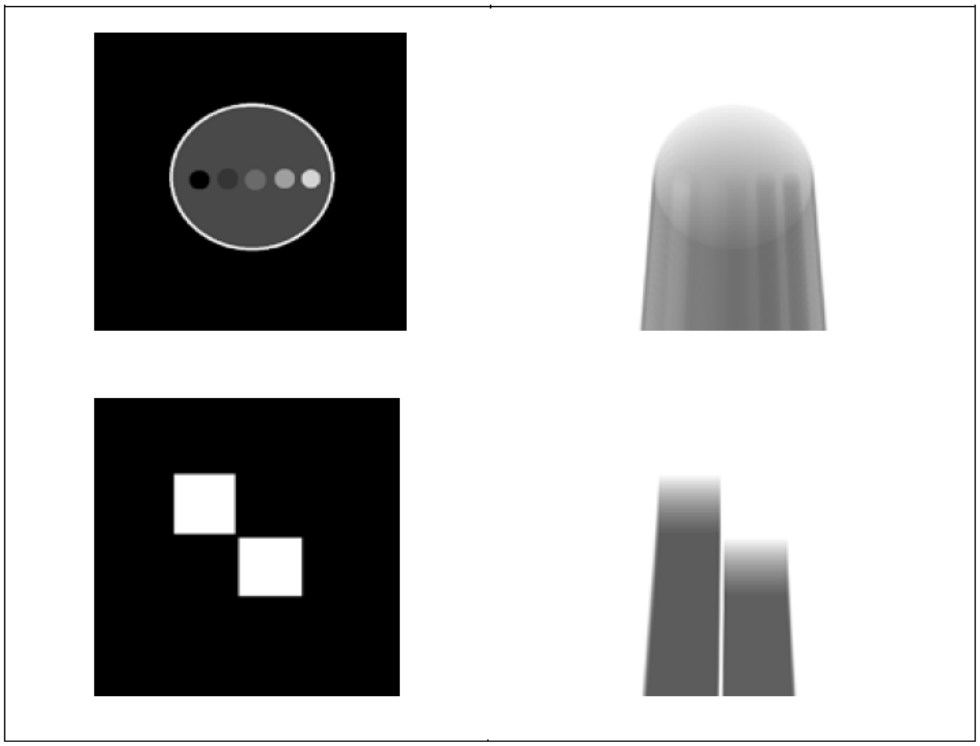

Figure 5.2. Examples of Attenuation Layer: Original Cross Section (Left), and Attenuation Layer (Right)

5.2. Scattering from all Pixels Computation

Computing scattering consists of three smaller steps. In the first step, a thread in the RoI takes one pixel and compute attenuation from the pixel to detector bin. In the next step, a thread combines two attenuation value at corresponding pixel position, which are source-to-pixel, and pixel-to-bin. Also, the phase function $\tau(\phi)$ is taken account. Finally, sum up all values in RoI position for all available detector bucket, Bin_j. Figure 5.3 shows pseudo code for this three step.

Algorithm 2. Computing Scattering from all Pixels in Parallel

```
for each bin Bin_j ← 0 to inf do
    for all threads (thread_x, thread_y) in RoI, do in parallel
        pixel_x ← thread_x + offset_x, pixel_y ← thread_y + offset_y
        Attenuation ←1.0
        for (Layer_x, Layer_y) = (pixel_x, pixel_y) → (bin_jx, bin_jy) do
            Attenuation ← Attenuation * Object(Layer_x, Layer_y) * μ(Layer_x, Layer_y)
        end
        Roi_j(pixel_x, pixel_y) ← Attenuation * AttenuationLayer(pixel_x, pixel_y)*τ(ϕ)
    end
    Bin_j = sum(Roi_j)
End
```

Figure 5.3. Pseudo code of computing scattering from all pixels

Figure 5.4 shows examples of how scattering effects on two square objects. From figure 5.3 (a) original image, three experiments, which are (b) non-scattering, (c) reasonable amount of scattering, and (d) extreme amount of

scattering. For the simulation, the amount of scattering is controlled by phase function in the object. For test case of (c) $\cos^{150}(x)$ is applied, and $\cos^{30}(x)$ is used for case (d).

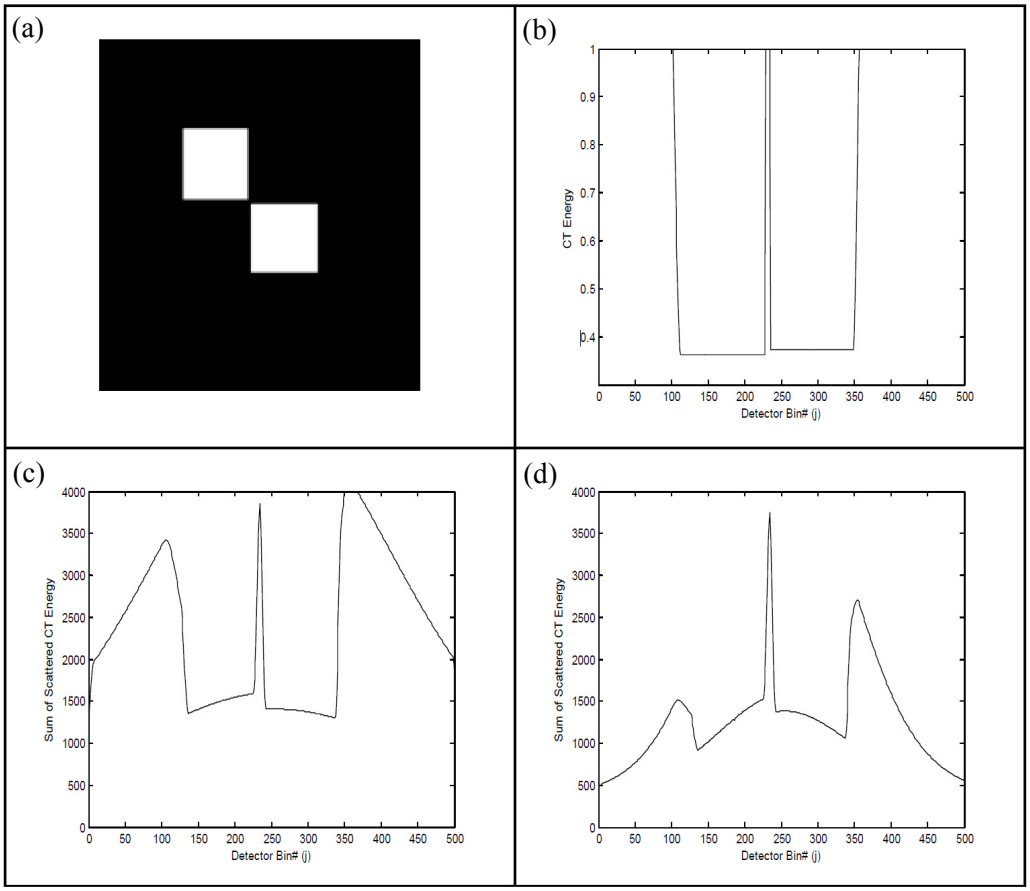

Figure 5.4. Example of scattering effect: (a) original cross-sectional image, intensity of the energy at the detector (b) with no scattering, (c) reasonable scattering, and (d) extreme scattering.

5.3. Simulation Performance

For the evaluation metric, I use computation time compared with CPU code. Since the performance of the GPU varies by number of cores, size of installed memory, and CUDA machine versions, it is hard to measure the performance of the algorithm itself within GPU experiments. Instead of using multiple threads of the GPU, CPU code use for-loop to iterate over all pixel positions. Figure 5.5 depicts the comparison result from five cross-section images with different size as input. Note that since the CPU alters its clock rate for power saving, the CPU time is averaged over 10 top results (when the clock rate reached maximum) out of 30 test cases. The performance of GPU dominants the CPU.

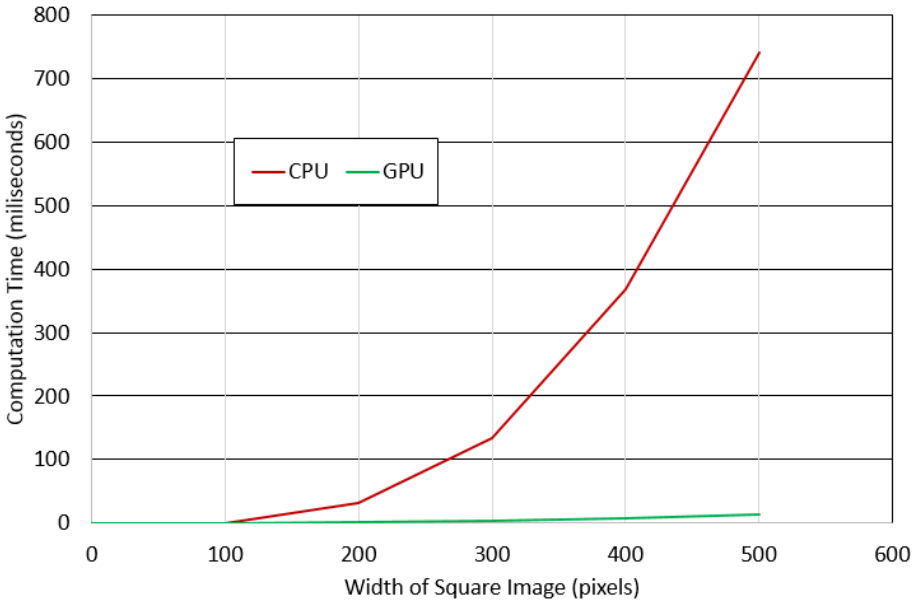

Figure 5.5. Result of experiment: Computation time of CPU and GPU, with input image size 100x100 pixels to 500x500 pixels

5.4. Improvement of the Algorithm

Although the performance of GPU dominates CPU, there is still room for improvement. The pseudo code of computing scattering algorithm in figure 5.3 has two for-loops within a loop. The loop is major drag of CPU computing, but it can be improved by parallelism, if outer- and inner-loop has no dependent relationship. Since the most outer loop goes towards each detector bins, it can be parallelized. The major issue, however, is memory consumption. It requires more memory to be allocated at the same time, which hinders GPU computing performance. Figure 5.6 shows performance improvement from iterative method on 2-D grid to 3-D grid.

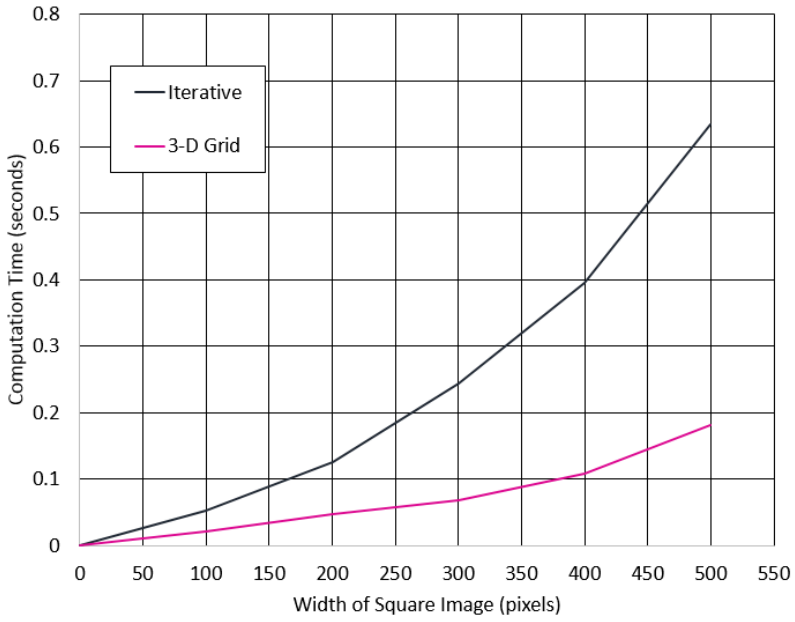

Figure 5.6. Performance comparison: 3-D grid of threads versus 2-D grids with iterative

As shown in figure 5.7, more active blocks of threads use more memory. It measures minimum memory required to process the algorithm. From the reason that memory operation is a bottleneck of GPU computation, the use of 3-D thread grids have less efficient, i.e., computational time linearly increases as the size of the image gets larger.

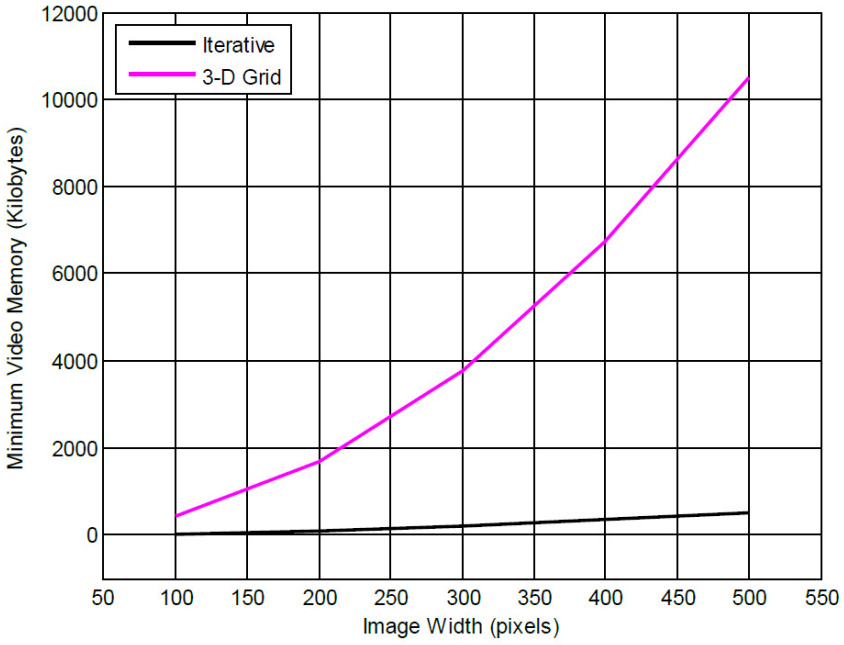

Figure 5.7. Video memory usage comparison: 3-D grid of threads versus 2-D grids with iterative method

Chapter 6

Conclusion and Future Work

In this study, we modeled an algorithm and implemented code for simulating fast, but considers all possible first-order scattering. To overcome the trade-offs between speed and accuracy, the developed algorithm exploited bi-directional ray tracing method, which enables simulating one or more order scattering. The suggested GPU algorithm showed dominant performance over CPU in terms of speed.

The experiment shows that using more thread block speeds up the overall computation. However, since the GPU memory operation is a bottleneck, it may result in small improvements depending on the GPU performance. Thus, use of more thread blocks needs careful consideration, and at the same time the optimization for less memory use is needed.

The evaluation metrics measures the performance of the algorithm itself. Further research and implementation will focus more on i) finding a correct phase function given a material and X-ray source intensity, and ii) reconstruction performance with scatter effects. This will lead far more sophisticated simulation and possibly applied to the general applications.

Bibliography

[1] G. Rigaud, M. K. Nguyen, and A. K. Louis, "Modeling and simulation results on a new Compton scattering tomography modality," *Simul. Model. Pract. Theory*, vol. 33, pp. 28–44, Apr. 2013.

[2] D. Lazos, Z. Kolitsi, and N. Pallikarakis, "A Software Data Generator for Radiographic Imaging Investigations," vol. 4, no. 1, pp. 76–79, 2000.

[3] P. Duvauchelle, N. Freud, V. Kaftandjian, and D. Babot, "A computer code to simulate X-ray imaging techniques," *Nucl. Instruments Methods Phys. Res. Sect. B Beam Interact. with Mater. Atoms*, vol. 170, no. 1–2, pp. 245–258, Sep. 2000.

[4] N. Freud, J.-M. Létang, and D. Babot, "A hybrid approach to simulate multiple photon scattering in X-ray imaging," *Nucl. Instruments Methods Phys. Res. Sect. B Beam Interact. with Mater. Atoms*, vol. 227, no. 4, pp. 551–558, Jan. 2005.

[5] N. Freud, P. Duvauchelle, S. a. Pistrui-Maximean, J.-M. Létang, and D. Babot, "Deterministic simulation of first-order scattering in virtual X-ray imaging," *Nucl. Instruments Methods Phys. Res. Sect. B Beam Interact. with Mater. Atoms*, vol. 222, no. 1–2, pp. 285–300, Jul. 2004.

[6] J. Giersch, A. Weidemann, and G. Anton, "ROSI—an object-oriented and parallel-computing Monte Carlo simulation for X-ray imaging," *Nucl. Instruments Methods Phys. Res. Sect. A Accel. Spectrometers, Detect. Assoc. Equip.*, vol. 509, no. 1–3, pp. 151–156, Aug. 2003.

[7] F. Inanc, B. Vasiliu, and D. Turner, "Parallel Implementation of the Integral Transport Equation-Based Radiography Simulation Code," *Nucl. Sci. Eng.*, vol. 137, no. 2, pp. 173–182, 2001.

[8] E. Lafortune and Y. Willems, "Bi-directional path tracing," *Proc. CompuGraphics*, vol. 93, pp. 145–153, 1993.

[9] J. Da Silva, "A generalized model for the conversion from ct numbers to linear attenuation coefficients," *IEEE Trans. Nucl. Sci.*, vol. 50, no. 5, pp. 1510–1515, Oct. 2003.

[10] A. Averbuch, I. Sedelnikov, and Y. Shkolnisky, “CT reconstruction from parallel and fan-beam projections by a 2-D discrete Radon transform.,” *IEEE Trans. Image Process.*, vol. 21, no. 2, pp. 733–41, Feb. 2012.

[11] X.-M. Hua, “Monte Carlo simulation of Comptonization in inhomogeneous media,” *Comput. Phys.*, vol. 11, no. 6, p. 660, 1997.

[12] S. B., I. J.C., M. R., and P. B., “Bidirectional Instant Radiosity,” in *Proceedings of the 17th Eurographics conference on Rendering Techniques*, 2006, pp. 389–397.

[13] H. Jensen and N. Christensen, “Photon maps in bidirectional Monte Carlo ray tracing of complex objects,” *Comput. Graph.*, vol. 19, no. 2, pp. 215–224, 1995.

[14] E. Veach, “Robust Monte Carlo methods for light transport simulation,” Stanford University, 1997.

[15] D. B. Kirk and W. Hwu, *Programming Massively Parallel Processors : A Hands-on Approach*, 1st ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2010.

[16] L. W. Goldman, “Principles of CT and CT technology.,” *J. Nucl. Med. Technol.*, vol. 35, no. 3, pp. 115–28; quiz 129–30, Sep. 2007.