RTE Solver Manual

Matthew A. Kupinski

Version 4.0
Aug, 2011

Abstract

The RTE Solver suite of programs computes solutions to the radiative transport equation (RTE) which fully describes the particle-like interactions of photons propagating through scattering media. In this document, we will talk about the GPU version of the software that we have developed to accomplish the task of simulating photon propagation in non-uniform tissues. The RTE Solver programs can model the complicated interactions that take place when probing human tissue with near-infrared sources. This software package can be used with any range of absorption and scattering coefficients and makes no simplifying assumptions other than the implicit assumption that wave phenomena are ignored. The RTE Solver programs can be used to model the propagation of photons through tissue using any type of source (laser, fiber, etc.) and with any detector setup (transmission, reflection, point detectors, etc). In addition, the RTE Solver can handle steady-state simulations or intensity-modulated sources. This document will both serve as a users guide to the software and a technical document describing the methods used.

TECHNOLOGY KEYWORDS: Light propagation; Neumann series; Equation of radiative transfer; Spherical harmonics, GPU.

APPLICATION KEYWORDS: Diffuse optical tomography

Contents

1 Introduction **5**
1.1 About the RTE Solver . . . 5
1.2 Revision history . . . 7

2 Using the Software **8**
2.1 Overview . . . 8
2.2 Installation . . . 8
2.3 The source code . . . 9
2.4 The parameter file . . . 9
2.5 Output files . . . 11

3 Examples **13**
3.1 Example 1: Uniform RTE . . . 13
3.2 Example 2: Increasing the absorption coefficient . . . 15
3.3 Example 3: Nonuniform phantom . . . 16

4 Technical details **19**
4.1 Radiometric Quantities . . . 19
4.1.1 The distribution function . . . 19
4.1.2 The source function . . . 19
4.1.3 Energy dependence . . . 19
4.2 The Radiative Transport Equation . . . 20
4.2.1 Absorption . . . 20
4.2.2 Emission . . . 20
4.2.3 Propagation . . . 20
4.2.4 Scattering . . . 21
4.2.5 Time dependence . . . 21
4.2.6 The Neumann series . . . 22
4.3 Spherical Harmonics . . . 22
4.3.1 The scattering operator $\boldsymbol{\mathcal{K}}$ . . . 23
4.3.2 The propagation operator $\boldsymbol{\mathcal{X}}$ . . . 23
4.4 Voxelization . . . 23

5 GPU Implementation and Numerical Algorithms **25**
5.1 GPU Implementation details . . . 25
5.2 `GenerateSourceBeam` . . . 26

5.3 ComputeTransImage 26
5.4 Other routines 27
5.4.1 Neuman 27
5.4.2 prop_abs 27
5.4.3 prop_scat 27
5.4.4 CombineOuts 27
5.4.5 compute_prop_abs and compute_diagonal_abs 27

Bibliography **27**

Chapter 1: Introduction

1.1 About the RTE Solver

When probing human tissue with near-infrared (NIR) light, it is well known scattering and absorption are the primary interactions taking place. The goal in diffuse optical tomography (DOT) is to probe the body with NIR light and to determine the absorption and scattering properties of the tissue at various locations. These absorption and scattering properties can be used by physicians to determine the absence or presence of various diseases. For example, the absorption varies greatly between water, deoxygenated hemoglobin, and oxygenated hemoglobin (See Figure 1.1). Thus, DOT can be used in breast-imaging applications to look for areas of increased vascularization which is indicative of cancer. For DOT imaging to be successful, we need a method of propagating the NIR photons through the tissue and predicting the detector output – the forward problem. We also need a method to find the map of absorption and scattering coefficients that is consistent with the measured data – the inverse problem. The RTE Solver is a very general and powerful method for solving the forward problem.

The RTE Solver takes as input information about the source, the distribution of absorption and scattering coefficients, and outputs the transmitted, reflected, and phase images. The RTE Solver software package can handle any type of source including laser sources of varying diameters and fiber sources. Uniform media, single lesion, and general non-uniform media can all be handled by the software. Other approaches to solving the forward problem include the use of the diffusion equation (DE) approximation to the RTE, and Monte Carlo (MC) methods. Table 1.1 compares standard MC and DE methods to our RTE Solver software. As this table indicates, the RTE Solver package can accomplish what DE and MC can and has many advantages over these other approaches.

The RTE Solver programs can be used to model almost any source/detector geometry and is thus an ideal tool for analyzing different configurations of DOT systems. Unlike other approaches which require contact detector and small fiber sources, the RTE Solver can simulate the more advanced geometries being devised by many different research groups. The first uses of the RTE Solver software will be to determine an ideal system configuration for DOT imaging of the breast and we will also use this software package in the implementation of the inverse problem.

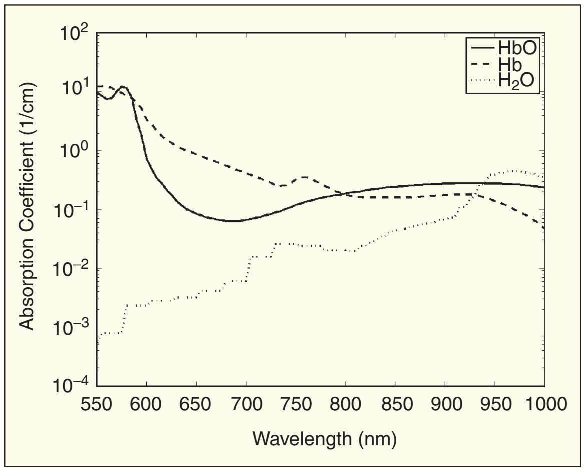

Figure 1.1: The absorption coefficient of water, oxygenated blood, and deoxygenated blood. At NIR wavelengths, there is a large variation in the absorption coefficients. Thus, NIR DOT is ideally suited to distinguish between these types of regions in the body. Figure courtesy of D. A. Boas.

Table 1.1: A comparison of the advantages of the RTE Solver over other techniques.

Feature	RTE	DE	MC
Can handle non-uniform phantoms	X	X	X
Can handle modulated sources	X	X	X
Can handle directed sources such as laser beams	X		X
Works for a wide range of absorption and scattering coefficients	X		X
Can accurately handle anistotropy	X		X
Can handle thin slabs of material	X		X
Output position and direction information of the exiting photons	X		X
Produces solutions that are free of noise and numerical instability	X		
Produces results quickly	X	X	
Requires little data storage to keep the output	X	X	

1.2 Revision history

Date	Version	Changes	Authors
January 1, 2009	1.0	Initial version. Worked well for models with small scattering coefficients. Failed to converge for more complicated models.	M. A. Kupinski
April 1, 2009	1.1	Modified the method used to compute the propagation kernel values for the diagonal elements. This change resulted in an algorithm that always converged. However, the converged answer was inaccurate for phantom with large scattering coefficients	M. A. Kupinski
July 1, 2009	1.2	Modified the method used to compute the off-diagonal elements of the propagation kernel. Modified the way in which the initial Neumann series term was computed. Changed the way in which the propagator was computed to speed up the algorithm for models with large scattering coefficients. Sped up the way in which the transmitted, reflected, and phase images were computed.	M. A. Kupinski E. Clarkson D. Kang
May 30, 2010	2.0	Changed the method in which the diagonal components of the propagation operator are computed. Added a new parameter, selfSubVox, to handle this new computation. Got rid of the correctionFactor parameters as it is no longer necessary. Got rid of support for Intel Compiler since GNU compiler now has complete openmp functionality. Modified examples based on new software. Renamed initial software package RTE_uniform to separate this software from other packages. Now include 2 other programs for small lesion RTE and non-uniform RTE.	M. A. Kupinski A. K. Jha D. Kang
January 4, 2011	3.0	Complete rewrite to execute quickly on GPU computers using NVidia's CUDA libraries. Now just a single version that handles non-uniform phantoms.	A. K. Jha M. A. Kupinski
Aug 8, 2011	3.0	Fix various issues with the RTE solver code and improve the GPU implementation.	A. K. Jha M. A. Kupinski

Chapter 2: Using the Software

2.1 Overview

The RTE Solver software is written in standard C and CUDA and consists of one primary program called `rte_gpu`. Once the distribution file is unarchived, the directory structure is as follows:

Makefile The file used by the `make` command to compile the RTE Solver package. For use with the CUDA `nvcc` compiler.

src/ The source files for the non-uniform, GPU-based RTE solver.

doc/ The directory containing this documentation file.

examples/ Example parameter files are kept in this directory. Some of the example runs are described in the next chapter.

utility/ Directory containing Matlab utility functions for displaying the output of the RTE Solver as well as other helpful routines.

bin/ Directory containing the compiled program `rte_gpu`.

Once compiled, the RTE Solver programs take a single argument as input which corresponds to the parameter file for that run. This parameters file is discussed in more detail in a subsequent section but contains all the necessary input and output specifications for that run of the RTE Solver. Thus, assuming that the program `rte_gpu` is in the current binary path, one can execute `> rte_gpu paramfile.par` to run the program.

2.2 Installation

This program should compile easily on any UNIX or UNIX-based computers include Mac OS computers, which have a CUDA enabled GPU. We have currently been testing this software using a Linux machine with Tesla C2050 GPU cards . CUDA SDK and CUDA toolkit of version greater than 3.0 should be installed on the system. The Linux operating system we have used successfully is the Ubuntu version 10.10, with CUDA toolkit 3.2.

The RTE Solver programs take up approximately 30MB of CPU RAM when running (this varies depending on how the parameters are set) and requires a processor that supports 64-bit computations. Modern Intel processors are all 64-bit. The programs are completely self-contained and do

Table 2.1: The source files for the `RTE_uniform` software package.

Filename	Description
`RTE_uniform.cu`	The main source file that solves the uniform RTE.
`kernels.cu`	All the kernels in the code
`inits.cu`	Code to initialize the many structures for organizing the geometry, the source, and the phantom.
`dist.cu`	Code to generate and manipulate distribution functions.
`sph_harm.cu`	Code for generating Legendre polynomials and spherical harmonic functions.
`rte.h`	Definitions of structures, and function prototypes.
`iniparser.cu`	Freely available source code for reading in the parameter file.
`dictionary.cu`	Freely available source code for storing the parameters read in from the file.
`iniparser.h` `dictionary.h`	Relevant header files for the parameter parser.

not require pre-existing libraries, except for the libraries that are provided by the CUDA software package. A third-party set of function for reading in parameter files is included as part of the RTE Solver package [4]. These source files are not used for any of the scientific computations; they are distributed freely and can be used without restrictions.

2.3 The source code

The code (in the `src/` consists of source files and header files that are all compiled together via the `Makefile`. The source files for all three software packages are listed in Table **??**. The files have been given the similar names in the CPU and GPU versions of the code for consistency. The `cu` extension is for CUDA C programs, while the `c` extension is for the C programs.

A use of the OpenMP library is made for introducing parallelism at the thread level.

2.4 The parameter file

The usage of all three programs is simply:

```
> rte_cpu parameterfile.par
```

where `parameterfile.par` is a text parameter file. This parameter file has five sections which must be specified in square brackets. The `[Runtime]` section of the parameter file (see Table 2.2) essentially names the output files for the transmitted, reflected, and transmitted phase images. The RTE Solver software package does not warn the user if an output file already exists so the user must take care in ensuring that each run of the program has a unique set of files outputs. The section `[Algorithm]` (Table 2.3 controls the basic operation of the RTE Solver algorithm.

Table 2.2: The `[Runtime]` parameters to control the RTE Solver software package.

Parameter	Description
`TransFile`	The name of the file to contain the transmitted image file.
`ReflectFile`	The name of the file to contain the reflected image file.
`PhaseFile`	The name of the file to contain the phase image file.

Table 2.3: The `[Algorithm]` parameters to control the RTE Solver software package.

Parameter	Description
`nTerms`	The number of terms in the Neumann series.
`nL`	The number of spherical harmonics to use. The actual number of terms is (nL+1)^2 since m always goes from $-l$ to l.

In this section the number of terms in the Neumann series and the number of spherical harmonic terms are specified. The section `[Geometry]` (Table 2.4) is where the geometry of the phantom is described. Also in this section, the parameters that control the sub-voxelization are described. Finally, the sections `[Phantom]` (Table 2.5) and `[Source]` (Table 2.6) control the parameters of the phantom and the source, respectively. Example parameter files are shown and discussed in the next chapter. There are differences in the parameter files for the three separate programs; these differences are noted in the tables.

Most of the parameters are easy to understand and need little explanation. However, a few parameters in the `[Geometry]` section require some explanation. The propagation kernel (described in detail in the Chapter 4) is at the heart of the RTE Solver program. This propagation kernel describes the movement of photons from one place to another and includes loss of photons from scatter and absorption. The accurate computation of the propagation kernel is required to obtain accurate output images.

Table 2.4: The `[Geometry]` parameters to control the RTE Solver software package.

Parameter	Description
`nX`	The number of samples in x.
`nY`	The number of samples in y.
`nZ`	The number of samples in z.
`xMin`	The starting x sample in [cm].
`yMin`	The starting y sample in [cm].
`zMin`	The starting z sample in [cm].
`xMax`	The ending x sample in [cm].
`yMax`	The ending y sample in [cm].
`zMax`	The ending z sample in [cm].

Table 2.5: The `[Phantom]` parameters to control the RTE Solver software package.

Parameter	Description
`NoTiss`	Number of tissue types in media.
`TissMapFile`	The filename that contains the tissue map.
`TissAbs`i	The absorption coefficient for the ith tissue type [1/cm]
`TissSc`i	The scatter coefficient for the ith tissue type [1/cm]
`g`	The anisotropy factor (between 0 and 1)
`n`	The index of refraction of the material
`c`	The speed of light in [cm/s].

Table 2.6: The `[Source]` parameters to control the RTE Solver software package.

Parameter	Description
`posX`	The x position of the source in [cm].
`posY`	The y position of the source in [cm].
`posZ`	The z position of the source in [cm].
`diam`	The diameter of the source in [cm].
`theta`	The θ angle of the source in [radians].
`phi`	The ϕ angle of the source in [radians].
`magnitude`	The strength of the source.
`modulation`	The frequency of the intensity modulations [1/s].

In `[Phantom]`, the user defines the total number of tissue types `NoTiss`. The absorption and scattering coefficients of each tissue type is determined by `TissAbs`i and `TissSc`i. Note that the numbering of tissues begins from 1, and the 0^{th} tissue type is reserved for air. Finally, the tissue map is contained in a binary file `TissMapFile` which is an unsigned short binary file of dimension `nX` by `nY` by `nZ`. We have also added a utility matlab script (create_phantom.m) to generate the tissue map file.

2.5 Output files

The RTE Solver outputs three images corresponding the reflection, transmission, and phase images. These image are binary files with a format summarized in Table 2.7. We have been using Matlab to display the images. The following Matlab function (included in the distribution file) can be used to read and display any of the image file outputs from the RTE Solver.

```
function [img] = LoadImage(nme)
% Matlab function to read in an output image from the RTE Solver
% img = LoadImage(nme)
%
% img is the output image in matrix form

fid = fopen(nme,'r');
% n is the x and y dimensions of the image
```

Table 2.7: The format of the output image files.

Description	Length
xDim	a single 16-bit unsigned integer
yDim	a single 16-bit unsigned integer
img	an array of doubles (32-bit) xDim $\times$ yDim in size.
	The ordering of img is rows first and then columns.

```
n = fread(fid,[1 2],'int');
img = fread(fid,prod(n),'double');
img = reshape(img,n);
fclose(fid);
```

Once `img` is loaded into matlab, a simple `imagesc(img)` will display the image on the screen.

The transmitted and reflected images are in units of numbers of photons per second incident on each detector element. If the `magnitude` parameter is set to 1, then the output can also be considered the probability that a photon hits a particular detector element. The phase image has units of radians.

Chapter 3: Examples

To illustrate the use of the RTE Solver, we will simulate a uniform phantom with a low scattering coefficient (Example 1), a mid-range scattering coefficient with a signal (Example 2), and a low scattering, non-uniform case (Example 3). A laser source will be incident near the middle of the phantom and normal to the surface.

3.1 Example 1: Uniform RTE

The following parameter file is used to simulate a tissue sample has an absorption coefficient of 0.01 cm^{-1} and a scattering coefficient of 1.0 cm^{-1}. The Neumann series is taken out to 100 terms and the total number of spherical harmonics is 4 (i.e., $(\mathrm{nL}+1)^2$).

```
[Runtime]
TransFile = Exp1Trans.out
ReflectFile = Exp1Ref.out
PhaseFile = Exp1Phase.out

[Algorithm]
nTerms = 20
nL = 0

[Geometry]
nX = 20
nY = 20
nZ = 20
xMin = -1
xMax = 1
yMin = -1
yMax = 1
zMin = 0
zMax = 2
subVox = 0
selfsubVox = 10
subThresh = 0.0
```

```
propThresh = 0.6

[Phantom]
NoTiss = 1
TissMapFile = Exp1.bin
TissAbs1 = 0.00
TissSc1 = 1.0
g = 0
n = 1
c = 2.9979e10

[Source]
posX = -.1
posY = -.1
posZ = 0
diam = 0.1
theta = 0.0
phi = 0.0
magnitude = 1
modulation = 0
```

The .bin file is a binary file that contains 1's and 2's for the different tissue types (20x20x20 array). The RTE Solver program is run on this program by typing

```
> rte_gpu Example1.par
```

The program then outputs the following text to the screen while running:

```
Loading in geometry and phantom information...
Computing the time-independent solution...
Creating the source...
Generating terms for propagator...
Neuman series for 50 terms...
   100 percent complete
     ...
```

Apart from this, some other diagnostic messages are displayed.

The output from this example run is three files specified in the `[Runtime]` section of the parameter file. Using the `load_results_gpu` Matlab function we can process these files using the following Matlab commands:

```
tr = load_results_gpu('Exam1Trans.out');

figure(1);
semilogy(tr(10,:));
```

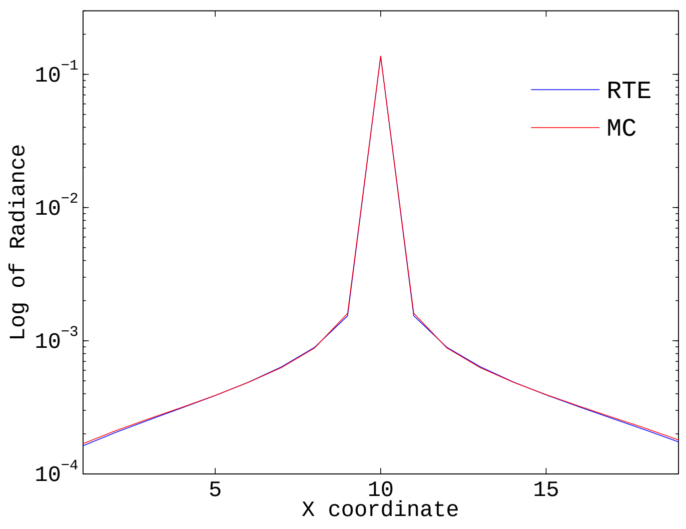

Figure 3.1: The profile of the center of the image for the RTE and the MC outputs. The images are displayed on a log scale to better display the data.

The results are shown in Figure 3.1. The output is also compared with the MC result in the same figure.

3.2 Example 2: Increasing the absorption coefficient

In this example, the scattering coefficient is 1 cm^{-1} and the absorption coefficient is set to 0.1 cm^{-1}. In this case, it is known that the DE does not given the correct response. The parameter file `Example2.par` is:

```
[Runtime]
TransFile = Exam2Trans.out
ReflectFile = Exam2Ref.out
PhaseFile = Exam2Phase.out

[Algorithm]
nTerms = 50
nL = 3

[Geometry]
nX = 50
nY = 50
nZ = 50
xMin = -0.5
xMax = 0.5
yMin = -0.5
```

```
yMax = 0.5
zMin = 0
zMax = 1.0
subVox = 0
selfsubVox = 1
subThresh = 0.0
propThresh = 10.0

[Phantom]
mus = 5.0
mua = 0.1
g = 0.0
n = 1.0
c = 2.9979e10

[Source]
posX = -0.01
posY = -0.01
posZ = 0
diam = 0.02
theta = 0.0
phi = 0.0
magnitude = 1
modulation = 0
```

The output images for this run are shown in Figure **??**.

3.3 Example 3: Nonuniform phantom

In the final example, we consider a non-uniform phantom. This phantom consists of two different regions each with different absorption coefficients, and replicates a tissue with a circular lesions. The scattering coefficients are the same for both the tissues and equal to 1 cm^{-1}. The absorptin coefficients are equal to 0.01 cm^{-1} for the background tissue and 0.1 cm^{-1} for the lesion tissue.

```
[Runtime]
TransFile = Exam4Trans.out
ReflectFile = Exam4Ref.out
PhaseFile = Exam4Phase.out

[Algorithm]
nTerms = 1
nL = 0
```

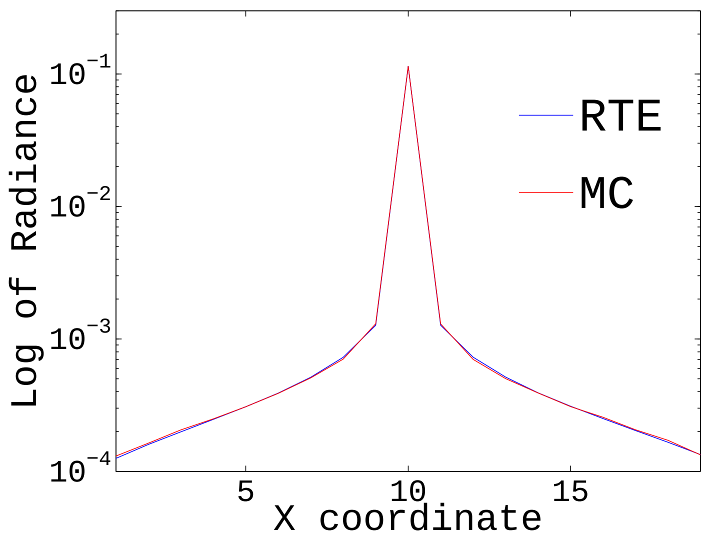

Figure 3.2: The profile of the center of the image for the RTE and the MC outputs. The images are displayed on a log scale to better display the data.

```
[Geometry]
nX = 20
nY = 20
nZ = 20
xMin = -2.0
xMax = 2.0
yMin = -2.0
yMax = 2.0
zMin = 0
zMax = 4
subVox = 5
subThresh = 0.3
propThresh = 0.5
selfsubVox = 5

[Phantom]
NoTiss = 1
TissMapFile = Exp4.bin
TissAbs1 = 0.01
TissSc1 = 1.0
TissAbs2 = 0.02
TissSc2 = 1.5
TissAbs3 = 0.03
TissSc3 = 2.0
g = .5
n = 1.33
```

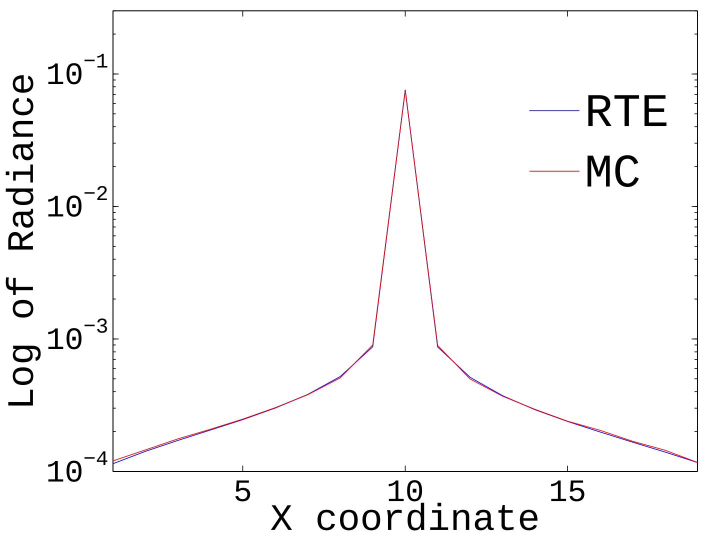

Figure 3.3: The profile of the center of the image for the RTE and the MC results. The images are displayed on a log scale to better display the data.

```
c = 2.9979e10

[Source]
posX = -0.1
posY = -0.1
posZ = 0.0
diam = 0.1
theta = 0.0
phi = 0.0
magnitude = 1
modulation = 0
```

The output images for this run are shown in Figure 3.3.

Chapter 4: Technical details

The key reference for our approach to modeling photon propagation can be found in Chapter 10 of Barrett and Myers [1]. Many of the mathematical descriptions below are taken from this reference.

4.1 Radiometric Quantities

The radiative transport equation or RTE describes the spatio-temporal propagation of photons through a media. The RTE models light as if it is made up of localized photons. Interference, diffraction, and other wave-properties of light are not taken into account with the RTE.

4.1.1 The distribution function The distribution function $w(\boldsymbol{r}, \hat{\boldsymbol{s}}, \mathcal{E}, t)$ is the fundamental quantity that we are attempting to describe using the RTE. In terms of photons $w(\boldsymbol{r}, \hat{\boldsymbol{s}}, \mathcal{E}, t)\Delta V \Delta\Omega\Delta\mathcal{E}$ can be interpreted as the number of photons contained in volume ΔV centered on $\boldsymbol{r}$, traveling in solid angle $\Delta\Omega$ about direction $\hat{\boldsymbol{s}}$, and having energies between $\mathcal{E}$ and $\mathcal{E} + \Delta\mathcal{E}$ at time t. It is clear that this distribution function is a complete characterization of the position, direction, and energy of the photons in media as a function of time.

4.1.2 The source function While $w(\boldsymbol{r}, \hat{\boldsymbol{s}}, \mathcal{E}, t)$ describes the distribution of photons in the media, the source function $\Xi(\boldsymbol{r}, \hat{\boldsymbol{s}}, \mathcal{E}, t)$ describes the injection of photons into the media. Much like the distribution function, $\Xi(\boldsymbol{r}, \hat{\boldsymbol{s}}, \mathcal{E}, t)\Delta V \Delta\Omega\Delta\mathcal{E}$ can be interpreted as the number of photons injected per second from volume ΔV in energy range $\Delta\mathcal{E}$, over solid angle $\Delta\Omega$, and at time t. The principal difference between $w(\boldsymbol{r}, \hat{\boldsymbol{s}}, \mathcal{E}, t)$ and $\Xi(\boldsymbol{r}, \hat{\boldsymbol{s}}, \mathcal{E}, t)$ is that $\Xi(\cdot)$ is a rate while $w(\cdot)$ is not.

4.1.3 Energy dependence For this work, we are assuming a monoenergetic laser source. Thus, the source function can be written as $\Xi(\boldsymbol{r}, \hat{\boldsymbol{s}}, t)$ without a dependence on energy. In addition, we will consider only elastic scattering (more on this later) within the media. This assumption implies that a scattered photon does not lose any energy. Thus, we can write the distribution function $w(\boldsymbol{r}, \hat{\boldsymbol{s}}, t)$ as independent of energy as well.

4.2 The Radiative Transport Equation

The RTE is given, very generally, as

$$\frac{dw}{dt} = \left[\frac{\partial w}{\partial t}\right]_{abs} + \left[\frac{\partial w}{\partial t}\right]_{em} + \left[\frac{\partial w}{\partial t}\right]_{prop} + \left[\frac{\partial w}{\partial t}\right]_{sc}. \tag{4.1}$$

The time derivative of the distribution function has contributions from absorption, emission (i.e., the source), propagation, and scatter, respectively. The next sections describe each of these four terms.

4.2.1 Absorption We can describe the change of the distribution function due to absorption as,

$$\left[\frac{\partial w}{\partial t}\right]_{abs} = -c_m \mu_{abs}(\boldsymbol{r}) w(\boldsymbol{r}, \hat{\boldsymbol{s}}, t), \tag{4.2}$$

where c_m is the speed of light in the media and $\mu_{abs}(\boldsymbol{r})$ is the absorption coefficient of the media as a function of position. We assume that the media is not changing during imaging and, thus, $\mu_{abs}(\boldsymbol{r})$ is not a function of time. In addition, we assume that the speed of light is not changing within the media. In actuality, we can account for slowly varying changes in the speed of light within the media. However, abrupt changes in the speed of light would cause boundary reflections which cannot be accounted for by the RTE. For example, non-contact sources will naturally have reflection occur at the surface of the probing object. These reflections are not taken into account by the RTE.

4.2.2 Emission The emission of light is fully described by the source function which was already defined as a rate of photons. Thus,

$$\left[\frac{\partial w}{\partial t}\right]_{em} = \Xi(\boldsymbol{r}, \hat{\boldsymbol{s}}, t). \tag{4.3}$$

If we were imaging with injected fluorescent dyes, then the source term would be more complicated and would depend on the distribution of the fluorescent dyes. For our purposes, the source term is the laser input.

4.2.3 Propagation The propagation term describes the movement of photons from one location to another. If we consider a small time interval Δt, then a photon will travel in a straight line of direction $\hat{\boldsymbol{s}}$. This propagation is described by,

$$w(\boldsymbol{r}, \hat{\boldsymbol{s}}, t) = w(\boldsymbol{r} + c_m \Delta t \hat{\boldsymbol{s}}, \hat{\boldsymbol{s}}, t + \Delta t). \tag{4.4}$$

Using a Taylor series expansion and ignoring higher order terms (see [1]), we find that,

$$\left[\frac{\partial w}{\partial t}\right]_{prop} \approx -c_m \hat{\boldsymbol{s}} \cdot \nabla w(\boldsymbol{r}, \hat{\boldsymbol{s}}, t). \tag{4.5}$$

4.2.4 Scattering To describe scattering, we must consider the scattering of photons *out* of a location and the scattering of photon *in* from another location. This process is described by,

$$\left[\frac{\partial w}{\partial t}\right]_{sc} = -c_m \mu_{sc}(\boldsymbol{r}) w(\boldsymbol{r}, \hat{\boldsymbol{s}}, t) + \int_{4\pi} d\Omega' \, K(\hat{\boldsymbol{s}}, \hat{\boldsymbol{s}}' | \boldsymbol{r}) w(\boldsymbol{r}, \hat{\boldsymbol{s}}', t). \tag{4.6}$$

The first term in the above equation describes the photons scattering out, hence the minus sign. Much like the absorption term, this process is described by a scattering coefficient $\mu_{sc}(\boldsymbol{r})$. The second, more complicated, term described the in-scattering of photons. Here, the kernel $K(\cdot)$ describes the probability that a photon traveling in one direction $\hat{\boldsymbol{s}}'$ will scatter to a new direction $\hat{\boldsymbol{s}}$. Because the scattering probability depends on position, the scattering kernel must also depend on position. This term is usually assumed to be a function of the inner product of these two unit vectors $\hat{\boldsymbol{s}} \cdot \hat{\boldsymbol{s}}'$ which we will exploit later. We will write this second term in operator notation as $\boldsymbol{\mathcal{K}} w(\boldsymbol{r}, \hat{\boldsymbol{s}}, t)$.

Finally, the RTE can be written as,

$$\frac{dw}{dt} = -c_m \mu_{tot}(\boldsymbol{r}) w(\boldsymbol{r}, \hat{\boldsymbol{s}}, t) + \Xi(\boldsymbol{r}, \hat{\boldsymbol{s}}, t) + \boldsymbol{\mathcal{K}} w(\boldsymbol{r}, \hat{\boldsymbol{s}}, t) - c_m \hat{\boldsymbol{s}} \cdot \nabla w(\boldsymbol{r}, \hat{\boldsymbol{s}}, t), \tag{4.7}$$

where $\mu_{tot} = \mu_{abs} + \mu_{sc}$.

4.2.5 Time dependence We will assume an intensity-modulated source. We model this modulation as a shifted cosine function on the source distribution. That is,

$$\Xi(\boldsymbol{r}, \hat{\boldsymbol{s}}, t) = \frac{1}{2} \Xi(\boldsymbol{r}, \hat{\boldsymbol{s}}) \left[1 + \frac{1}{2}\exp(j\omega t) + \frac{1}{2}\exp(-j\omega t)\right], \tag{4.8}$$

where ω is the modulation frequency. For this specific time dependence, we can separate the RTE into three separate problems. 1) The first problem is the steady-state RTE where dw/dt is 0. This is equivalent to the first term in Eqn. 4.8 where the source is $\frac{1}{2}\Xi(\boldsymbol{r}, \hat{\boldsymbol{s}})$. 2) The second problem has a source given by $\frac{1}{4}\Xi(\boldsymbol{r}, \hat{\boldsymbol{s}}) \exp(j\omega t)$. The solution for the distribution function must also have a $\exp(j\omega t)$ dependence. 3) The third problem has the $\exp(-j\omega t)$ dependent. The solution to this third problem must be the complex conjugate of the solution to the second problem. Thus, we must solve the following two equations,

$$\begin{aligned} 0 &= -c_m \mu_{tot}(\boldsymbol{r}) w(\boldsymbol{r}, \hat{\boldsymbol{s}}) + \Xi(\boldsymbol{r}, \hat{\boldsymbol{s}}) + \boldsymbol{\mathcal{K}} w(\boldsymbol{r}, \hat{\boldsymbol{s}}) - c_m \hat{\boldsymbol{s}} \cdot \nabla w(\boldsymbol{r}, \hat{\boldsymbol{s}}), && (4.9) \\ j\omega w(\boldsymbol{r}, \hat{\boldsymbol{s}}) &= -c_m \mu_{tot}(\boldsymbol{r}) w(\boldsymbol{r}, \hat{\boldsymbol{s}}) + \Xi(\boldsymbol{r}, \hat{\boldsymbol{s}}) + \boldsymbol{\mathcal{K}} w(\boldsymbol{r}, \hat{\boldsymbol{s}}) - c_m \hat{\boldsymbol{s}} \cdot \nabla w(\boldsymbol{r}, \hat{\boldsymbol{s}}). && (4.10) \end{aligned}$$

Note that solving Eqn. 4.10 can be accomplished by solving Eqn. 4.9 but by replacing $\mu_{tot}(\boldsymbol{r})$ with $\mu_{tot}(\boldsymbol{r}) + j\omega/c_m$. Thus, we introduce a complex total coefficient. If we label the solution to Eqn. 4.9 as $w_1(\boldsymbol{r}, \hat{\boldsymbol{s}})$ and the solution to Eqn. 4.10 as $w_2(\boldsymbol{r}, \hat{\boldsymbol{s}})$, then the overall solution for the distribution function is,

$$\begin{aligned} w(\boldsymbol{r}, \hat{\boldsymbol{s}}, t) &= \frac{1}{2} w_1(\boldsymbol{r}, \hat{\boldsymbol{s}}) + \frac{1}{4} w_2(\boldsymbol{r}, \hat{\boldsymbol{s}}) \exp(j\omega t) + \frac{1}{4} w_2^*(\boldsymbol{r}, \hat{\boldsymbol{s}}) \exp(-j\omega t) && (4.11) \\ &= \frac{1}{2} w_1(\boldsymbol{r}, \hat{\boldsymbol{s}}) + \frac{1}{2} |w_2(\boldsymbol{r}, \hat{\boldsymbol{s}})| \cos(\omega t + \phi(\boldsymbol{r}, \hat{\boldsymbol{s}})), && (4.12) \end{aligned}$$

where $\phi(\boldsymbol{r}, \hat{\boldsymbol{s}})$ is the phase function of the complex solution $w_2(\boldsymbol{r}, \hat{\boldsymbol{s}})$.

4.2.6 The Neumann series If we take our RTE and integrate along $\hat{\boldsymbol{s}}$, then we arrive at and integral-form of the steady-state RTE which is given by

$$w(\boldsymbol{r}, \hat{\boldsymbol{s}}) = \frac{1}{c_m} \int_0^\infty dl\, \Xi(\boldsymbol{r} - \hat{\boldsymbol{s}}l, \hat{\boldsymbol{s}}) \exp\left[-\int_0^l dl'\, \mu_{tot}(\boldsymbol{r} - \hat{\boldsymbol{s}}l') \right] + \frac{1}{c_m} \int_0^\infty dl\, [\boldsymbol{\mathcal{K}} w](\boldsymbol{r} - \hat{\boldsymbol{s}}l, \hat{\boldsymbol{s}}) \exp\left[-\int_0^l dl'\, \mu_{tot}(\boldsymbol{r} - \hat{\boldsymbol{s}}l') \right]. \tag{4.13}$$

In operator notation, this is given by,

$$w = \boldsymbol{\mathcal{X}}\Xi + \boldsymbol{\mathcal{X}}\boldsymbol{\mathcal{K}}w, \tag{4.14}$$

where $\boldsymbol{\mathcal{X}}$ is the linear x-ray transform describing the propagation of photons interrupted by absorption and out-scatter. Eqn. 4.14 can be written as,

$$[\boldsymbol{\mathcal{I}} - \boldsymbol{\mathcal{X}}\boldsymbol{\mathcal{K}}]\, w = \boldsymbol{\mathcal{X}}\Xi. \tag{4.15}$$

A solution to the above equation is given by the Neumann series,

$$w(\boldsymbol{r}, \hat{\boldsymbol{s}}) = \boldsymbol{\mathcal{X}} \left[\Xi(\boldsymbol{r}, \hat{\boldsymbol{s}}) + \boldsymbol{\mathcal{K}}\boldsymbol{\mathcal{X}}\Xi(\boldsymbol{r}, \hat{\boldsymbol{s}}) + \boldsymbol{\mathcal{K}}\boldsymbol{\mathcal{X}}\boldsymbol{\mathcal{K}}\boldsymbol{\mathcal{X}}\Xi(\boldsymbol{r}, \hat{\boldsymbol{s}}) + \ldots \right]. \tag{4.16}$$

The first term represents the direct propagation of photons through the media without scatter. Subsequent terms with the operator $[\boldsymbol{\mathcal{K}}\boldsymbol{\mathcal{X}}]^n$ represent photons that scatter n times.

4.3 Spherical Harmonics

The distribution function $w(\boldsymbol{r}, \hat{\boldsymbol{s}})$ is a function of five variables x, y, z, θ, and ϕ. If we are to store a sampled distribution function in a computer using 100 samples for each variable, we would require 74 Gigabytes of storage assuming double precision. The integral operators described in the previous section are, in general, functions of ten variables. Spherical harmonics [3] are a natural way of splitting up the spatial and directional portions of a distribution function. Using spherical harmonics, we can write any distribution function as,

$$w(\boldsymbol{r}, \hat{\boldsymbol{s}}) = \sum_{l=0}^{\infty} \sum_{m=-l}^{l} W_{lm}(\boldsymbol{r}) Y_{lm}(\hat{\boldsymbol{s}}), \tag{4.17}$$

where $Y_{lm}(\hat{\boldsymbol{s}})$ are the spherical harmonic functions and $W_{lm}(\boldsymbol{r})$ are the spatially-dependent coefficients given by,

$$W_{lm}(\boldsymbol{r}) = \int_{4\pi} d\Omega\, Y_{lm}^*(\hat{\boldsymbol{s}}) w(\boldsymbol{r}, \hat{\boldsymbol{s}}). \tag{4.18}$$

The spherical harmonic functions are a complete and orthonormal basis for functions of $\hat{\boldsymbol{s}}$. Thus, if we truncate the infinite series in Eqn. 4.17 at $l = L$, then we would need $N_x \times N_y \times N_z \times (L+1)^2$ numbers to store a distribution function. Using 100 samples for x, y, and z and using L=4, we would require only 26 MB of storage for an entire distribution function.

To ease notation, we represent all the spherical harmonic coefficients $W_{lm}(\boldsymbol{r})$ as a vector $\boldsymbol{W}(\boldsymbol{r})$.

4.3.1 The scattering operator $\boldsymbol{\mathcal{K}}$ The scattering operator $\boldsymbol{\mathcal{K}}$ given in Eqn. 4.6 is assumed to depend on $\hat{\boldsymbol{s}} \cdot \hat{\boldsymbol{s}}'$. Thus, in spherical harmonics, the scattering kernel is diagonal. The full details of this derivation are given in [1]. The end result of this analysis is that the $\boldsymbol{\mathcal{K}}$ operator in spherical harmonics $\boldsymbol{\mathcal{K}}_{sphr}$ is given by a diagonal matrix operator with elements,

$$[\boldsymbol{\mathcal{K}}_{sphr}]_{l,l',m,m'} = \frac{4\pi}{2l+1} k_l \delta_{ll'} \delta_{mm'}, \tag{4.19}$$

where k_l is the scattering probability kernel represented in Legendre polynomials. For the Henyey-Greenstein scattering, $k_l = \frac{c_m \mu_{sc}}{4\pi}(2l+1)g^l$ where g is the anisotropy factor[1]. If g is 0 (i.e., isotropic scattering), then the scattering kernel has the effect of zeroing out all $W_{lm}(\boldsymbol{r})$ where $l > 0$ or treating each scatter as an isotropic source.

4.3.2 The propagation operator $\boldsymbol{\mathcal{X}}$ The x-ray transform operator $\boldsymbol{\mathcal{X}}$ is much more difficult in spherical harmonics because this operator does not diagonalize like the scattering operator does. The x-ray transform kernel in spherical harmonics $\boldsymbol{\mathcal{X}}_{sphr}$ is given by

$$[\boldsymbol{\mathcal{X}}_{sphr}]_{lm,l'm'}(\boldsymbol{r}, \boldsymbol{r}') = \tag{4.20}$$
$$\frac{1}{c_m} \frac{1}{|\boldsymbol{r} - \boldsymbol{r}'|^2} Y^*_{lm}\left(\frac{\boldsymbol{r} - \boldsymbol{r}'}{|\boldsymbol{r} - \boldsymbol{r}'|}\right) Y_{l'm'}\left(\frac{\boldsymbol{r} - \boldsymbol{r}'}{|\boldsymbol{r} - \boldsymbol{r}'|}\right) \exp\left[-\int_0^{|\boldsymbol{r}-\boldsymbol{r}'|} dt'\, \mu_{tot}\left(\boldsymbol{r} - t'\frac{\boldsymbol{r} - \boldsymbol{r}'}{|\boldsymbol{r} - \boldsymbol{r}'|}\right)\right].$$

The details of this derivation are given in [1]. The operator acting on a distribution function is represented as,

$$[\boldsymbol{\mathcal{X}}_{sphr}\boldsymbol{W}]_{lm}(\boldsymbol{r}) = \sum_{l'm'} \int_\infty d^3\boldsymbol{r}'\; [\boldsymbol{\mathcal{X}}_{sphr}]_{lm,l'm'}(\boldsymbol{r}, \boldsymbol{r}') W_{l'm'}(\boldsymbol{r}'). \tag{4.21}$$

This operator requires an integral over all space and a sum over all spherical harmonic coefficients to get just one of the coefficients in the output distribution function. This operator represents the principal computational effort of our technique, i.e., it takes the most time.

4.4 Voxelization

The distribution function in spherical harmonics, i.e., $W_{lm}(\boldsymbol{r})$ is still a function of the continuous vector $\boldsymbol{r}$. Any computer representation requires a discretization or voxelization. We define the function $\phi_i(\boldsymbol{r})$ to be 0 outside the ith voxel and 1 inside that voxel. Using this basis, we can represent the distribution function as,

$$W_{lm}(\boldsymbol{r}) = \sum_i W_{lmi} \phi_i(\boldsymbol{r}). \tag{4.22}$$

We now store just the coefficients W_{lmi} for all l, m, and voxels i. This voxelization does not changes the representation of the scattering operator since that particular kernel does not have a

[1]We must re-confirm this equation as it may be the key to our inconsistent results.

spatial dependence. However, our propagation kernel under this voxelized representation of the distribution function is

$$U_{lmi} = \frac{1}{\Delta v} \sum_{l'm'i'} V_{l'm'i'} \int_{\phi_i(\boldsymbol{r})} d^3r \int_{\phi_{i'}(\boldsymbol{r}')} d^3r' \, [\boldsymbol{\mathcal{X}}_{sphr}]_{lm,l'm'} (\boldsymbol{r}, \boldsymbol{r}'), \tag{4.23}$$

where $V_{l'm'i'}$ is an input distribution function represented using voxels and spherical harmonics, U_{lmi} is an output distribution function under the same representation, and Δv is the volume of a single voxel. We now see that the propagation kernel using spherical harmonics and a voxelized spatial representation is a matrix operator with elements

$$[\boldsymbol{\mathcal{X}}_{mat}]_{ll',mm',ii'} = \frac{1}{\Delta v} \int_{\phi_i(\boldsymbol{r})} d^3r \int_{\phi_{i'}(\boldsymbol{r}')} d^3r' \, [\boldsymbol{\mathcal{X}}_{sphr}]_{lm,l'm'} (\boldsymbol{r}, \boldsymbol{r}'). \tag{4.24}$$

Chapter 5: GPU Implementation and Numerical Algorithms

In this section, I will describe the GPU impelementation features and some of the more complicated routines used in the RTE Solver program.

5.1 GPU Implementation details

The RTE code has been implemented on the GPU with the following features:

1. Single Instruction Multiple data (SIMD) code
    (a) No branch instructions.
    (b) Boundary voxels follow the same code as non-boundary voxels.
2. Parallelized to considerable extent using different techniques so that:
    (a) Each threads memory requirements are minimum so that at the same time, many kernels can be run on the GPU.
    (b) Each kernel itself is not long, else the execution time increases and again many kernels cannot be run on the GPU at the same time.
    (c) The kernel has to perform optimized memory access to global memory.
3. Memory requirement as minimal as possible on the GPU.
4. Most terms are computed on the fly.
5. The following optimizations were implemented:
    (a) Using constant memory.
    (b) Using shared memory wherever possible.
    (c) Having a tissue map (lesser memory).

Speedup factors for the GPU implementation of this algorithm are around 300x faster than the equivalent CPU versions of the software. For more details on the GPU implementation, the reader is referred to the report along with this manual.

5.2 `GenerateSourceBeam`

For a laser source normally incident on the phantom shown above, we represent the source function as,

$$\Xi(\boldsymbol{r}, \hat{\boldsymbol{s}}) = \alpha\delta(z)\mathrm{circ}(x, y; 2)\delta(\hat{\boldsymbol{s}}). \tag{5.1}$$

The delta function on z indicates that the photons are incident only at one z-plane. The circ function represents the circular beam profile in x and y. The term α represents the source strength. Finally, the delta function on $\hat{\boldsymbol{s}}$ represents the fact that all the photons are traveling straight downward. Spherical harmonics are efficient at representing spherical sources but not at representing, for example, the laser source given in the above equation. Fortunately, we can analytically compute the first two terms of the Neumann series without resorting to spherical harmonics. The first term $\boldsymbol{\mathcal{X}}\Xi(\boldsymbol{r}, \hat{\boldsymbol{s}})$ is simply,

$$\boldsymbol{\mathcal{X}}\Xi(\boldsymbol{r}, \hat{\boldsymbol{s}}) = \frac{\alpha}{c_m}\mathrm{circ}(x, y; 2)\delta(\hat{\boldsymbol{s}})\exp(-\mu_{tot}z). \tag{5.2}$$

Because of the delta function of direction, this term is never converted into spherical harmonics. Instead, when the transmitted output images are computed, the effect of this first term is directly added into the transmitted image and the phase image.

The second term $\boldsymbol{\mathcal{K}}\boldsymbol{\mathcal{X}}\Xi(\boldsymbol{r}, \hat{\boldsymbol{s}})$ is given by,

$$\boldsymbol{\mathcal{K}}\boldsymbol{\mathcal{X}}\Xi(\boldsymbol{r}, \hat{\boldsymbol{s}}) = K(\cos(\theta))\frac{\alpha}{c_m}\mathrm{circ}(x, y; 2)\exp(-\mu_{tot}z). \tag{5.3}$$

Because we have gone through one scattering, the conversion of this distribution function to a spherical harmonic representation and voxel representation is given by,

$$[\boldsymbol{\mathcal{K}}\boldsymbol{\mathcal{X}}\Xi]_{lmi} = g^l\sqrt{\frac{2l+1}{4\pi}}\frac{\alpha\mu_s}{\Delta v\mu_{tot}}\mathrm{circ}(x_i, y_i; 2)\left[\exp(-\mu_{tot}(z_i + \Delta z)) - \exp(-\mu_{tot}z_i)\right] \tag{5.4}$$

when m is 0. When $m \neq 0$, then the value is 0. After this term is created, we then start the Neumann series.

5.3 `ComputeTransImage`

The distribution function is not directly measured by an imaging system. To produce transmitted, reflected, or phase image, we must integrate the distribution functions. (NOTE: This section describes the transmitted image but the description for the other images is similar.) Specifically, the mth pixel of a transmitted image is given by,

$$g_i = \int_P d^2r \int_{2\pi} d\Omega \int_0^\tau d_i(\boldsymbol{r}, \hat{\boldsymbol{s}}, t)w(\boldsymbol{r}, \hat{\boldsymbol{s}}, t). \tag{5.5}$$

In our initial studies we consider a contact detector which has the same x, y pixelization as our distribution function. Thus, the transmitted image is given by,

$$g_i = c\Delta x\Delta y \int_{2\pi} d\Omega\, w(x_i, y_i, z_{final}, \hat{\boldsymbol{s}}). \tag{5.6}$$

Here, the integral is over the 2π steradians because we only consider photons traveling in a downward direction. The speed of light appears in this equation to convert the number of photons to the number of photons crossing a plane in a one second interval. In spherical harmonics, this becomes,

$$g_i = c\Delta x \Delta y \int_0^{pi/2} d\theta \int_0^{2\pi} d\phi \sum_{lm} W_{lmi} Y_{lm}(\theta, \phi) \sin(\theta) \cos(\theta). \tag{5.7}$$

Note for $m \neq 0$, the integral over ϕ is 0. For $m = 0$ the ϕ integral is 2π. Thus, this equation simplifies to

$$g_i = c\Delta x \Delta y 2\pi \sum_l W_{l0i} \frac{\sqrt{2l+1}}{\sqrt{4\pi}} \int_0^{\pi/2} d\theta \, P_l(\cos(\theta)) \sin(\theta) \cos(\theta), \tag{5.8}$$

where we have also replaced the spherical harmonic function $Y_{lm}(\theta, \phi)$ with its form in Legendre polynomials. Making a change of variables we arrive at the final expression for the output image as,

$$g_i = c\Delta x \Delta y 2\pi \sum_l W_{l0i} \frac{\sqrt{2l+1}}{\sqrt{4\pi}} \int_0^1 du \, P_l(u) u \tag{5.9}$$

There are closed-form solutions for the integral shown in Eqn. 5.9

5.4 Other routines

5.4.1 `Neuman` This function invokes the Neuman series. The absorption and scattering kernel are called from this function repeatedly.

5.4.2 `prop_abs` This function calls the kerneks for performs the matrix multiplication shown in Eqn. 4.23. For every voxel in the output distribution function, we only consider those other voxels that are a distance `propThresh` or less.

5.4.3 `prop_scat` This function performs the scatter operation. This is a relatively simply routine but deserves mention as the scatter kernel is a major component of the Neumann series.

5.4.4 `CombineOuts` This function performs the mathematical operation shown in Eqn. 4.12. This is what combines the two Neumann series runs when intensity modulation is used.

5.4.5 `compute_prop_abs and compute_diagonal_abs` These are the main GPU kernels present in the code, apart from the other smaller kernels.

Bibliography

[1] H. .H. Barrett and K. J. Myers, *Foundations of Image Science*, John Wiley & Sons, Hoboken, New Jersey, 2004.

[2] `http://www.cygwin.com/`

[3] E. W. Hobson, *The Theory of Spherical and Ellipsoidal Harmonics*, Chelsea Pub. Co., 1955.

[4] `http://ndevilla.free.fr/iniparser/`