An Improved Monte Carlo Simulator for Optical Coherence Tomography

Junyao Pu

Honours Thesis Student

Dr. Chris Bidinosti, Dr. Sherif Sherif

Co-Supervisors

Dr. Melanie Martin

Committee Member

University of Winnipeg

Winnipeg, MB Canada

May 3, 2019

Abstract

Numerical simulations of light propagation in tissue can help us to understand the fundamental principles of optical coherence tomography (OCT) better, as well as to improve OCT methods and hardware. In my project, I developed a new OCT simulator for inhomogeneous turbid media based on an improved sequential Monte Carlo (MC) method that uses multinomial resampling. This improvement allows my simulator to overcome the weight degeneracy problem that would otherwise occur with standard sequential importance sampling methods. This approach may eventually reduce simulation time and/or increase the simulation accuracy. My simulator is still computationally expensive, however, and will benefit from further optimization. Future work will include the exploration of other advanced MC methods, such as systematic resampling and residual resampling, as well as implementation of the simulator to run on a on graphics processing unit (GPUs) for faster performance.

Acknowledgements

I would like to express my very great appreciation to my supervisors, Dr. Chris Bidinosti and Dr. Sherif Sherif, who spent countless hours helping me in this project. It was a great opportunity to work under their supervision. I would also like to thank my committee member Dr. Melanie Martin for her valuable feedback.

Thank you as well to all my family members for their continuous support and love.

Contents

List of Figures

1 Introduction

Optical coherence tomography (OCT) is a high resolution imaging technique used in medicine [1]. The goal of this thesis is to develop a fast and accurate numerical simulator of light propagation in turbid medium, which will ultimately improve and innovate the OCT method. The thesis is laid out in the following manner. First, I introduce some basic information and set up of the OCT system. In Chapter 2, I explain how to model light propagating in medium through the Radiative Transfer Equation (RTE). The general method of Monte Carlo integration is presented in the Chapter 3, along with a basic sequential Monte Carlo based solution of the RTE. I also describe how it can be improved with multinomial resampling algorithm. In Chapter 4, I describe the main computational components of my OCT simulator and how its structure is modified to include multinomial resampling. In Chapters 5 and 6, I conclude the results of validation tests of the new simulator and give some suggestions for future work that may further improve performance.

1.1 Optical Coherence Tomography

Optical coherence tomography was first introduced in 1991 [1]. It is an imaging technology that is similar to ultrasound except that it uses light waves instead of sound waves. Both imaging technologies use the interference between light or sound wave to construct a image. OCT has several advantages, which include high resolution, fast imaging times, and the use of non-ionizing optical radiation. The resolution of OCT is around 1 μm to 15 μm. One of the limitations of the method is its small penetration depth. Figure 1 illustrates the comparison of OCT with different imaging technologies. Due to the high resolution of OCT, it has several applications in ophthalmology, dentistry and dermatology. The most well-know application of OCT is to image retina of the eye. In addition, at universities and research institutions, research into OCT is becoming more and more popular due to its huge potential in the field of medicine.

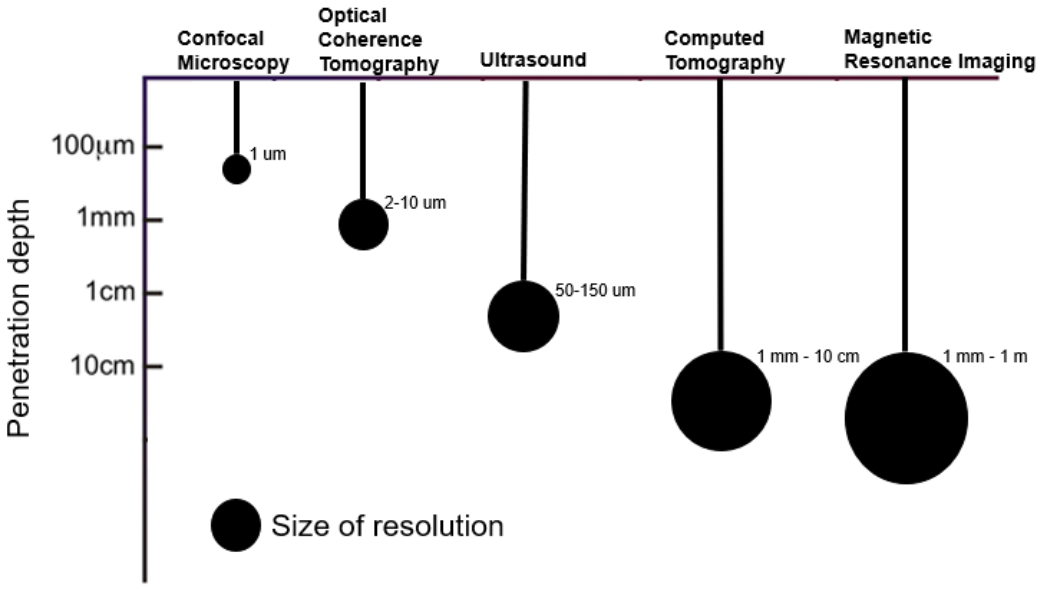

Figure 1: A comparison of OCT and other medical imaging tecnologies in resolution and penetration depth. The OCT has a resolution around 1 μm to 15 μm and penetration depth around 1 mm. It fills the gap between confocal microscopy and ultrasound.

The OCT method is based on interference between the reference light and the backscattered light from the sample. Fig. 2 shows the basic setup of the OCT method. Here, the light starts from a low coherence light source, which is then split into two equal beams going into the reference arm and sample arm respectively. Both beams are reflected back to the detector, the interference of the two are recorded for imaging construction.

Simulating the OCT signal in the medium by using a fast and accurate OCT simulator would be cheap and efficient. Furthermore, it can be used to improve the design of future OCT system. Thus, a powerful OCT simulator is extremely helpful tool in the project.

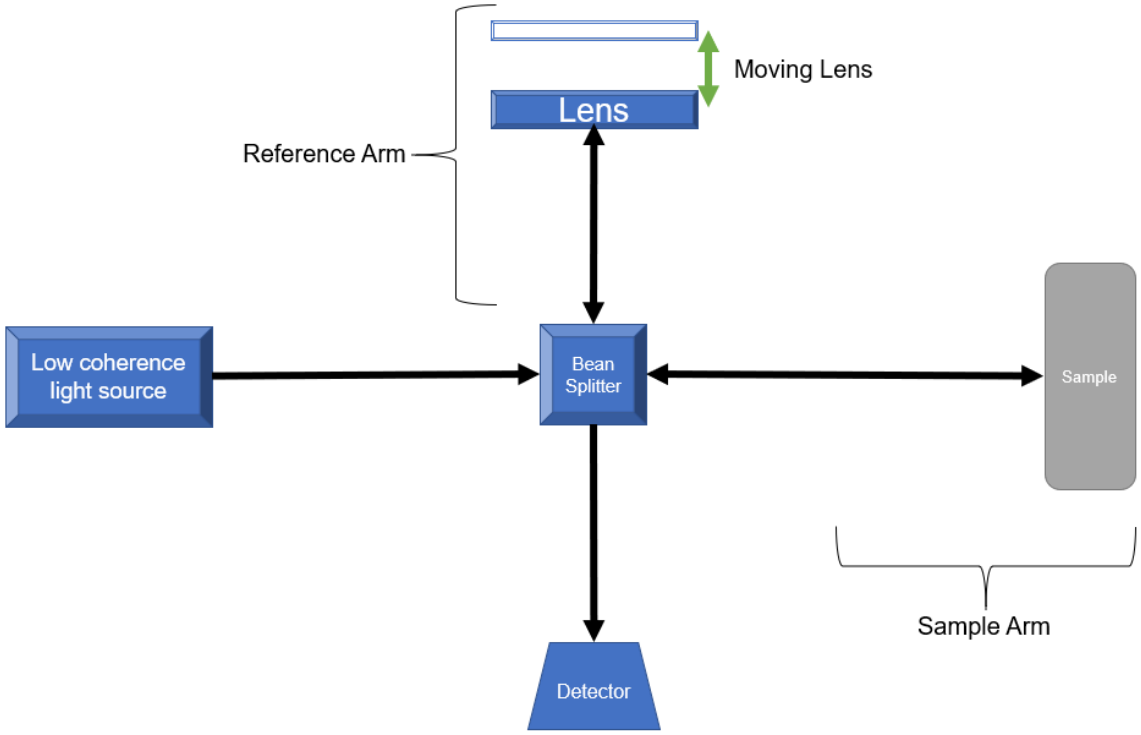

Figure 2: The basic setup for OCT method. Light is emitted from the low coherence light source. Later, it will be separated equally into two path, the reference arm and the sample arm. After that, both of them will be reflected back to the detector, the interference of them is recorded. By repeating those process while moving the lens of reference arm, we can get information of the sample in difference depth.

2 Modeling light propagation

To build an OCT simulator, I first need to understand how light is propagating in the sample medium. In the last 100 years, people have been using two approaches to model the light propagation [7, 8]. One is the electromagnetic (EM) approach, another one is the radiative transfer (RT) approach. The EM approach uses the electromagnetic wave to simulate the light propagation, but the RT approach only considers the energy transfer of light. In my project, the RT approach was selected to model light propagation, because the EM approach is only suitable for simple problem. In addition, the RT approach also simplifies my computational problem by only simulating the energy intensity.

Thus, in this thesis, I will focus on the RT approach, and will make use of the energy transfer in a medium given by the so-called radiative transfer equation (RTE). The RTE can be derived from Maxwells equations by using the pointing theorem and the law of energy conservation [7], as will be shown in the next section.

2.1 Derivation of the integro-differential RTE formula

Let's assume I have a medium which is homogeneous with dielectric constant ϵ and magnetic permeability μ. The electric field $\boldsymbol{E}$ and the magnetic field $\boldsymbol{H}$ are mutually orthogonal in the medium. The energy of light is given by the Poynting vector $\boldsymbol{S}$ [6][7] :

$$\boldsymbol{S} = \boldsymbol{E} \times \boldsymbol{H}^* . \tag{1}$$

I can write the fields in Eq. 1 as $\boldsymbol{E} = |\boldsymbol{E}|\,\hat{\boldsymbol{e}}$ and $\boldsymbol{H} = |\boldsymbol{H}|\,\hat{\boldsymbol{h}}$, which is in term of their magnitude and direction unit vector. Because the direction of energy propagation is given by the unitary vector $\hat{\boldsymbol{s}} = \hat{\boldsymbol{e}} \times \hat{\boldsymbol{h}}$, the Poynting vector can be rewritten as $\boldsymbol{S} = |\boldsymbol{E}||\boldsymbol{H}|\,\hat{\boldsymbol{s}}$.

Usually, the frequencies of light is much higher than the frequencies measured by detectors. So, the detector can only measure the time-averaged Poynting vector $\langle \boldsymbol{S}(\boldsymbol{r}) \rangle$. The time-averaged Poynting vector $\langle \boldsymbol{S}(\boldsymbol{r}) \rangle$ conservation of energy at position $\boldsymbol{r}$ is given by the

Poynting's theorem [7].

$$\frac{1}{c}\frac{d\big\langle \boldsymbol{S}(\boldsymbol{r})\big\rangle \cdot \hat{\boldsymbol{s}}}{dt} + \big\langle \frac{dP_{abs}(\boldsymbol{r})}{dV}\big\rangle + \nabla \cdot \big\langle \boldsymbol{S}(\boldsymbol{r})\big\rangle = 0\,, \tag{2}$$

where the three terms represent the loss of energy in the direction of the beam, the absorbed energy per unit volume in the sample, and the divergence of the beam, respectively. The constant c is the speed of light. The second term can also be written as $dP_{abs}/dV = \boldsymbol{J}\cdot\boldsymbol{E}$, where $\boldsymbol{J} = \sigma\boldsymbol{E}$ is the current density in a medium of electrical conductivity σ. Applying the law of energy conservation for any direction $\hat{\boldsymbol{v}}$ in Eq. 2, I can now write [7]

$$\frac{1}{c}\frac{d\big\langle \boldsymbol{S}(\boldsymbol{r})\big\rangle \cdot \hat{\boldsymbol{v}}}{dt} + \big\langle \frac{dp_{abs}(\boldsymbol{r})}{dV}\big\rangle(\hat{\boldsymbol{s}}\cdot\hat{\boldsymbol{v}}) + \hat{\boldsymbol{v}}\cdot\nabla(\big\langle \boldsymbol{S}(\boldsymbol{r})\big\rangle \cdot \hat{\boldsymbol{v}}) = 0\,. \tag{3}$$

Now define $I(\boldsymbol{r},\hat{\boldsymbol{v}})$ to be the specific intensity, which is the energy per unit area per unit solid angle transfer at $\boldsymbol{r}$ with direction $\hat{\boldsymbol{v}}$:

$$I(\boldsymbol{r},\hat{\boldsymbol{v}}) = \frac{1}{4\pi\delta V}\int_{\delta V}\big\langle S(\boldsymbol{r}-\boldsymbol{r}')\big\rangle\cdot\hat{\boldsymbol{v}}d^3\boldsymbol{r}'\,. \tag{4}$$

Also define $P(\hat{\boldsymbol{v}}',\hat{\boldsymbol{v}})$ to be the scattering phase function, which is that portion of energy that changes its propagation direction from $\hat{\boldsymbol{v}}'$ to $\hat{\boldsymbol{v}}$ [7]. The scattering phase function depends on the scattering behaviour of the medium and it is usually estimated by the Henyey-Greenstein's phase function [7].

Now, averaging Eq. 3 over a small differential volume, the first term becomes

$$\frac{1}{c}\frac{d}{dt}I(\boldsymbol{r},\hat{\boldsymbol{v}}') \tag{5}$$

and the second term is

$$\mu_a I(\boldsymbol{r},\hat{\boldsymbol{v}}')\,, \tag{6}$$

where μ_a is just the absorption coefficient.

Rewrite the time-averaged Poynting vector, $\left\langle \boldsymbol{S}(\boldsymbol{r}) \right\rangle$ as the sum of two functions $\left\langle \boldsymbol{S}^{(inc)}(\boldsymbol{r}) \right\rangle$ and $\left\langle \boldsymbol{S}^{(sc)}(\boldsymbol{r}) \right\rangle$, which are the contributions from non-scattered and scattered light. In Eq. 3, the third term has contributions from three components: one is from $\left\langle \boldsymbol{S}^{(inc)}(\boldsymbol{r}) \right\rangle$ and the other two are due to $\left\langle \boldsymbol{S}^{(sc)}(\boldsymbol{r}) \right\rangle$. These three components are the spatial change in the specific intensity

$$\hat{\boldsymbol{v}} \cdot \nabla I(\boldsymbol{r}, \hat{\boldsymbol{v}}) \,; \tag{7}$$

the energy lost by scattering

$$\mu_s I(\boldsymbol{r}, \hat{\boldsymbol{v}}) \,, \tag{8}$$

where the μ_s is the scattering coefficient; and the energy increase due to the scattered light from adjacent volumes

$$-\mu_s \int_{4\pi} I(\boldsymbol{r}, \hat{\boldsymbol{v}}') p(\hat{\boldsymbol{v}}', \hat{\boldsymbol{v}}) d\hat{\boldsymbol{v}}' \,, \tag{9}$$

where $d\hat{\boldsymbol{v}}'$ is the differential solid angle and μ_s is the scattering coefficient.

I can now write the familiar integro-differential form of the RTE from the above defined functions [7][18]:

$$\frac{1}{c}\frac{d}{dt} I(\boldsymbol{t}, \hat{\boldsymbol{v}}) + \hat{\boldsymbol{v}} \cdot \nabla I(\boldsymbol{r}, \hat{\boldsymbol{v}}) + \mu_t I(\boldsymbol{r}, \hat{\boldsymbol{v}}) = \epsilon(\boldsymbol{r}, \hat{\boldsymbol{v}}) + \mu_s \int_{4\pi} I(\boldsymbol{r}, \hat{\boldsymbol{v}}') p(\hat{\boldsymbol{v}}', \hat{\boldsymbol{v}}) d\hat{\boldsymbol{v}}' \,. \tag{10}$$

Where c is the speed of light in the medium. $I(t, \hat{\boldsymbol{v}})$ is the specific optical intensity, representing the power per unit area that flows in the direction $\hat{\boldsymbol{v}}$ at time t. $\mu_t = \mu_s + \mu_a$ is the extinction coefficient, they are optical properties of mediums. $\epsilon(\boldsymbol{r}, \hat{\boldsymbol{v}})$ is the source term, and $p(\hat{\boldsymbol{v}}', \hat{\boldsymbol{v}})$ is the scattering phase function, which is the probability of photon propagating in direction $\hat{\boldsymbol{v}}$ to be scattered by $\boldsymbol{v}$.

2.2 The solution of RTE as a Neumann series

The RTE represents all the physics of light propagation in a medium. If I want to simulate light propagation in my sample medium, I must solve the RTE. However, it is difficult to

solve the RTE analytically, so I must estimate its solution using some numerical method. I will now show how the RTE can be rewritten with a solution that can be evaluated by a numerical Monte Carlo (MC) method [5].

To start let $X(\boldsymbol{r}, \hat{\boldsymbol{v}})$ be the photon emission rate density, so the right hand side of Eq. 10 is

$$X(\boldsymbol{r}, \hat{\boldsymbol{v}}) = \epsilon(\boldsymbol{r}, \hat{\boldsymbol{v}}) + \mu_s \int_{4\pi} I(\boldsymbol{r}, \hat{\boldsymbol{v}}') p(\hat{\boldsymbol{v}}', \hat{\boldsymbol{v}}) d\hat{\boldsymbol{v}}' \,. \tag{11}$$

Because I only consider with steady state solution of Eq.10 in this thesis, the first term is time varying term, I will neglect this term. The second term of the Eq. 10 is the inner product of $\hat{\boldsymbol{v}} \cdot \nabla I(\boldsymbol{r}, \hat{\boldsymbol{v}})$, which is the spatial change of the specific intensity along the direction $\hat{\boldsymbol{v}}$. Letting dk be the differential distance along the direction, one sees from Fig. 3 that ∇I is equal to $\frac{dl}{dk}$.

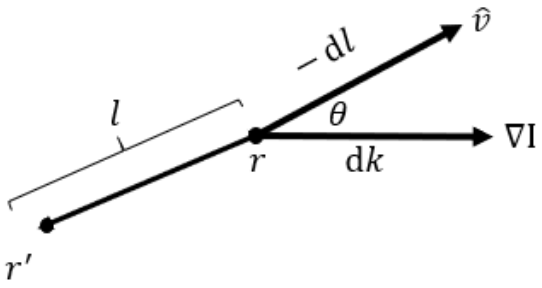

Figure 3: The change of rate specific intensity along the direction $\hat{\boldsymbol{v}}$, it represents the inner product of $\hat{\boldsymbol{v}}$ and $\nabla I(\boldsymbol{r}, \hat{\boldsymbol{v}})$.

Also from Fig. 3, one sees that θ is the angle between the gradient and direction $\hat{\boldsymbol{v}}$, and l is the distance between $\boldsymbol{r}$ and $\boldsymbol{r}'$. As a result, the second term of the RTE can be written as

$$\hat{\boldsymbol{v}} \cdot \nabla I(\boldsymbol{r}, \hat{\boldsymbol{v}}) = \frac{dI(\boldsymbol{r}, \hat{\boldsymbol{v}})}{dk} cos(\theta) = \frac{dI(\boldsymbol{r}, \hat{\boldsymbol{v}})}{dk} \frac{dk}{-dl} = -\frac{dI(\boldsymbol{r}, \hat{\boldsymbol{v}})}{dl} \,. \tag{12}$$

Using Eq. 11 and 12, I can rewrite Eq. 10 as

$$-\frac{dI(\boldsymbol{r}, \hat{\boldsymbol{v}})}{dl} + \mu_t I(\boldsymbol{r}, \hat{\boldsymbol{v}}) = X(\boldsymbol{r}, \hat{\boldsymbol{v}}) \,. \tag{13}$$

Next I multiply both sides by $e^{(-\mu_t l)}$, and let $\boldsymbol{r}'$ be the point along the direction $-\hat{\boldsymbol{v}}$ away from $\boldsymbol{r}$ by a distance l, so $\boldsymbol{r}' \equiv \boldsymbol{r} - l \cdot \hat{\boldsymbol{v}}$. Then, integrate over l to give

$$I(\boldsymbol{r}, \hat{\boldsymbol{v}}) = \int_{l=0}^{\infty} \epsilon(\boldsymbol{r} - l\hat{\boldsymbol{v}}, \hat{\boldsymbol{v}}) e^{-\mu_t l} dl + \mu_s \int_{l=0}^{\infty} \int_{4\pi} I(r - l\hat{\boldsymbol{v}}, \hat{\boldsymbol{v}}) e^{-\mu_t l} \boldsymbol{p}(\hat{\boldsymbol{v}}', \hat{\boldsymbol{v}})\, dl\, d\hat{\boldsymbol{v}}' \,. \tag{14}$$

Now, let $\boldsymbol{p}$ be the state of the radiation defined by both position $\boldsymbol{r}$ and propagation direction $\boldsymbol{v}$, so $\boldsymbol{p} \equiv (\boldsymbol{r}, \hat{\boldsymbol{v}})$, $\boldsymbol{p}' \equiv (\boldsymbol{r}', \hat{\boldsymbol{v}}')$ and $d\boldsymbol{p}' \equiv dl d\hat{\boldsymbol{v}}'$, I can write Eq. 14 as

$$I(\boldsymbol{p}) = \tilde{\epsilon} + \int I(\boldsymbol{p}') B(\boldsymbol{p}', \boldsymbol{p}) d\boldsymbol{p}' \,, \tag{15}$$

where

$$\tilde{\epsilon}(\boldsymbol{p}) = \frac{1}{\mu_t} \int_{l=0}^{\infty} \epsilon(\boldsymbol{r} - l\hat{\boldsymbol{v}}, \hat{\boldsymbol{v}}) \mu_t e^{-\mu_t l} dl \tag{16}$$

and

$$B(\boldsymbol{p}', \boldsymbol{p}) = (\frac{\mu_s}{\mu_t}) \mu_t e^{-\mu_t l} \boldsymbol{p}(\hat{\boldsymbol{v}}, \hat{\boldsymbol{v}}) \,. \tag{17}$$

Equation 15 is the RTE re-written as a Fredhlom integral equation of the second kind. The solution of Fredholm integral equation of the second kind is known and can be written as a Neumann series of integrals with increasing dimension [5]. For my problem the solution takes the following form:

$$\begin{aligned} I(\boldsymbol{p}) = \tilde{\epsilon}(\boldsymbol{p}) &+ \int \tilde{\epsilon}(\boldsymbol{p}_1) B(\boldsymbol{p}_1, \boldsymbol{p}) d\boldsymbol{p}_1 + \iint \tilde{\epsilon}(\boldsymbol{p}_1) B(\boldsymbol{p}_1, \boldsymbol{p}_2) B(\boldsymbol{p}_2, p) d\boldsymbol{p}_1 d\boldsymbol{p}_2 \\ &+ \iiint \tilde{\epsilon}(\boldsymbol{p}_1) B(\boldsymbol{p}_1, \boldsymbol{p}_2) B(\boldsymbol{p}_2, \boldsymbol{p}_3) B(\boldsymbol{p}_3, \boldsymbol{p}) d\boldsymbol{p}_1 d\boldsymbol{p}_2 d\boldsymbol{p}_3 + \dots \,, \end{aligned} \tag{18}$$

where the first term represents non-scattered photons, the second term represents single-scattered photons, and so on. I now have a mathematical expression for the solution of RTE written as a series of integrals. It is still very complex, however, and beyond analytic solution. In the next chapter, I will show how to evaluate those integrals numerically using MC methods with some advanced techniques.

3 Monte Carlo methods for solving the RTE

In practice, there are many integrals that cannot be solved analytically. Fortunately, Monte Carlo (MC) methods provide a convenient and accurate means to evaluate complex integrals numerically. As noted above, for my OCT simulations, light propagation is modeled by the RTE, which has a solution in terms of the Neumann series of integrals, but cannot be solved analytically. As a result, MC methods are an idea tool for my OCT simulator. In this chapter, I will first present how to evaluate a simple integral using MC methods, and then discuss more advanced MC techniques for evaluating the complex integrals of Eq. 18. I will end the chapter with a discussion of the drawbacks of those techniques and how one can improve them. These improvements form the work of this thesis.

3.1 Monte Carlo integration

Consider the following integral

$$I = \int f(x)dx\,, \tag{19}$$

which can be rewritten as the integral of an function $g(x)$ multiplied by a probability target distribution function $p(x)$:

$$I = \int f(x)dx = \int g(x)p(x)dx \tag{20}$$

Equation 20 can be estimated as the summation [19]

$$\hat{I} = \frac{1}{N}\sum_{i=1}^{\infty} g(x_i)\,, \tag{21}$$

where x_i is randomly sampled from the target distribution function $p(x)$ and N is the total number of samples. Numerical methods such as these that rely on random sampling are called Monte Carlo in reference to the famous gambling casino. In the limit $N \to \infty$, $\hat{I} = I$. For finite N,the error for this estimation is given by $|e_1| \cong \frac{\sigma_1}{\sqrt{N}}$, where variance ${\sigma_1}^2$ is given

by [20]

$$\sigma_1{}^2 \equiv \int g^2(x)p(x)dx - I^2 \,. \tag{22}$$

Let's consider a situation from which the target distribution function $p(x)$ is difficult or impossible to sample. In this case, Evaluate the integral by using the basic MC method above with a new sampling technique known as importance sampling. This approach allows one to sample from a simpler, more convenient, proposal distribution function, instead of sampling from the original target distribution. In order to capture the features of $p(x)$, however, importance weights must be used to statistically weight how likely the sample would be if it were coming from the target distribution, as explained below.

For the purpose of importance sampling, which is illustrated in Fig. 4. I can rewrite Eq. 20 as

$$I = \int g(x)p(x)dx = \int g(x)\frac{p(x)}{q(x)}q(x)\,dx\,, \tag{23}$$

where $q(x)$ is the proposal distribution function which is easy to sample from, and it has to be greater than zero. Letting $w(x) = \frac{p(x)}{q(x)}$ be the importance weight, Eq. 23 becomes

$$I = \int_a^b [g(x)w(x)]q(x)\,dx\,. \tag{24}$$

Now the function in this integral is $[g(x)w(x)]$ and the distribution function is $q(x)$, thus the MC estimation of this integral is equal to

$$\hat{I} = \frac{1}{N}\sum_{i=1}^{\infty} g(x_i)w(x_i)\,, \tag{25}$$

where x_i are randomly sampled from the proposal distribution $q(x)$ and again N is the total number of samples. The error of the estimation here is $|e_2| \cong \frac{\sigma_2}{\sqrt{N}}$, where the variance $\sigma_2{}^2$ is given by [20]

$$\sigma_2{}^2 \equiv \int w^2(x)q(x)dx - I^2 \,. \tag{26}$$

Figure 4: If I have a target distribution (left), it is hard to sample from. I will use a proposal distribution (middle), it is a uniform distribution and easy to sample. Therefor, my samples are from the proposal distribution with important weight (left). The important weight is represented by the red sphere, which tells me how likely those samples are from the target distribution.

3.2 The sequential Monte Carlo method

Let's consider Eq. 18 again, where I am trying to evaluate this complex equation to model light propagation in my OCT sample. It is a series of integrals, with increasing dimensionality, that could be evaluated individually using the standard MC method with importance sampling described above. However, such an approach would be very inefficient, as computational complexity increases at least linearly with dimension [9]. A more suitable evaluation of this kind of equation can be achieved with the sequential Monte Carlo method (SMC) described below, which allows for a more efficient use of samples.

3.2.1 Sequential importance sampling

The core idea of SMC is to use sequential importance sampling (SIS) technique to evaluate the integrals of Eq. 18 sequentially [19], whereby the numerical samples used to evaluate previous integral will be reused to evaluate the next integral. To understand this method, Suppose I have a series of proposal distributions with increasing dimension given by

$$q_n(x_{1:n}) = q_{n-1}(x_{1:n-1})q_n(x_n|x_{1:n-1})\,, \tag{27}$$

which is equivalent to

$$q_n(x_{1:n}) = q_{x_1} \prod_{k=2}^{n} q_k(x_k|x_{1:k-1}) \,, \tag{28}$$

where $x_{1:n} := x_1, x_2, ..., x_n$. This means that I have sample $x_{1:n}$ from $q_n(x_{1:n})$. For example, one would sample x_1 from $q_1(x_1)$ at $n = 1$, sample x_2 from $q_2(x_2|x_1)$ at $n = 2$ and sample x_n from $q_n(x_n|x_{n-1})$ at $n = n$ [4]. This simplifies my problem, since I sample from a relatively low dimensional distribution.

3.2.2 Evaluating the RTE using SMC

In Eq. 18, the solution of RTE is written as a Newmann series. Let Q be the inner product of $I(p)$ and $h(p)$:

$$Q \cong \int \left(\sum_{k=1}^{K} \int_{(k+2)_{dim}} \tilde{\epsilon}(\boldsymbol{p}_1) \prod_{i=1}^{k-1} B(\boldsymbol{p}_i, \boldsymbol{p}_{i+1}) d\boldsymbol{p}_1 ... d\boldsymbol{p}_i) \right) h(\boldsymbol{p}_k) d\boldsymbol{p}_k \,, \tag{29}$$

where the Q is the quantity of interest, $I(p)$ is the specific optical intensity, and $h(p)$ is the weighting function. Plugging Eq. 16 and Eq. 17 into Eq. 29, I can rewrite Q as

$$Q \cong \sum_{k=1}^{k} \frac{1}{\mu_t} \left(\frac{\mu_s}{\mu_t} \right)^{k-1} \int_{(k+2)_{dim}} \epsilon(r_0, \hat{v}_1) f_1(l_1) \prod_{i=i}^{k-1} T(\boldsymbol{p}_i, \boldsymbol{p}_{i+1}) h(\boldsymbol{p}_k) dr_0 dl_1 d\check{v}_1 d\boldsymbol{p}_2 ... d\boldsymbol{p}_k \,, \tag{30}$$

where $\boldsymbol{r}_{i+1} = \boldsymbol{r}_i + l_{i+1}\hat{\boldsymbol{v}}_{i+1}$ and $T(\boldsymbol{p}_i, \boldsymbol{p}_{i+1}) = f_l(l_{i+1}) p(\hat{\boldsymbol{v}}_i, \hat{\boldsymbol{v}}_{i+1})$. This process is shown diagrammatically in Fig. 5 with regard to a photon packet propagating in the OCT sample. The term $\epsilon(\boldsymbol{r}_0, \hat{\boldsymbol{v}}_1) f_1(l_1)$ can be considered to be a scaled probability distribution function of $\boldsymbol{p}_1$, while $T(\boldsymbol{p}_i, \boldsymbol{p}_{i+1})$ is a Markov probability transition kernel [5]. As a result, $\epsilon(\boldsymbol{r}_0, \hat{\boldsymbol{v}}_1)$ can be normalized by dividing the total power of the light source. By sampling from these probability functions, the MC estimation of Q is [5]

$$Q \cong \sum_{k=1}^{k} \frac{1}{\mu_t} \left(\frac{\mu_s}{\mu_t} \right)^{k-1} \sum_{n=1}^{N} h(\boldsymbol{p}_k^{(n)}) \tag{31}$$

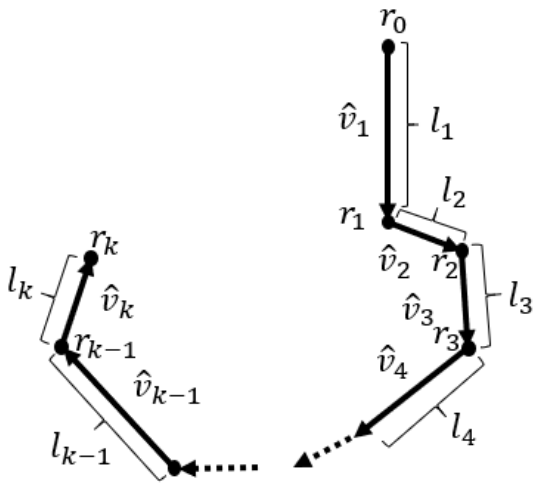

Figure 5: The photon packet has initial position r_{i-1}, it propagates in medium with a randomly sampled moving distance l_i and propagation direction $\hat{v}_i$. Those randomly sampled parameters l_i and $\hat{v}_i$ are used to calculate the next position r_i. In the MC simulation, the current propagation direction $\hat{v}_i$ and position r_i are used to estimate the solution of RTE.

3.3 Improving SMC with resampling

The SMC method is a powerful tool for evaluating complex integrals like those from the solution of RTE in Eq. 18. The drawback of the SMC method, however, is an effect known as weight degeneracy, whereby the variance of importance weights increases exponentially as the number of sampling steps [19]. One can fix this problem by using resampling methods. In this section, we will introduce a resampling method, and describe how I will implement this method in my simulator.

The idea behind resampling is to preserve samples with large weights while discarding those with small weights. The selection probability is based on each sample's normalized weight. Figure 6 illustrates the resampling process. Because I terminate the tracing on those low weight samples by discarding them earlier, I would expect a reduction in computational time. In addition, by preserving those high weight samples, one expects an increase in the simulator's accuracy.

There are several algorithms available for resampling, such as multinomial resampling,

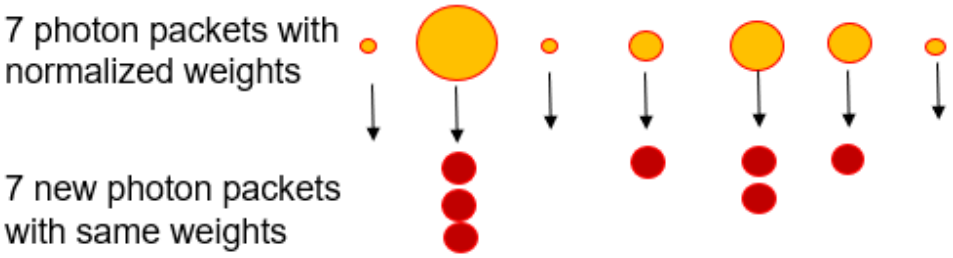

Figure 6: initially I have 7 samples with importance wights from previous distribution, the normalized importance weights are represented by the size of sphere (Yellow). The resampling method is that selects new samples from those 7 samples according to their normalized weight. And set the new sample's weight W = 1 (Red), therefor, they are equally weighted in the new sample set.

systematic resampling and residual resampling [17]. I have chosen to use multinomial resampling in my simulator. In this resampling process, the selection probability of each sample depends on its normalized weight. Thus the number of times N_i for each sample in the population set is selected as a binomial distribution, Bin(N_i,W_i). For a series of samples, it is distributed according to a multinomial distribution [17]. the multinomial resampling process is illustrated in Fig. 7

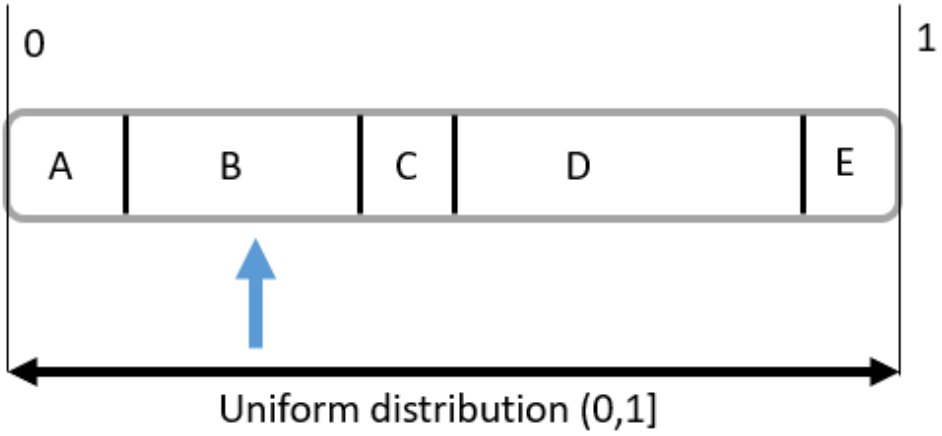

Figure 7: Samples A B C D and E are stored in a column, the column has space from 0 to 1. Those samples occupy the column's space according to their normalized weight. I select one sample by randomly generates a number from a uniform distribution $(1, 0]$. The sample is selected as the number generated by uniform distribution points on that sample's space, which mean sample with large space has big change to be selected [21]

4 The OCT simulator

In 1998, Smithies *et al.* introduced the first OCT simulator [10]. It was only capable of simulating homogeneous turbid media, which is to say media that has the same optical properties everywhere. In 1999, Yao and Wang built an OCT simulator that can simulate multilayered turbid media, *ie.*, media with different layers and different optical properties [11]. Later, Lima *et al.* used an advanced importance sampling technique to speed up Yao's OCT simulator for 100 times [12]. In 2007, Kirillin *et al.*, applied sinusoidal functions to simulate the OCT signal for media with multilayers [16]. More recently, Periyasamy and Pramanik introduced an OCT simulator which can simulate spherical, cylindrical, ellipsoidal or cuboidal shaped object in multilatered turbid media [15].

Dr. Sherif Sherif and his group at the University of Manitoba have been developing advanced OCT simulators for many years now. Malektaji *et al.* introduced an OCT simulator for any shaped turbid medium constructed from tetrahedrons with different optical properties [2]. This allows arbitrarily shaped regions with desired accuracy, and reduces the simulation computational time. In addition, they also used an advanced importance sampling method to speed up the simulator with required accuracy. It was still computationally expensive, however, and in 2017 Escobar *et al.* implemented the simulator on graphics processing units (GPUs) with the Compute Unified Device Architecture (CUDA) platform by NVIDIA. The GPU-based simulator was more than one order of magnitude faster than the CPU-based simulator [3].

Late in 2018, Sobhy *et al.* derived the numerical solution of the Radiative Transfer Equation for the SMC simulation of photon transport in homogenous turbid media [4][5], which forms the basis of this work. It is hoped that this approach will ultimately provide greater accuracy, and it is my goal to develop appropriate CPU-based algorithms first before pursuing GPU-based parallelization. I will now review the workflow of a CPU-based OCT simulator, and highlight how the simulator structure is changed after multinomial resampling algorithm is applied.

4.1 Simulator workflow

In the OCT simulator, the medium is constructed by a homogeneous background slab embedded with arbitrarily shaped objects, as shown in Fig. 8. The objects are built up from many small tetrahedrons, and different objects can have different optical parameters, such as the scattering coefficient μ_s, absorption coefficient μ_a refractive index n, and the anisotropy factor g [2]. For light propagation, I do not model individual quantized photons, but rather generate 'photon packet' with a macroscopic properties such as intensity. This approach reduces computational time. The simulator workflow is illustrated in Fig. 9. In the next section, I will discuss some important components of the simulator.

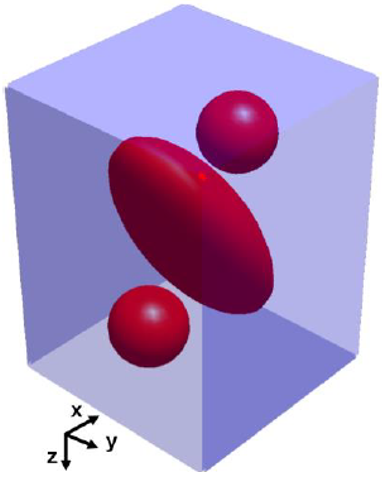

Figure 8: Visualization of the simulated medium with a homogeneous background slab contain an ellipsoid and two spheres. Each object in the homogeneous slab are given different optical properties. The light can propagates in the object and experiences different absorption and scattering event due to the given optical properties.

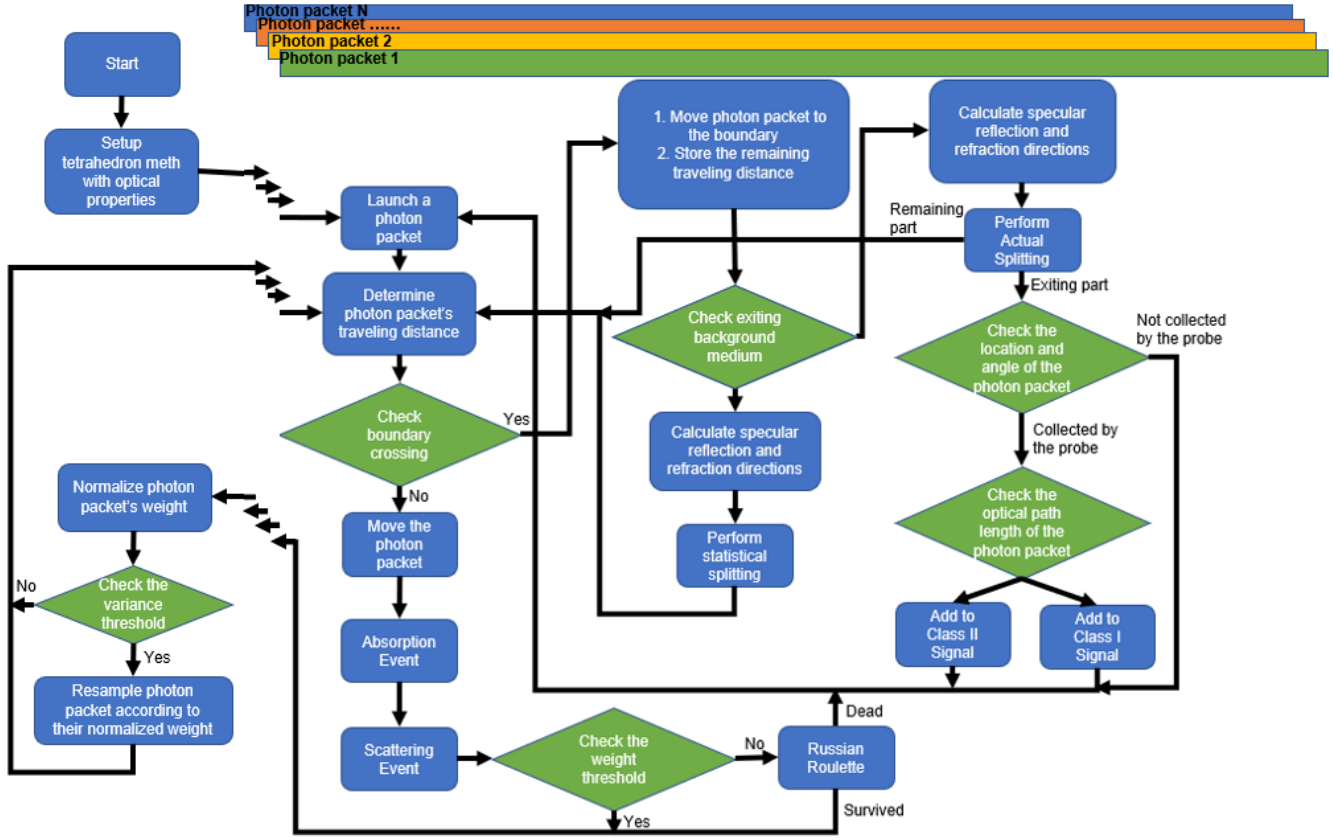

Figure 9: In this thesis, let's focus on those alive photon packets. The workflow of the simulator is first construct the simulated medium, then launch those photon packet in the medium. Those photon packets are propagating in the medium with absorption and scattering. If they are survived at each propagation step, they will be resampled. Then repeat the propagation in the medium again.

4.1.1 Launch a photon packet

Once the simulated medium is ready, photon packets are launched from the same location representing the source light from the OCT probe. The travel direction of the photon packet is initially set normal to the surface of the medium and the intensity is given an initial weight $W = 1$. The weight will be reduced due to absorption and scattering events while it propagating in medium.

4.1.2 Travel distance between interaction sites

After the photon packet is launched, it propagates inside the medium. The photon packet experiences absorption and scattering events at each interaction site. The distance between the interaction sites l is a random variable, representing the stochastic process of light propogation in a medium. The cumulative distribution function (CDF) of this free path l is given by

$$F_l = 1 - exp(-\mu_t l) \tag{32}$$

where $\mu_t = \mu_s + \mu_a$ is the total extinction coefficient of the regions in which photon packet travelling.

4.1.3 Absorption and scattering events

For the photon packet travelling in the medium, a portion of it is absorbed at each absorption event. The change of the weight of the packet is given by

$$\triangle W = W \cdot \frac{\mu_a}{\mu_t}\,, \tag{33}$$

where the μ_a is the absorption coefficient and the μ_t is the extinction coefficient of the medium. After the absorption event, the photon packet will have a weight

$$W = W - \triangle W\,. \tag{34}$$

For the scattering event, the direction of photon packet will randomly change according to the Henye-Greenstein's scattering phase function [7].

4.1.4 Termination of a photon packet

The tracing of a photon packet will be terminated if the packet exits the background medium, is collected by the probe, or is completely absorbed by the medium. However, tracing a

photon packet until its weight is reduced to a very small number or zero is computational expensive. Therefore, the tracing is terminated at a certain threshold. In my simulator, the threshold is given 10^{-4} which is small enough to consider killing the photon packet. If the weight is less than 10^{-4}, the photon packet has $\frac{1}{m}$ probability to be survived with a new weight ($W \cdot m$), where m is 10 in my simulator (same as previous simulator). Otherwise, the photo packet is dead, and the simulator will start to trace next photon packet. This method is called Russian Roulette, and it is used to speed up my simulation.

4.1.5 Class I and Class II OCT signal estimation

When the photon packet exits the medium and is collected by the probe, it is used to estimate the OCT signals as either Class I or Class II type. Class I diffusive reflectance represent the ballistic and quasi-ballistic photons; they have undergone only a single scattering event. Class II diffusive reflectance are the multiple scattered photons.In OCT simulation, Class II diffusive reflectance is the main source of error for the simulation and the limitation of OCT imaging depth [10] [13].

The fiber probe in my simulator are from Yao and Wang [11], which has radius d_{max} and acceptance angle θ_{max}. The spatial-temporal indicator function is used to classify the photon packet into Class I and Class II. The Class I diffuse reflectance spatial-temporal indicator function I_1 is given by [2] [4]

$$I_1(z,i) := \begin{cases} 1, & \text{for } l_c > |\triangle l^i - 2z_m^{(i)} ax|, 2r^{(i)} < \Phi_f, \theta_z^{(i)} < \theta_f, |\triangle l^{(i)} - 2z| < l_c \\ 0, & \text{otherwise} \end{cases}, \quad (35)$$

where l_c is the coherence length of the source, z_{max} is the maximum depth of the photon packet, $\triangle l^i$ is the length of the i^{th} photon packet, $r^{(i)}$ is the distance between the exiting point to the probe point and $\theta_z^{(i)}$ is the angle between the photon packet direction and z-axis.

Similarly, the Class II diffuse reflectance I_2 is given [2] [4]

$$I_2(z,i) := \begin{cases} 1, & \text{for } l_c < |\triangle l^i - 2z_m^{(i)} ax|, 2r^{(i)} < \Phi_f, \theta_z^{(i)} < \theta_f, |\triangle l^{(i)} - 2z| < l_c. \\ 0, & \text{otherwise.} \end{cases} \tag{36}$$

For both Class I and Class II, their diffusive reflectance $R_1(z)$ and $R_2(z)$ at a depth z is estimated by [2] [4]

$$R_{1,2}(z) = \frac{1}{N} \sum_{i=1}^{N} I_{1,2}(z,i) L(i) W(i)\,, \tag{37}$$

where N is the total number of photon packet, $L_{(i)}$ is the likelihood ratio in the i^{th} photon packet, and $W(i)$ is the weight of the photon packet. The variance of this estimation is given [2] [4]

$$\hat{e}_{1,2}^2(z) = \frac{1}{N(N-1)} \sum_{i=1}^{N} \left(I_{1,2}(z,i) L(i) W(i) - \hat{R}_{I_1,I_2}(z) \right)^2 \tag{38}$$

4.1.6 Resampling to reduce computational time

As mentioned above, resampling is a strategy to fix the weight degeneracy problem of SIS. In the simulator, the normalized weight of the photon packet is used as the selection probability for resampling. The simulator stops at each propagation step and waits for all photon packets to finish before calculating their normalized weights:

$$W_i' = \frac{W_i}{\sum_{i=1}^{N} W_i}\,, \tag{39}$$

where W_i is the weight of the i-th photon packet and N is the total number of packet. The variance of the normalized weights is

$$\sigma^2 = \frac{\sum_{i=1}^{N} (W_i - \overline{W})^2}{N}\,, \tag{40}$$

where $\overline{W}$ is the mean of weights. The calculated variance is compared with a variance threshold to chose doing resampling or not doing resampling, in my simulator the variance

threshold is 0.0001. 0.0001 is used because the simulator should do resampling before the variance is getting too big. the resampling method is implemented by the multinational resamplineg algorithm as presented in Chapter 3. A simple example is illustrated in Fig. 6 on page 15.

4.2 Simulator structure

The structure of the simulator needed to be changed after the introduction of the resampling method, as shown in Fig. 10. The structure of the previous simulator is very simple, consisting of a pipeline tracing the full life of all photon packets sequentially, one after the other. However, my new simulator structure requires more calculations and data transfer to and from the computer memory. It traces the first photon packet, stops at the first propagation step. After that, it traces next photon packet until all photon packets are finished their first propagation step. The simulator calculates the variance and normalized weights of all photon packet to compare with variance threshold. repeating those process for each propagation step until all photon packets are dead.

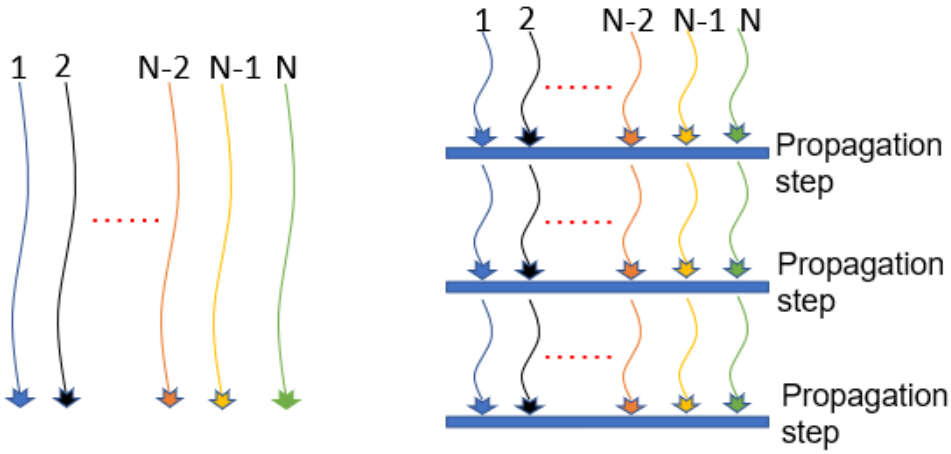

Figure 10: Left hand side represents the structure of previous simulator. It trace each photon packet from birth to death individually. The right hand side is the structure of my simulator. It has to stop at each propagation step for all photon packets. Then,calculate the normalized weights and do resampling process. Therefore, my simulator is doing more calculation than previous simulator.

5 Simulation results

I implemented my simulator on an overclocked Intel i7-6820HK CPU (4 Ghz) with DDR4 16GB RAM. My simulator was compared with the previous CPU-based simulator develop by Dr. Sherif's group.

5.1 OCT Simulation for ellipsoid and two sphere inside a slab

Simulations using $2 \cdot 10^6$ photon packets are compared in Fig. 11. The medium used in my simulation was the ellipsoid and two spheres inside a homogeneous background slab. The abstract view of this medium is shown in Fig. 8, and the mesh view is also shown in Fig. 12. Each object has its own optical properties as given in Table 1.

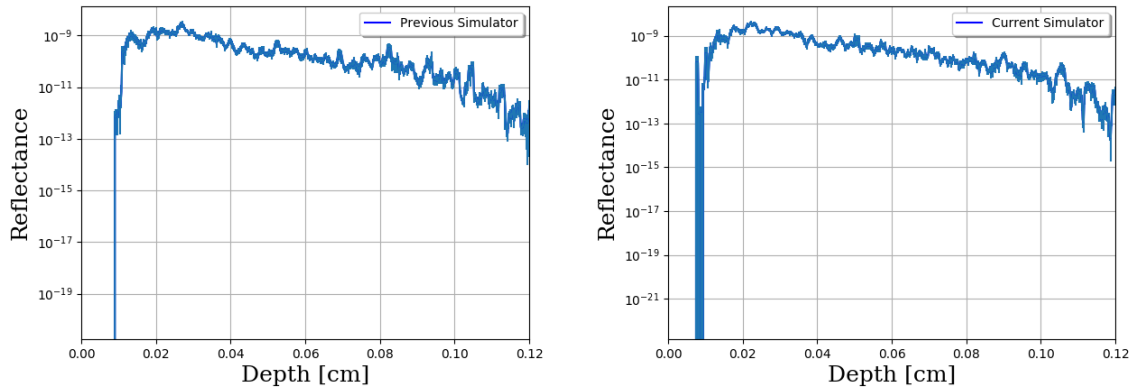

Figure 11: Left hand side is the axis image (A-scan) simulated by the previous simulator, and the right hand side is the result of my simulator. Comparing both A-scans, my simulator can generate the similar A-scan as the previous simulator with the resampling method is applied.

The simulation time is compared between the previous simulator and my simulator. The previous simulator spent 28 seconds to complete the calculation, while my simulator required 3808 second for the same scan. My simulator is presently about 140 times slower, then. The main reason is because the previous simulator was fully optimized, but my simulator has not yet been optimized. Also, the previous simulator has a simpler structure with fewer calculations compared with my simulator.

Medium	Optical Properties		
	u_s (cm^{-1})	u_a (cm^{-1})	g
Slab	1.5	60	0.9
Spheres	3	120	0.9
Ellipsoid	3	120	0.9

Table 1: The simulated mediums has different optical properties, such as absorption coefficient μ_s, scattering coefficient μ_s and the anisotropy factor g. They are given to each object in the medium at the beginning of the simulation.

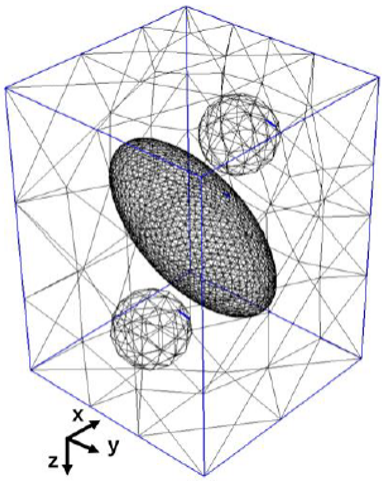

Figure 12: Tetrahedron mesh of the medium, The medium are constructed by total 5248 vertices and 29697 tetrahedrons meshs.

The A-scan comparison in Fig. 11 dose show that similar numerical values are been achieved with both simulators. While this is encouraging, it cannot yet be considered a test of accuracy between the two simulators. Accuracy comparison requires a huge number of photon packet to simulate a full OCT image. Given, that the previous simulator took 360 hours to generate a full image [14], my simulator would presently take far too long to make such a comparison worthwhile. As a result, a proper comparison of accuracy between simulators will only be meaningful once my simulator is fully optimized, so we can tell how much improvement we have done in my simulator.

5.2 Simulation time versus numbers of photon packet

I also tested my simulator with different number of photon packet. The simulated medium is same as section 5.1, an ellipsoid and two spheres inside a homogeneous background slab. The computational time of my simulator is given in Fig. 13. It shows that the computational time has a increasing growth as the number of photon packet increase. The reason for that could be the inefficiency of the multinomial resampling algorithm or further optimization is required for my simulator.

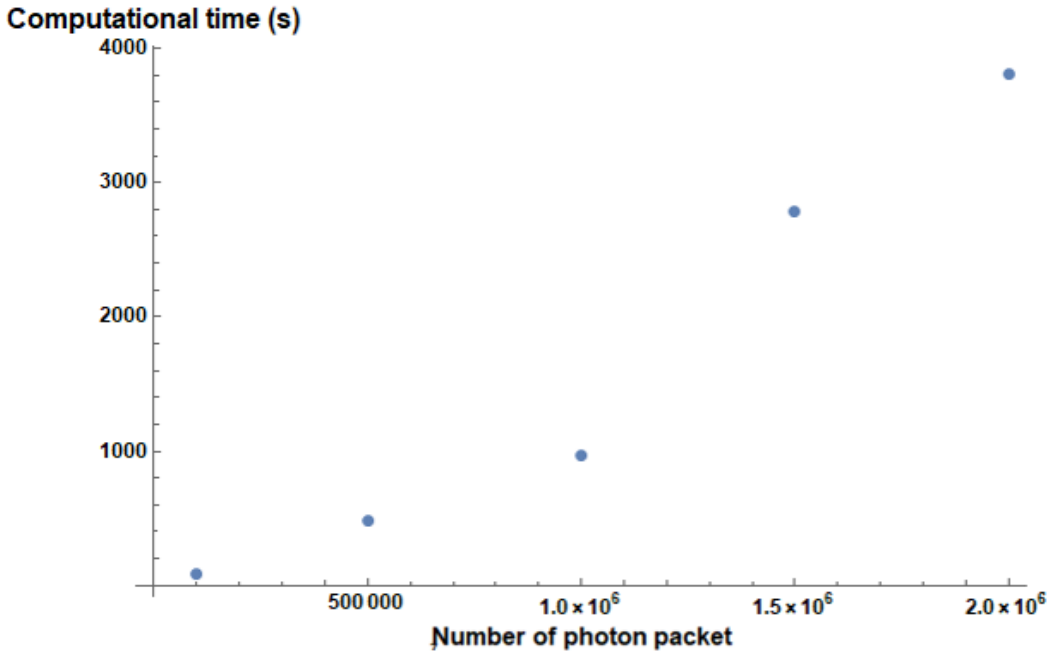

Figure 13: The number of photon packet is increasing from 10^5 to $2 \cdot 10^6$ evenly, but the computational time has a increasing growth. This is due to the multinomial resampling. As the number of photon packet increases, the multinomial resampling do more calculation to finish the selection process

6 Conclusion and future work

6.1 Conclusion

In this project, the goal was to try new methods in the OCT simulator for reduce the computational time and increase the accuracy of simulation. I have successfully applied the resampling method to my simulator with multinomial resampling algorithm. I also validate my simulator by comparing the A-scan of some mediums with previous simulator. In addition, I have found the computational time for different number of photon packet in the simulator.

6.2 Future work

To determine how much improvement I have done with the new method, further optimization is required. Furthermore, implementing the previous simulator on GPUs could reduce the simulation computation time by 100 times [3]. I should try my new simulator running on GPUs as well. The multinomial resampling algorithm is not the efficient way to implement the resampling method, I could try different algorithm such as systematic resampling and residual resampling, they are more efficient but also more complex.

References

[1] D. Huang, E. A. Swanson, C. P. Lin, J. S. Schuman, W. G. Stinson, W. Chang, M. R. Hee, T. Flotte, K. Gregory, C. A. Puliafito, and A. Et, "Optical coherence tomography", Science, vol. 254, no. 5035, pp. 11781181, Nov. 1991.

[2] Siavash Malektaji, "Monte Carlo simulation of optical coherence tomography of media with arbitrary spatial distributions", M.Sc. thesis, University of Manitoba, 2014.

[3] Mauricio Rodrigo Escobar Ivanauskas, "Massively parallel simulator of optical coherence tomography of inhomogeneous media", M.Sc. thesis, University of Manitoba, 2015.

[4] Micheal Sobhy, "Fast simulation of finite-beam optical coherence tomography of inhomogeneous turbid media", M.Sc. thesis, University of Manitoba, 2018.

[5] Micheal Sobhy, Ishan Wickramasingha, Ivan T. Lima JR., Sherif Sherif "Fast simulation of finite-beam optical coherence tomography of inhomogeneous turbid media using a sequential Monte Carlo solution of the Radiative Transfer Equation", unpublished paper (will be published on the biomimetica optics express)

[6] Saleh, B. E., Teich, M. C., Saleh, B. E. "Fundamentals of photonics", New York: Wiley, 1991.

[7] Lorenzo, J. R., "Principles of diffuse light propagation: light propagation in tissues with applications in biology and medicine", World Scientific, 2012.

[8] Carter, L. L., Cashwell, E. D. "Particle-transport simulation with the Monte Carlo method", Los Alamos Scientific Lab, 1975.

[9] Arnaud Doucet, Adam M. Johansen, "A Tutorial on Particle Filtering and Smoothing: Fifteen years later" , 2008.

[10] D. J. Smithies, T. Lindmo, Z. Chen, J. S. Nelson and T. E. Milner, "Signal attenuation and localization in optical coherence tomography studied by Monte Carlo simulation",Physics in Medicine and Biology, vol. 43, no. 10, pp. 3025-3044, 1998.

[11] G. Yao, L. Wang, "Monte Carlo simulation of an optical coherence tomography signal in homogeneous turbid media",Phys. Med. Biol , 44 (9) 1999.

[12] I.T. Lima, A. Kalra, S.S. Sherif, "Improved importance sampling for Monte Carlo simulation of time-domain optical coherence tomography",Biomed. Opt. Ex- press, 2 (5) (2011) 10691081.

[13] C. Zhu and Q. Liu, "Review of Monte Carlo modeling of light transport in tissues",Journal of biomedical optics, vol. 18, no. 5, pp. 050902-050902, 2013.

[14] S. Malektaji, I. T. Lima Jr. and S. S. Sherif, "Monte Carlo simulation of optical coherence tomography for turbid media with arbitrary spatial distributions",Journal of Biomedical Optics 94, vol. 19, no. 4, pp. 046001-046001, 2014.

[15] V. Periyasamy, M. Pramanik, "Monte Carlo simulation of light transport in turbid medium with embedded objectspherical, cylindrical, ellipsoidal, or cuboidal objects embedded within multilayered tissues",J. Biomed. Opt. 19 (4) 2014.

[16] M.Y. Kirillin, E. Alarousu, T. Fabritius, R.A. Myllyl, A.V. Priezzhev, "Visualization of paper structure by optical coherence tomography: Monte Carlo simulations and experimental study",J. Opt. Soc. Rapid 2 2007.

[17] James V. Candy, *Bayesian Signal Processing Classical, Modern, and Particle Filtering Methods*, 1st edition Wiley-IEEE Press (April 6 2009).

[18] Wang, L. V., Wu, H. I. "Biomedical optics: principles and imaging",John Wiley Sons. 2012.

[19] A. Smith, Arnaud Doucet, Nando de Freitas, Neil Gordon, *Sequential Monte Carlo Methods in Practice*, 2nd edition (Springer, 2010).

[20] Dunn, W. L., Shultis, J. K. "Exploring monte carlo methods",Elsevier. 2011.

[21] Tiancheng Li, Miodrag Bolic, Petar M. "Resampling Methods for Particle Filtering",IEEE signal processing magazine. 2015.

A Code

Some selected code in my simulator is presented below. they are important because first one is describing the main structure of the simulator, and second one is the idea to implement the resampling method.

A.1 Monte Carlo simulation of OCT signal

"DoOneRun" function is one Monte Carlo simulation for the OCT signal in medium. The function is called after the simulated medium is constructed. It is the main body of the simulator, the difficulty of writing this function is how to reorganize the simulation workflow with multinomial resampling algorithm. "DoOneRun" function shows a number of loop functions and data transfer operations that increase the complexity of simulator.

```
140  /*************************************************************
141   *  Execute Monte Carlo simulation for one independent run.
142   *************************************************************/
143  void DoOneRun(short NumRuns, SimulationStruct *In_Ptr,
  •  TetrahedronStruct *RootTetrahedron) {
144      register long i_photon;
  •      /* index to photon. register for speed.*/
145      OutStruct out_parm;
  •      /* distribution of photons.*/
146      TetrahedronPhotonStruct photon;
  •      /* The photon that is being traced */
147      TetrahedronPhotonStruct photon_cont;
  •      /* The secondary photon due to the splitting in biased
  •      scattering which will be trace */
148
149      long num_photons = In_Ptr->number_of_photons;
150
151  #if THINKCPROFILER
152      InitProfile(200,200); cecho2file("prof.rpt",0, stdout);
153  #endif
154
```

```
155       InitOutputData(*In_Ptr, &out_parm);
  •       /* initiating input and output structure */
156       /* The specular reflection at the beginning */
157       out_parm.Rsp = Rspecular(regionspecs[0].n,
  •       regionspecs[RootTetrahedron->region].n);
158
159       i_photon = num_photons;
160
161       PunchTime(0, "");
162
163       photon.FstBackReflectionFlag = 0;  //No backreflection
164
165       //Preparing for resampling
166       TetrahedronPhotonStruct photonparket[num_photons];
167       TetrahedronPhotonStruct photonparket_cont[num_photons];
168
169       TetrahedronPhotonStruct saved_photonparket[num_photons];
170       TetrahedronPhotonStruct
  •       saved_photonparket_cont[num_photons];
171
172       TetrahedronPhotonStruct
  •       resampled_photonparket[num_photons];
173       TetrahedronPhotonStruct
  •       resampled_photonparket_cont[num_photons];
174
175       double photonparket_weight[num_photons];
176       double photonparket_variance = 0;
177       long num_resampling = 0;
178
179       //tracing all photon packets
180       for (int i = 0; i <num_photons; i++ ){
181         NewPhoton(In_Ptr, out_parm.Rsp, &photonparket[i],
  •         &photonparket_cont[i], RootTetrahedron);
182       }
183
184       //stop for each propagation step
185       for(;;){
186         for (int i = 0; i <num_photons; i++ ){
187           if(!photonparket[i].dead){
188             HopDropSpin(In_Ptr, &photonparket[i],
  •             &photonparket_cont[i], &out_parm);
189           }
190         }
```

```
192         for(int i =0; i < num_photons; i++){
193           photonparket_weight[i] = photonparket[i].w;
194         }
195         photonparket_variance = var(photonparket_weight,
  •         num_photons);
196         printf("The variance is : %lf
  •         \n\n",photonparket_variance);
197
198         //do resampling if the variance is greater than the
  •         variance threshold
199         if(photonparket_variance >= variance_threshold){
200           printf("The multinomial resampling is
  •           processing!\n\n");
201           long saved_num = 0;
202
203           for (int i = 0; i <num_photons; i++ ){
204             if(!photonparket[i].dead){
205               saved_photonparket[saved_num] = photonparket[i];
206               saved_photonparket_cont[saved_num] =
  •               photonparket_cont[i];
207               saved_num++;
208             }
209           }
210           double saved_photonparket_weight[saved_num];
211           for(int i =0; i < saved_num; i++){
212             saved_photonparket_weight[i] =
  •             saved_photonparket[i].w;
213           }
214           //resampling according to the normalized weight
215           Weight_normalization(saved_photonparket_weight,saved_nu
  •           m);
216           Multinomial_resampling(saved_photonparket,
  •           resampled_photonparket,
  •           saved_photonparket_cont,resampled_photonparket_cont,
  •           saved_photonparket_weight, saved_num);
217           printf("saved_num :%ld and num_resampling:
  •           %ld\n",saved_num,num_resampling );
218           num_resampling++;
219           //replace samples by new samples after resampling
220           for(int i = 0; i < saved_num ;i++){
221             photonparket[i]=resampled_photonparket[i];
222             photonparket_cont[i]=resampled_photonparket_cont[i];
223             photonparket[i].w = 1;                //all weights
```

```
224           }
225           num_photons = saved_num;
226           for(int i =0; i < saved_num; i++){
227             saved_photonparket_weight[i] = photonparket[i].w;
228           }
229         }
230         //loop termination
231         int stop_on = 1;
232         for(int i =0; i < num_photons; i++){
233           if(!photonparket[i].dead){
234             stop_on = 0;
235           }
236         }
237         if(stop_on){
238           break;
239           }
240       }
241       //record the result
242       WriteResult(In_Ptr, &out_parm);
243       printf("The data file is saved\n\n");
244
```

A.2 Main components of multinomial resampling algorithm

Some important components of multinomial resampling such as calculating mean, variance, and normalization of weight of samples. The most important function is the "multinomial_resampling", it uses those calculated value form those functions implement resampling method.

```
29  //set the variance_threshold
30  #define variance_threshold 0.0001
31
32  //calculate the weight mean
33  double mean(double values[], long n){
34      double sum = 0;
35      for (int i = 0; i < n; i++)
36      {
37          sum += values[i];
38      }
39      return sum / n;
40  }
```

```
42  //calculate the weight variance
43  double var(double values[], long n){
44      double valuesMean = mean(values, n);
45      double sum = 0;
46      for (int i = 0; i < n; i++)
47      {
48          sum += (values[i] - valuesMean) * (values[i] -
 •          valuesMean);
49      }
50      return sum / (n-1);
51  }
52
53  //normalize weight
54  void Weight_normalization(double *weight, long total_num){
55    double tot_wight = 0;
56    for(int i = 0; i < total_num; i++){
57      tot_wight+=weight[i];
58    }
59    for(int i = 0; i < total_num; i++){
60      weight[i]=weight[i]/tot_wight;
61    }
62  }
63
```

```
64  //Multinomial_resampling
65  void Multinomial_resampling(TetrahedronPhotonStruct
 •  *saved_photon, TetrahedronPhotonStruct *resampled_photon,
 •  TetrahedronPhotonStruct *saved_photon_cont,
 •  TetrahedronPhotonStruct *resampled_photon_cont, double *weight
 •  ,long total_num){
66    for(int i = 0; i < total_num; i++){
67      double random_num = ((double)rand())/RAND_MAX;
68      double gap = 0;
69      for(int j=0; j < total_num; j++){
70        if(gap < random_num & random_num <= gap + weight[j]){
71          resampled_photon[i] = saved_photon[j];
72          resampled_photon_cont[i] = saved_photon_cont[j];
73          break;
74        }
75        gap += weight[j];
76      }
77    }
78  }
```