Advanced Monte Carlo Simulation and Machine Learning for Frequency Domain Optical Coherence Tomography

Thesis by

Sinan Zhao

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

2016

(Defended July 10, 2015)

To my wife Shuang, whose love made this possible.

Acknowledgements

My time at Caltech has been a transforming experience, at all levels. I'm blessed with the fortune to meet and interact with a number of great people, who I would like to take the opportunity to acknowledge here. Caltech is such a wonderful and resourceful place that, even if you only have a vague idea about what you want to achieve, it is able to help you crystallize your thoughts and materialize your dreams.

First and foremost, my sincere thanks are due to my advisor Prof. Amnon Yariv, who admitted me to Caltech, introduced me to the field of optoelectronics, and allowed me to pursue freely my interests in the field of machine learning, applied mathematics, and computer science. Without his support and encouragement, this thesis would not have been possible. His guidance, challenging questions, and suggestions has greatly accelerated my development as a researcher and allowed me to flourish both professionally and intellectually.

Not only is Prof. Yariv a gifted scientist, he is also a such versatile man that all of us wish to emulate: he is capable of speaking seven languages and knows just about everything of history and art (he even gave us a lecture about classical music on an applied physics class), not to mention that he has been an expert in surfing and has traveled all around the globe, including the Antarctic. His unique and well-rounded experience has inspired us from the highest level and motivated us to always take a step further.

My special thanks are due to Prof. Yaser Abu-Mostafa, who has made great influences on me during my journey at Caltech. Meeting Yaser was an absolute turning point of my life and career. Not only did he introduce me to the exciting field of machine learning, but he also helped me apply it to solving real world problems.

His on-line course *Learning From Data* is so intriguing that you cannot help watching it time and time again. The quality of the course is certainly way off the chart, but at the same time it just feels great to hear that man talking. The half year span where I could meet Yaser several times a week working on an industrial project definitely ranks high among my best times at Caltech. Yaser is the ultimate guru in solving real-world machine learning problems. His level of expertise and wisdom makes working with him the most enjoyable and rewarding experience you could ever ask for, and also his attention to technical details renders him the ideal boss. I will never forget the time when I had an idea in the middle of the night for the project, and discovered that Yaser was still at his office, ready for an immediate discussion with me.

I thank Profs. Bruno Crosignani, P. P. Vaidyanathan, Willie Ng for being part of my thesis committee, as well as Prof. Mark Simons who was part of my candidacy committee. I enjoyed working with Prof. Bruno Crosignani on solving interesting physics problems, along with the fun discussions regarding how to help me improve my taste of wine. I thank Prof. Willie Ng for collaborating with us on the iPhod project while he was at HRL Laboratories.

I would like to thank the Caltech departments and professors who employed me as a Teaching Assistant. Without their help it won't be possible for me to survive these years. I'm especially grateful for Prof. Houman Owhadi, who has employed me three times as his TA. I even gave a graduate-level lecture at Caltech on his behalf while he was out of town, and I still remember my nervousness on that occasion. I also thank the head TAs of ACM95/100, Gerardo Cruz and Adam Neumann, and Maria Lopez at Computing & Mathematical Sciences, for constantly looking after me in case I need a TAship each and every term.

My thanks are also due to the Yariv group members, both old and new - Xiankai Sun, Naresh Satyan, Hsi-Chun Liu, Christos Santis, Arseny Vasilyev, Jacob Sendowski, Mark Harfouche, Dongwan Kim, Marilena Dimotsantou, Paula Popescu, Reg Lee, and George Rakuljic - for their help and friendship, for all of the mind-blowing discussions and most importantly, for making the department a fun place

to work in. I could not appreciate more about the diversity and intelligence of this group, which has inspired me in various aspects of my life during my time at Caltech. I sincerely wish the best of luck for everyone in their future endeavors.

I feel extremely lucky to meet lots of unbelievably bright friends at Caltech. I sincerely thank Hongchao Zhou, Tong Chen, Liling Gu, Jinghao Huang, Daiqi Linghu, Zhao Liu, Dongyang Kang, Xin Ning, Zhenhua Liu, Jiang Li, Maolin Ci, Dunzhu Li, Chenguang Ji, Kuang Shen, as well as the CaltechC 2010 members, for all the fun we had together and the memories we shared, they are so priceless and will stick in my mind for the years to come.

I would like to thank my parents, who have always been there for me throughout my years abroad. I will be forever grateful for their constant love and unconditional support.

Lastly, I would like to express my deepest thanks and gratitude to my lovely wife, Shuang (Vivian) Wu. This thesis would not have been possible without her encouragement, sacrifice, and love. She is such a fun character and at the same time extremely talented. There are an endless number of things that I could learn from her, from Japanese to classical music, from singing to playing piano, not to mention that she also manages to beat me in heads-up poker from time to time, which is completely unconceivable. My life would never be out of joy with her company. Meeting her in Pasadena and living with her at Caltech has been the most wonderful thing ever happened to my life.

Abstract

Optical Coherence Tomography(OCT) is a popular, rapidly growing imaging technique with an increasing number of biomedical applications due to its noninvasive nature. However, there are three major challenges in understanding and improving an OCT system:

- Obtaining an OCT image is not easy. It either takes a real medical experiment or requires days of computer simulation. Without much data, it is difficult to study the physical processes underlying OCT imaging of different objects simply because there aren't many imaged objects.

- Interpretation of an OCT image is also hard. This challenge is more profound than it appears. For instance, it would require a trained expert to tell from an OCT image of human skin whether there is a lesion or not. This is expensive in its own right, but even the expert cannot be sure about the exact size of the lesion or the width of the various skin layers. The take-away message is that analyzing an OCT image even from a high level would usually require a trained expert, and pixel-level interpretation is simply unrealistic. The reason is simple: we have OCT images but not their underlying ground-truth structure, so there is nothing to learn from.

- The imaging depth of OCT is very limited (millimeter or sub-millimeter on human tissues). While OCT utilizes infrared light for illumination to stay non-invasive, the downside of this is that photons at such long wavelengths can only penetrate a limited depth into the tissue before getting back-scattered. To image a particular region of a tissue, photons first need to reach that re-

gion. As a result, OCT signals from deeper regions of the tissue are both weak (since few photons reached there) and distorted (due to multiple scatterings of the contributing photons). This fact alone makes OCT images very hard to interpret.

This thesis addresses the above challenges by successfully developing an advanced Monte Carlo simulation platform which is $10,000$ times faster than the state-of-the-art simulator in the literature, bringing down the simulation time from 360 hours to a single minute. This powerful simulation tool not only enables us to efficiently generate as many OCT images of objects with arbitrary structure and shape as we want on a common desktop computer, but it also provides us the underlying ground-truth of the simulated images at the same time because we dictate them at the beginning of the simulation. This is one of the key contributions of this thesis. What allows us to build such a powerful simulation tool includes a thorough understanding of the signal formation process, clever implementation of the importance sampling/photon splitting procedure, efficient use of a voxel-based mesh system in determining photon-mesh interception, and a parallel computation of different A-scans that consist a full OCT image, among other programing and mathematical tricks, which will be explained in detail later in the thesis.

Next we aim at the inverse problem: given an OCT image, predict/reconstruct its ground-truth structure on a pixel level. By solving this problem we would be able to interpret an OCT image completely and precisely without the help from a trained expert. It turns out that we can do much better. For simple structures we are able to reconstruct the ground-truth of an OCT image more than 98% correctly, and for more complicated structures (e.g., a multi-layered brain structure) we are looking at 93%. We achieved this through extensive uses of *Machine Learning*. The success of the Monte Carlo simulation already puts us in a great position by providing us with a great deal of data (effectively unlimited), in the form of $(image, truth)$ pairs. Through a transformation of the high-dimensional response variable, we convert the learning task into a multi-output multi-class classification problem and a multi-output regression problem. We then build a hierarchy architecture of machine learning models

(committee of experts) and train different parts of the architecture with specifically designed data sets. In prediction, an unseen OCT image first goes through a classification model to determine its structure (e.g., the number and the types of layers present in the image); then the image is handed to a regression model that is trained specifically for that particular structure to predict the length of the different layers and by doing so reconstruct the ground-truth of the image. We also demonstrate that ideas from *Deep Learning* can be useful to further improve the performance.

It is worth pointing out that solving the inverse problem automatically improves the imaging depth, since previously the lower half of an OCT image (i.e., greater depth) can be hardly seen but now becomes fully resolved. Interestingly, although OCT signals consisting the lower half of the image are weak, messy, and uninterpretable to human eyes, they still carry enough information which when fed into a well-trained machine learning model spits out precisely the true structure of the object being imaged. This is just another case where Artificial Intelligence (AI) outperforms human. To the best knowledge of the author, this thesis is not only a success but also the first attempt to reconstruct an OCT image at a pixel level. To even give a try on this kind of task, it would require fully annotated OCT images and a lot of them (hundreds or even thousands). This is clearly impossible without a powerful simulation tool like the one developed in this thesis.

Contents

List of Figures

List of Tables

Listings

Glossary of Acronyms

OCT Optical Coherence Tomography

TD-OCT Time Domain Optical Coherence Tomography

FD-OCT Frequency Domain Optical Coherence Tomography

MC Monte Carlo

ML Machine Learning

DT Decision Trees

FFT Fast Fourier Transform

Chapter 1

Overview

1.1 Introduction

Optical Coherence Tomography (OCT) is rapidly becoming an important imaging technique for numerous medical and biological applications [3, 4]. It is a sub-surface imaging technique that uses either a low-coherence light source (time-domain systems) or a wavelength-swept laser source (frequency-domain systems). It can provide real-time imaging and with a depth resolution of 1μm or less [5, 6], which is two orders of magnitude higher resolution than ultrasound imaging. The penetration depth, which is highly tissue-dependent, can reach up to 3 mm and is typically limited to a few millimeters. OCT is also able to produce images inside the body when integrated with optical fiber probes. The use of infrared and visible light is safer to most biological samples than ionizing radiation like X-rays or gamma rays, and it also allows for spectroscopic characterization of an object, e.g., a tumor in tissue [7]. A recent thorough review of OCT technology, including frequency-domain OCT, and applications of OCT can be found in the *Handbook of Non-invasive Methods and the Skin* [8].

1.2 Basic OCT Principle

OCT is an interferometric technique, relying on interference between a split and later re-combined broadband optical field. The principle of OCT is shown in Figure 1.1. A Michelson interferometer (Figure 1.1A) can be used to measure the ability of

light to interfere with itself, i.e., the ability to amplify or blur itself ("constructive" and "destructive" interference, respectively). Light is split into two paths using a beam-splitter (half-transparent mirror). The two beams recombine at the beam-splitter, and detected. Interference between the two reflections is possible only when the path-lengths of the two arms are matched within the so-called coherence length of the light source. The coherence length is determined by the spectral width of the light—a broad optical spectrum corresponds to a short coherence length, and a narrow optical spectrum corresponds to a long coherence length. When using a light source with a large coherence length, interference arises for even very large differences in path-length. When using a source with small coherence length, interference only arises when the two path-lengths are matched within the coherence length of the light, which may be micrometer size. It is exactly this effect that is used in OCT for distinguishing signals from different depths of the sample. The axial resolution is essentially the coherence length, so that a small coherence length corresponds to high axial resolution.

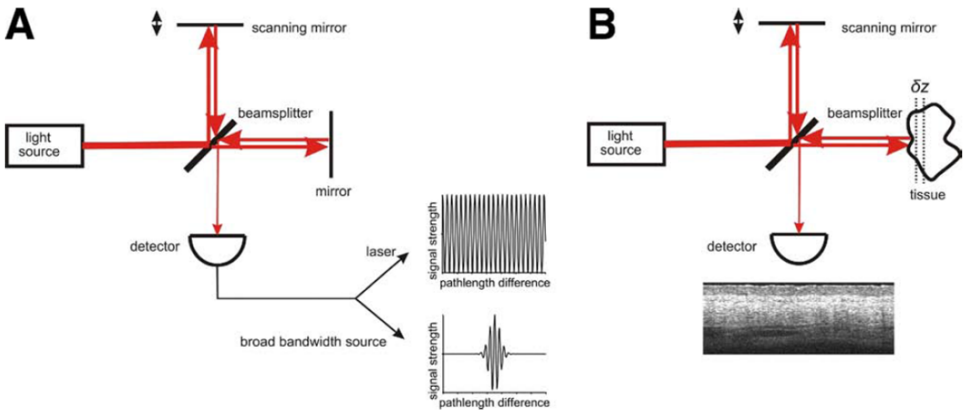


Figure 1.1. (A) A Michelson interferometer. (B) A Michelson interferometer with the fixed mirror replaced by a sample. An OCT image (B-scan) of the sample is shown below the detector.

If one of the mirrors in the Michelson interferometer is replaced by a biological sample as shown in Figure 1.1B, every position of the scanning mirror corresponds

to the collected signal from a thin slice in the sample. In other words, it becomes possible to determine the location of reflection. The thickness δz of the slice that contributes to the signal (Figure 1.1B) is equal to the depth resolution of the system and is inversely proportional to the bandwidth of the light source. The mechanism for selecting signal from a specific depth is also referred to as coherence gating. By moving the scanning mirror, the coherence gate successively selects an interference signal from different depths. In this way, a depth scan recording can be obtained, also referred to as an A-scan. The depth scanning range is limited by the mirror displacement. Transverse resolution is determined by the spot size, which is given by the focusing optics.

Two-dimensional data are obtained by moving the beam across the sample and acquiring data (B-scan). By translating the beam in 2 directions over a surface area, 3-dimensional data can be acquired (C-scan). Acquiring 2-and 3-dimensional data is in general possible in real-time. The interference signal is amplified, filtered to improve the signal-to-noise ratio, and then digitized and transferred to a computer. From the digital signal, the reflection strength is extracted and mapped, using either a gray scale or color palette, thereby generating an OCT image [9].

1.3 Present Challenges in OCT

In Figure 1.2 we illustrate a typical clinical trial of OCT [10], where a person with a nodular basal cell carcinoma (BCC) lesion, a type of non-melanoma skin cancer, is examined by an OCT imaging system. From this example we can see that even though the OCT image does a good job in terms of qualitatively describing what is going on under the skin, it is far from presenting us the anatomical structure (i.e., the ground-truth), which is the ultimate goal of any imaging. While the OCT image indeed shows us that there is BCC lesion, we can hardly conclude anything more than that. For example, we cannot tell exactly the size and position of the lesion, nor the interfaces of the various skin layers (epidermis, upper dermis, etc.). What's worse is that the OCT image becomes increasingly darker and blurred as we go deeper into the

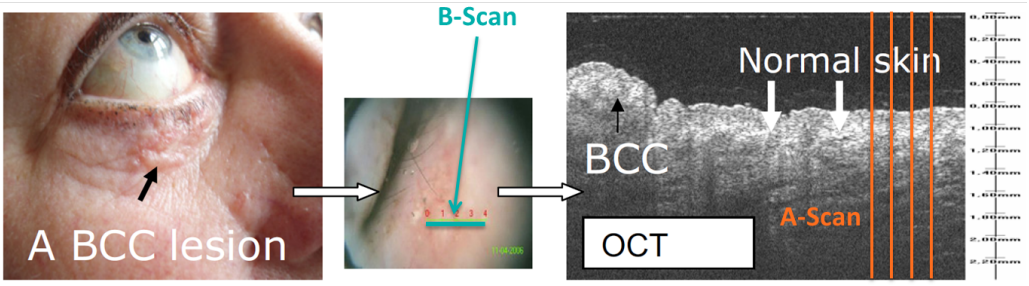


Figure 1.2. An example of a nodular BCC lesion that was easily delineated laterally with OCT is shown. The black arrow points at the lesion in the clinical photo and at the same lesion in the OCT image. In between, the image from the OCT probe is seen with a green line indicating where the OCT scan was performed. White arrows indicate the adjacent normal skin in the OCT image.

skin, where the lower half of the OCT image is completely black so there is nothing really for us to see.

As hinted above, despite the advantages and developments of OCT in recent years, there are three major challenges in understanding and improving an OCT system:

- *Obtaining an OCT image is not easy.* It either takes a real medical experiment, like a clinical trial, or requires days of computer simulation [11]. Without much data, it is difficult to study the physical processes underlying OCT imaging of different objects simply because there aren't many imaged objects. For specular reflective objects, e.g., a mirror, the corresponding OCT signal can be derived analytically, because the photons go through a deterministic process. However, once you replace the mirror with a real biological tissue, it is a fundamentally different problem as the physical process now becomes *light propagation through random media*. As the name suggests, to obtain an OCT image of a biological tissue, one has to model how light (photons) interacts with the tissue. In other words, we have to determine how each individual photon gets reflected, refracted, scattered, and absorbed by the tissue, as well as whether the photon has escaped or been detected. Therefore, to obtain an OCT image of a realis-

tic biological tissue, we have to either conduct a real experiment or perform a computer simulation. The gold standard for the latter endeavor is Monte Carlo simulation.

- *Interpretation of an OCT image is also hard.* This challenge is more profound than it appears. For instance, it would require a trained expert to tell from an OCT image of human skin, like the one shown in Figure 1.2, whether there is a lesion or not. This is expensive in its own right, since you have to bring in that expert (a doctor for example), but even the expert cannot be sure about the exact size of the lesion or the widths of the various skin layers. Ideally, we would like to recover the anatomical structure of the object being imaged, which is the goal of any imaging. We shall not blame the expert because the task itself is beyond human capabilities. This is because the physical process dictates that the photons contributing to the OCT image inevitably go through multiple scattering events and therefore the image itself is distorted, except for the surface or close to the surface region of the image. Moreover, the problem is not only hard but also *unlearnable*. What I mean by this is that for most of the OCT images we have, we do not know their corresponding anatomical or ground-truth structure. Therefore we couldn't even try to learn how to infer the ground-truth from its OCT image because there are no such examples. The take-away message is that analyzing an OCT image even from a high level would usually require a trained expert, and pixel-level interpretation is simply unrealistic. The reason is simple: we have OCT images but not their underlying ground truth, so there is nothing to learn from.

- *The imaging depth of OCT is very limited.* While OCT utilizes infrared light for illumination in order to stay noninvasive, the downside of this is that photons at such long wavelengths can only penetrate a limited depth into the tissue (millimeter or sub-millimeter on human tissues) before getting back-scattered. This is simply a fact determined by the optical properties of the tissue as well as the physical laws of light propagation. To image a particular region of a tissue,

photons first need to reach that region, and then get back-scattered from that region. The deeper the region, the fewer photons could reach and get scattered from that region. As a result, OCT signals from deeper regions of the tissue, e.g., the lower half of Figure 1.2, are both weak (since few photons reached it) and distorted (due to multiple scatterings of the contributing photons). This fact alone makes OCT images, especially the part of deeper regions, very hard to interpret. As a result, the usefulness of an OCT image is limited to its surface regions unless we can interpret the much weaker signals from relatively deeper regions.

In summary, obtaining an OCT image is not inexpensive, which would take either an experimental trial or through expensive computer simulations. Interpreting an OCT image is also hard, i.e., we cannot recover the anatomical structure of the imaged object. This is due to the fact that the physics underlying the image formation process distorts and weakens the signal, as well as the lack of fully annotated OCT images for us to learn from. Moreover, because of the multiple scattering events that the photons need to go through, deeper regions of a biological tissue (beyond a few millimeters) are beyond the reach of all but a few photons, resulting in a limited imaging depth of OCT.

1.4 Summary of Our Contributions

This thesis addresses the challenges described in the previous section by successfully developing an advanced Monte Carlo simulation platform which is $10,000$ times faster than the state-of-the-art, bringing down the simulation time of an OCT image from 360 hours to a single minute. This powerful simulation tool not only enables us to efficiently generate an arbitrarily high number of OCT images of objects with arbitrary structure and shape on a common desktop computer, but it also provides us the underlying ground-truth of the simulated images at the same time because we dictate them at the beginning of the simulation. This is one of the key contributions of this thesis. And what allowed us to build such a powerful simulation tool includes

a thorough understanding of the signal formation process, particularly for frequency domain OCT, a clever implementation of the importance sampling/photon splitting procedure, efficient use of a voxel-based mesh system in determining photon-mesh interception, and a parallel computation of different A-scans that consist a full OCT image, among other programing and mathematical tools, which will be explained in detail in later chapters of this thesis.

Next we aim at solving the inverse problem, which is, given an OCT image, predict/reconstruct its ground-truth structure on a pixel level. By solving this problem we would be able to interpret an OCT image completely and precisely without a trained expert. It turns out that we can do much better. For simple structures we are able to reconstruct the ground-truth of an OCT image more than 98% correctly, and for more complicated structures (e.g., a multi-layered brain structure) we are looking at 93%. In other words, we managed to beat the hypothetical expert by a large margin. We achieved this through extensive uses of *Machine Learning*. The success of the Monte Carlo simulation already puts us in a great position by providing us with a great deal of data (effectively unlimited), in the form of $(image, truth)$ pairs. These are the training examples for us or a machine learning algorithm to learn from. The machine learning problem is not easy though, since given an OCT image, say with 512×512 pixels, we need to predict the same amount (which is 512×512) of pixels in order to fully reconstruct the ground truth. In a typical computer vision problem, we are only required to predict a single, high-level output, say a classification label given an image. So it seems that our problem is orders of magnitude harder. The good news is that we have effectively unlimited data to learn from, plus the data will be fully annotated on the pixel level, and more importantly we have full control in designing and generating the data set which we think will help to better train machine learning models. The fast Monte Carlo simulation platform developed in this thesis makes such a scheme possible, since previously simulating a single OCT images would take days or even weeks.

The learning and prediction are done at the A-scan level. In our setup, each A-scan consists of 1910 points (pixels). In other words, the machine learning model

would take a 1910-dimensional input, and need to produce a 1910-dimensional output. The first thing that comes into mind is dimension reduction, i.e., it is necessary to transform of the high-dimensional response variable into a smaller dimension, so that the problem is manageable. By recognizing that each realistic biological tissue displays a layered structure at the A-scan level, we convert the learning task into a multi-output multi-class classification problem and a multi-output regression problem. The goal of the multi-output multi-class classification problem is to predict the order and the types of the layers comprising the A-scan, while the goal multi-output regression problem is simply to predict the depth of each layers. This is another key step towards solving the inverse problem, since it reduces the dimension of the output from 1910 to less than 40. We then build a hierarchy architecture of machine learning models (committee of experts) and train different parts of the architecture with specifically designed data sets. Again, we are able to do this by virtue of the fast Monte Carlo platform we developed. In prediction, an unseen OCT image is decomposed into A-scans, and each A-scan first goes through a classification model to determine its structure (e.g., the number and the types of layers present in the A-scan), and then it is handed to a regression model that is trained specifically for that particular structure to predict the length of the different layers and by doing so reconstruct the ground-truth of the A-scan. In the end we conduct a pooling step by averaging the prediction of adjacent A-scans before constructing our final answer for the ground-truth. This pooling step is based on the prior knowledge that there is a local continuity in the structure of a natural biological tissue, and it turns out that this gives us another boost in the prediction performance. We also demonstrate in the end of the thesis that ideas from *Deep Learning* can be useful to further improve the system.

It is worth pointing out that solving the inverse problem automatically improves the imaging depth, since previously the lower half of an OCT image (i.e., greater depth) can be hardly seen (as shown in Figure 1.2) but now can be fully resolved. Interestingly, although OCT signals coming from the lower half of the image are weak, messy, and uninterpretable to human eyes, they nevertheless carry enough

information which, when fed into a well-trained machine learning model, spits out precisely the true structure of the object being imaged. This is just another case where Artificial Intelligence (AI) outperforms humans. Clearly, with the development of computational power as well as advances in machine learning and AI, OCT systems will continue to improve. It is not hard to imagine that in the near future an OCT imaging system would present us with the anatomical structure outright and possibly beyond a few millimeters. Hopefully this thesis represents an important first step towards this goal. To the best knowledge of the author, this thesis is not only a success but also the first attempt to reconstruct an OCT image at a pixel level. To even give a try on this kind of task, it would require fully annotated OCT images and a lot of them (hundreds or even thousands), and this is clearly impossible without a powerful simulation tool like the one developed in this thesis.

1.5 Organization of the Thesis

This thesis is organized as follows. In Chapter 2 we first explain the signal formation process of Frequency Domain OCT (FD-OCT), starting from single, mirror-like targets. We will show how A-scans and OCT images can be constructed using the OCT signal. Next we move on to discuss how the situation is changed when we try to image a layered biological tissue. Chapter 2 concludes by stating the reasons why we need simulation to construct OCT images of real biological tissues.

Chapter 3 is intended to be a review of the standard Monte Carlo simulation procedure. By the end of the last chapter we already understand why we need a Monte Carlo simulation in oder to generate OCT images. In this chapter we begin with a general discussion of the principles of Monte Carlo simulation of light propagation in biological tissues, the big picture, important concepts and building blocks, as well as how can we use these to construct a platform to simulate OCT images. We then dig deeper into the issues behind conventional Monte Carlo simulation to gain some insight as to why it took days to simulate a single OCT image previously. A thorough understanding of these issues is a prerequisite to come up with ways to speed up the

simulation, which will be the topic of Chapter 4.

In Chapter 4 we will show in detail how we managed to bring down the simulation time of an OCT image from 360 hours to a single minute. We begin by identifying three main challenges that prevent us from fast Monte Carlo simulation of OCT images and then address them accordingly. To make this chapter less technical, most of the source code listings are left to the appendix which the interested reader is welcome to visit in order to fully appreciate the simulation procedure.

Starting from Chapter 5 of this thesis we will focus on the inverse problem: given an OCT image, predict its ground-truth structure on a pixel level. Solving this inverse problem would lead us to a completely new philosophy of medical imaging by handing the doctor and the patient precisely the anatomical structure, i.e., what you see is what you will get. No one has ever attempted it because there is simply not enough annotated data in the OCT world, but our advanced Monte Carlo platform has completely solved this issue. It is always a good idea to start by examining simple structures in order to better appreciate harder, more complex problems. This is the theme of Chapter 5, which is a warm-up towards solving the learning problem.

In Chapter 6 we will solve the learning problem introduced in the previous chapter, using all sorts of machine learning techniques. We begin with some high-level discussion of the challenges for the problem as well as how we may solve it. The first step is to transform the output variable, y_i, to reduce its dimension. Through this clever transformation, we arrive at a multi-output multi-class classification problem and a multi-output regression problem. As we explore the data further, we pick up more and more insights as to how to do the job better. Our advanced Monte Carlo platform also comes in handy, as it allows us to essentially generate data at will so that we can intentionally design different data sets to train the machine learning models for better out-of-sample performance. We then build a hierarchy architecture of machine learning models (committee of experts) based on extremely randomized trees (extra trees), and train different parts of the architecture with specifically designed data sets. In prediction, an unseen OCT image first goes through a classification model to determine its structure (e.g., the number and the types of layers present in

the image); then the image is handed to a regression model that is trained specifically for that particular structure to predict the length of the different layers and by doing so reconstruct the ground-truth of the image. We will also demonstrate in the next chapter that ideas from *Deep Learning* can be useful to further improve the performance.

In Chapter 7 we will first repeat what we have done in Chapter 6 to see if our hierarchical model is able to generalize to more complex, realistic problems. We will use the brain structure as our example, which is much more complicated than the air-dermis structure we have previously conquered. We then move on to summarize this thesis and discuss the new philosophy made possible by this thesis when it comes to OCT imaging. We conclude this thesis by taking a tour of future research opportunities in the outlook section.

Original results of this thesis are presented in Chapters 4, 6, and 7.

Chapter 2

Understanding the OCT Signal

2.1 Chapter Overview

In this chapter we are trying to accomplish a number of things. First, we will explain the signal formation process of Frequency Domain OCT (FD-OCT), starting from single, mirror-like targets. We will show how A-scans and OCT images can be constructed using the OCT signal. Next we move on to discuss how the situation is changed when we try to image a layered biological tissue. We conclude this chapter by stating the reasons of why we need simulation to construct OCT images of real biological tissues.

2.2 Signal Formation in FD-OCT Systems

The OCT principle described in Chapter 1 is essentially the time domain OCT (TD-OCT) system, where a reference mirror is scanned to match the optical path from reflections within the sample. We move the scanning mirror in *time* to image different depth of the object, thus the name. In contrast, FD-OCT has the advantage that no moving parts are required to obtain axial scans. The reference path length is fixed and the detection system is replaced with a spectrometer, as shown in Figure 2.1. The detected intensity spectrum is then Fourier transformed into the time domain to reconstruct the depth resolved sample optical structure. The essence of FD-OCT is to translate targets at different locations in the sample into distinct frequency

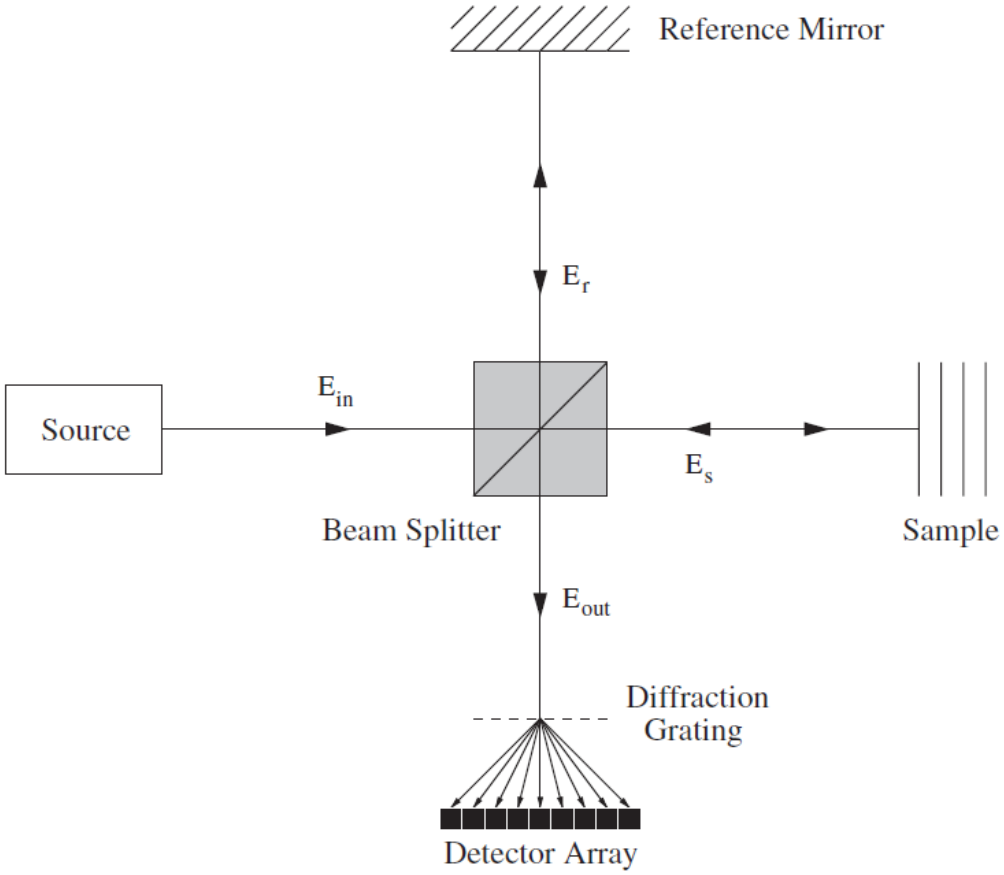


Figure 2.1. A FD-OCT system. The output light field is split by a diffraction grating, and component frequencies are detected by a linear detector array.

components in the output signal, $\mathbf{E}_{out}$.

To illustrate this, let us examine the problem of detecting the depth information of a single-scatterer using a linearly chirped, swept frequency laser. For simplicity, we consider a noiseless laser whose frequency changes linearly with time. A single scatterer is illuminated with such a chirped field, and the reflected light is collected. The normalized electric field at the source is given by:

$$E(t) = \cos\left(\phi_0 + \omega_0 t + \frac{\xi t^2}{2}\right), \tag{2.1}$$

where ξ is the slope of the optical chirp, and ϕ_0 and ω_0 are the initial phase and frequency, respectively. The instantaneous optical frequency is given by the time

derivative of the argument of the cosine in Equation 2.1:

$$\omega_{source}(t) = \frac{d}{dt}\left(\phi_0 + \omega_0 t + \frac{\xi t^2}{2}\right) = \omega_0 + \xi t. \tag{2.2}$$

The time evolution of the frequencies of the launched and reflected beams is shown in Figure 2.2. Because the chirp is precisely linear, a scatterer with a round-trip time delay τ (and a corresponding displacement $c\tau/2$ from the source) results in constant frequency difference $\xi\tau$ between the launched and reflected waves. Therefore, detecting this frequency difference would give us the depth information of the scatterer. In other words, the job of the FD-OCT system is to translate scatterers at different depths into different frequency components.

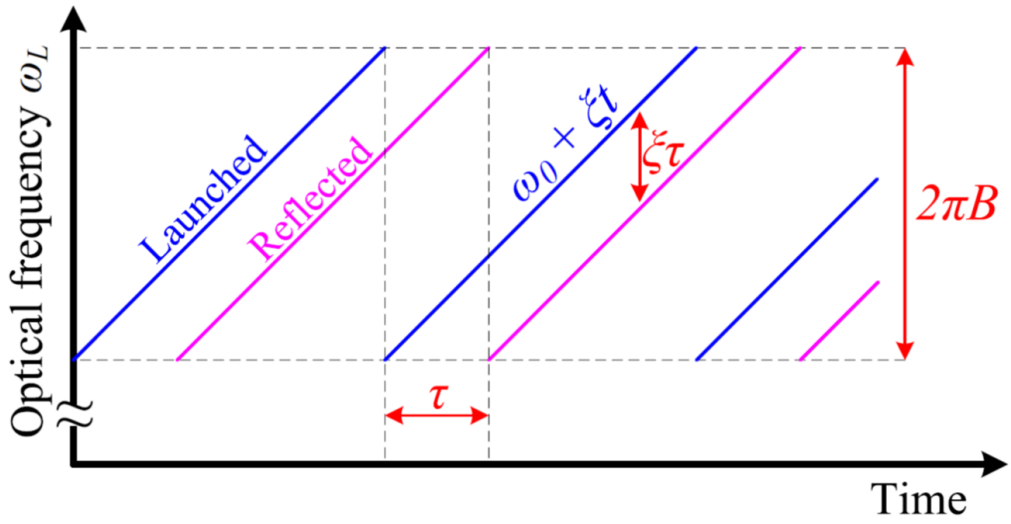


Figure 2.2. Time evolution of the optical frequencies of the launched and reflected waves in a single-scatterer OCT experiment [12].

2.2.1 A Toy Example

The best way to illustrate the working principles of FD-OCT is through a toy example, as shown in Figure 2.3. In this simple setup the sample is consisted of only one mirror like, specular reflective target, located at depth z_0. We also have an array of light source/detector pairs, located at $z = 0$, to represent conducting different A-scans.

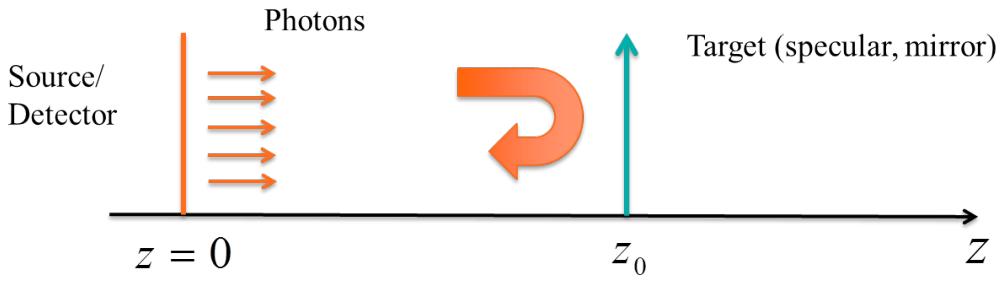


Figure 2.3. A toy example illustrating the FD-OCT Principle. In this simple setup, photons are emitted from different lateral positions of the source and then collected by an array of detectors, representing different A-scans. The sample only consists one mirror like, specular reflective target, located at depth z_0.

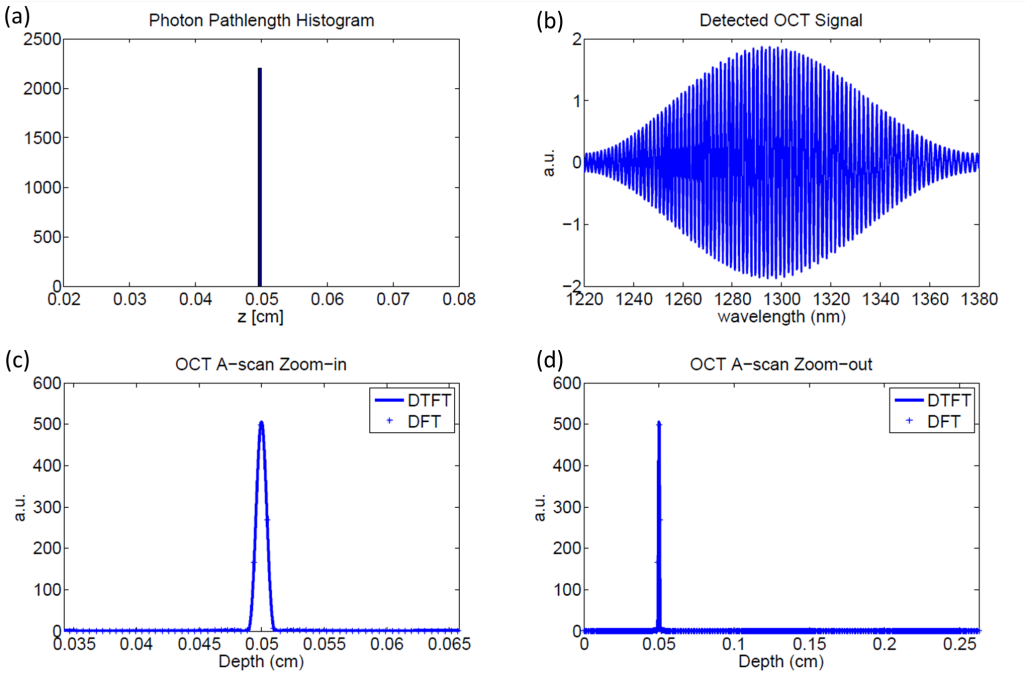


Figure 2.4. Image formation process for the toy example with a single target. (a) histogram of the photon pathlength; (b) OCT signal at the detector; (c) DTFT of the OCT signal, zoomed in to show the presence of the target; (d) DTFT of the OCT signal, zoomed out to show the entire A-scan.

Now, if we turn on the light source/detector pairs (e.g., a fiber), one at a time,

the photons will hit the target, get back-reflected and detected by the collecting fiber. All the photons that reach the detector would have a path length of exactly $s = 2z_0$, because the reflection is specular. During their round trip, these photons traveled straight lines and therefore their path lengths exactly matches twice the distance between the target and the source. Such photons are called ballistic or Class I photons in the literature. These Class I photons will have a contribution to the OCT signal of the form:

$$E_i = \sqrt{W_i} \cos\left(2z_0 \times k\right), \tag{2.3}$$

where E_i represents the contribution of the i-th photon, W_i is the weight (or strength) of the i-th photon, and $k = 2\pi/\lambda$ is the wavenumber of the laser source. In FD-OCT, the contributions of all the photons add up coherently. Now let's consider the situation that the wavenumber (frequency) of the laser source changes linearly with time, from k_{start} to k_{stop}. In other words, the wavenumber of the source would look like, in discrete time:

$$\mathbf{k}_{source} = [k_{start}, k_{start} + \Delta k, k_{start} + 2\Delta k, ..., k_{stop} - \Delta k, k_{stop}]. \tag{2.4}$$

By plugging in the vector $\mathbf{k}_{source}$ into Equation 2.3, we would recover a truncated, discrete cosine wave, if we treat $2z_0$ as its frequency and k as its time variable.

Next, since we are doing imaging, our goal is to recover the target position, namely z_0, from this truncated, discrete cosine wave. To do this, we rely on standard digital signal processing (DSP) by applying a Hamming Window, followed by a Fast Fourier Transform of the zero-padded signal. This procedure is demonstrated in Figure 2.4, where we successfully constructed an A-scan from the photon path distribution and it corresponds nicely to the position of the target.

2.2.2 Another Toy Example with Two Targets

We could repeat the same process with the presence of two mirror like targets, as shown in Figure 2.5, where we again start with the path length distributions and end

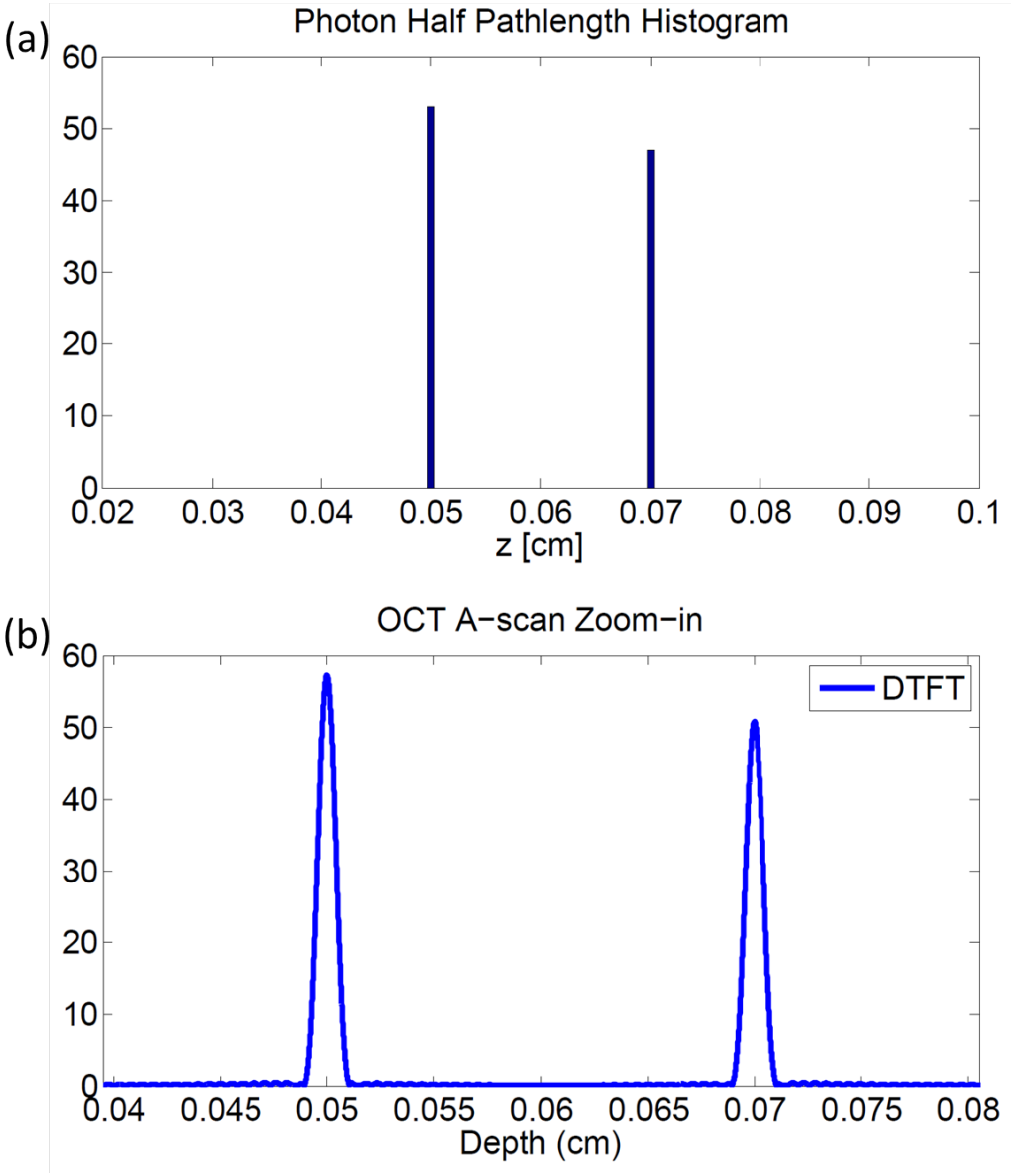


Figure 2.5. Image formation process for the toy example with two targets. (a) histogram of the photon half pathlength; (b) DTFT of the OCT signal at the detector, which gives us A-scan, displaying the two distinct targets;

up with A-scans. The result is exactly the same except that there are two peaks shown in the A-scan, as there should be. We can go a step further by stitching multiple A-scans and construct an OCT image, as shown in Figure 2.6. We could also dictate the number of contributing photons in each A-scan, and more contributing photons would generally produce better contrast of the resulting OCT image, as demonstrated

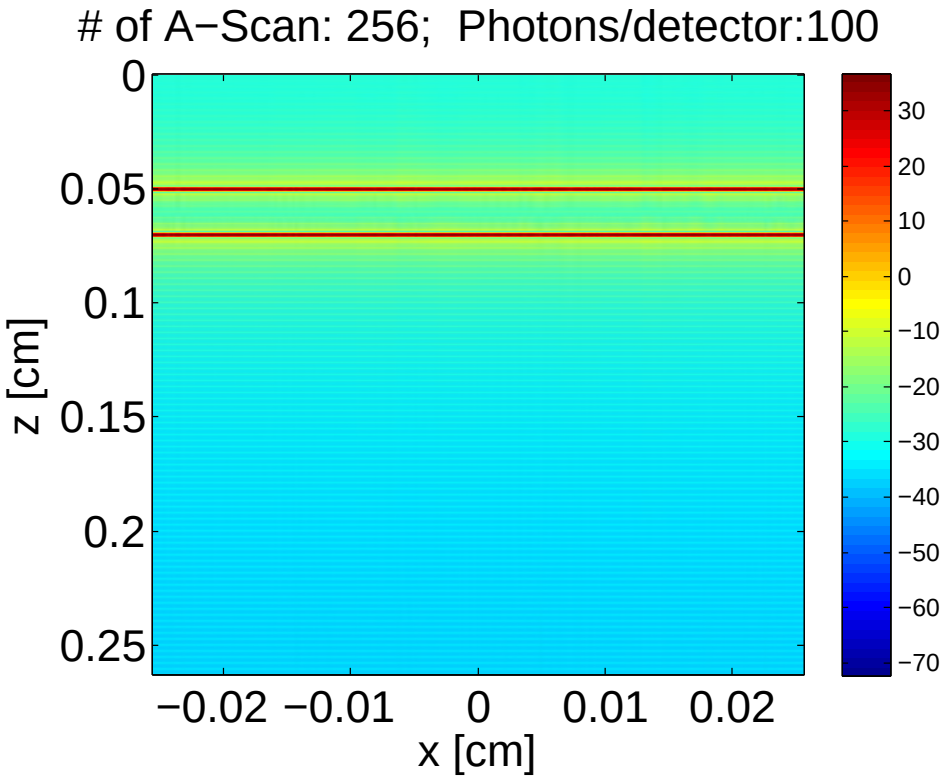


Figure 2.6. OCT Image formed by the toy example with two mirror like targets, consisted of 256 A-scans, where each A-scan has 100 contributing photons.

in Figure 2.7.

It is worth pointing out that in generating OCT images with the toy examples with two mirror-like targets, there has been very little simulation involved. We simply dictate the photon path lengths, which only has two possibilities, and then compute Fourier Transforms. There is no randomness in the process, except for maybe we flip a coin to determine the number of photons coming off each target. In other words, if we are only dealing with mirror-like targets in OCT, there is not much need for simulation since almost everything is *deterministic*.

2.3 From Mirrors to Biological Tissues

Life would be much easier if biological tissues, which we care about, behave like mirrors. If that is the case, the OCT imaging problem is completely solved, since

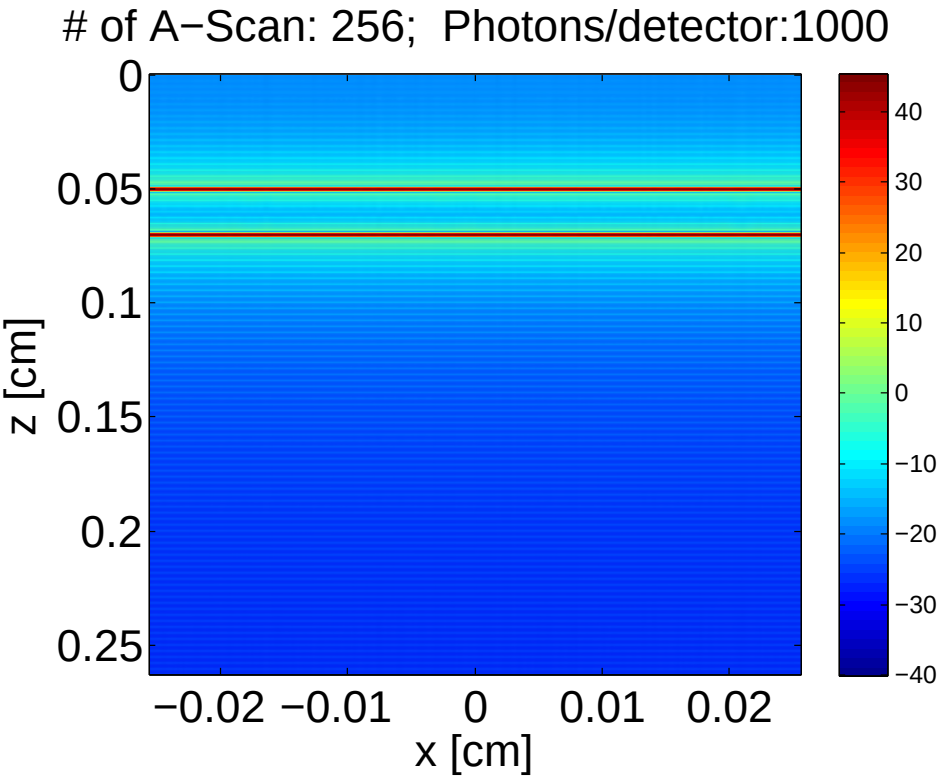


Figure 2.7. OCT Image formed by the toy example with two mirror like targets, consisted of 256 A-scans, where each A-scan has 1000 contributing photons.

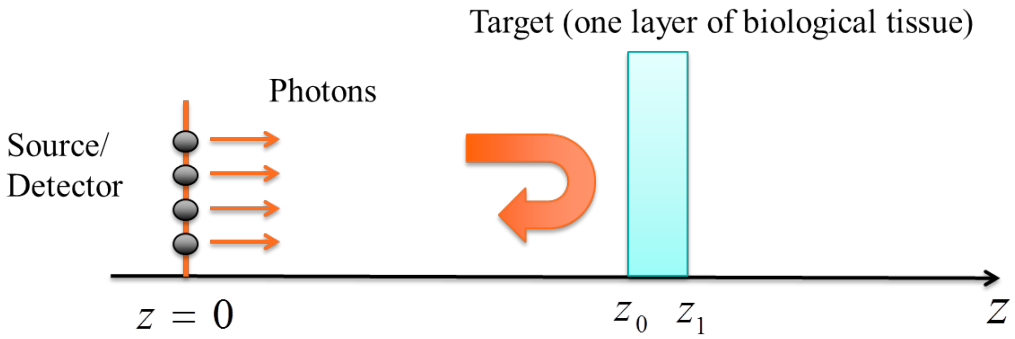


Figure 2.8. OCT imaging in a more realistic setting. The previous two mirror-like targets are replaced with a layers of biological tissue, located from depth z_0 to z_1.

the ground-truth, anatomical structure and the OCT image would be a deterministic one-to-one mapping, except for a smallish resolution issue. Unfortunately they don't, and not even close.

2.3.1 Photon Path Length Becomes A Distribution

So things become interesting when we replace the two mirror-like targets with a layer of biological tissue, as shown in Figure 2.8, as this time the photons detected would have a *distribution* of path lengths, ranging from $2z_0$ to $2z_1$, *approximately*. It is only an approximation because due to multiple scattering, the photon path length my exceed $2z_1$. When imaging a biological tissue, like the layered one in Figure 2.8, the photons will no longer go through a deterministic process as in the previous cases.

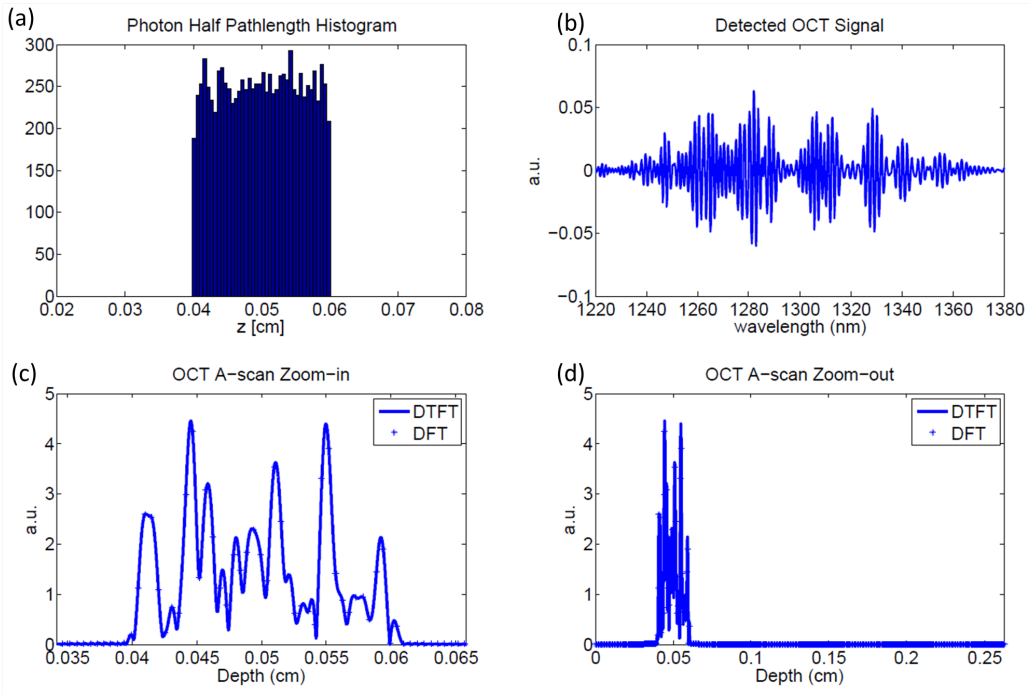


Figure 2.9. Image formation process for the example with one layer of biological tissue as the target. (a) histogram of the photon half pathlength; (b) OCT signal at the detector; (c) DTFT of the OCT signal, zoomed in to show the presence of the target; (d) DTFT of the OCT signal, zoomed out to show the entire A-scan.

The first notable difference is that the imaging photons can reach and get back-scattered anywhere within the tissue, and thus the path length of the photon is no longer a deterministic variable. As a photon enters the biological tissue, what happens is that the photon may travel an "interaction-free" path, which itself is a random

variable, and then interact with the tissue in the form of absorption and scattering. The photon has a certain probability of being absorbed by the tissue, and thus no longer contribute to the OCT signal, each time it interact with the tissue. Also, the tissue can and will alter the direction of propagation of the photon via scattering. Depending on the property of the biological tissue, the photon might be more likely to be scattered to one direction or the other, each time it experiences a scattering event. In other words, as the photon enters the tissue, it would go through a sequence of random events, after which the photon may leave the tissue or get absorbed by the tissue, becoming no longer relevant to the OCT signal, or it may be back-scattered and enters the detector, thus contributing to the OCT signal.

For those photons that survive the process and make it back to the detector, their path length would form a distribution, ranging approximately from $2z_0$ to $2z_1$, as explained earlier. It is instrumental to see what would the OCT signal and the corresponding A-scan look like when they are formed by a distribution of photon path lengths. This is demonstrated in Figure 2.9, where we choose a uniform distribution for the photon path lengths. This choice is obviously incorrect since the imaging photons are more likely to come from the surface region of the tissue (near z_0) than from the end of the tissue (near z_1). Nevertheless, this shows us what happens when you have effectively a *continuum* of targets.

2.3.2 A-scans Are No Longer Identical

The other consequence is that when there is a continuum of targets, the shape of the resulting A-scans are no longer identical, as shown in Figure 2.10. Here will see that even though the photon path distributions (in this case histograms) are similar, their corresponding A-scans are drastically different. They have different number of peaks (speckles), whose location and magnitude are also different.

This phenomenon is a natural result of coherent interference, where the spacing of the frequency components are narrower than the resolution of the OCT system, determined by the span of the frequency chirp. In practice, it means that A-scans

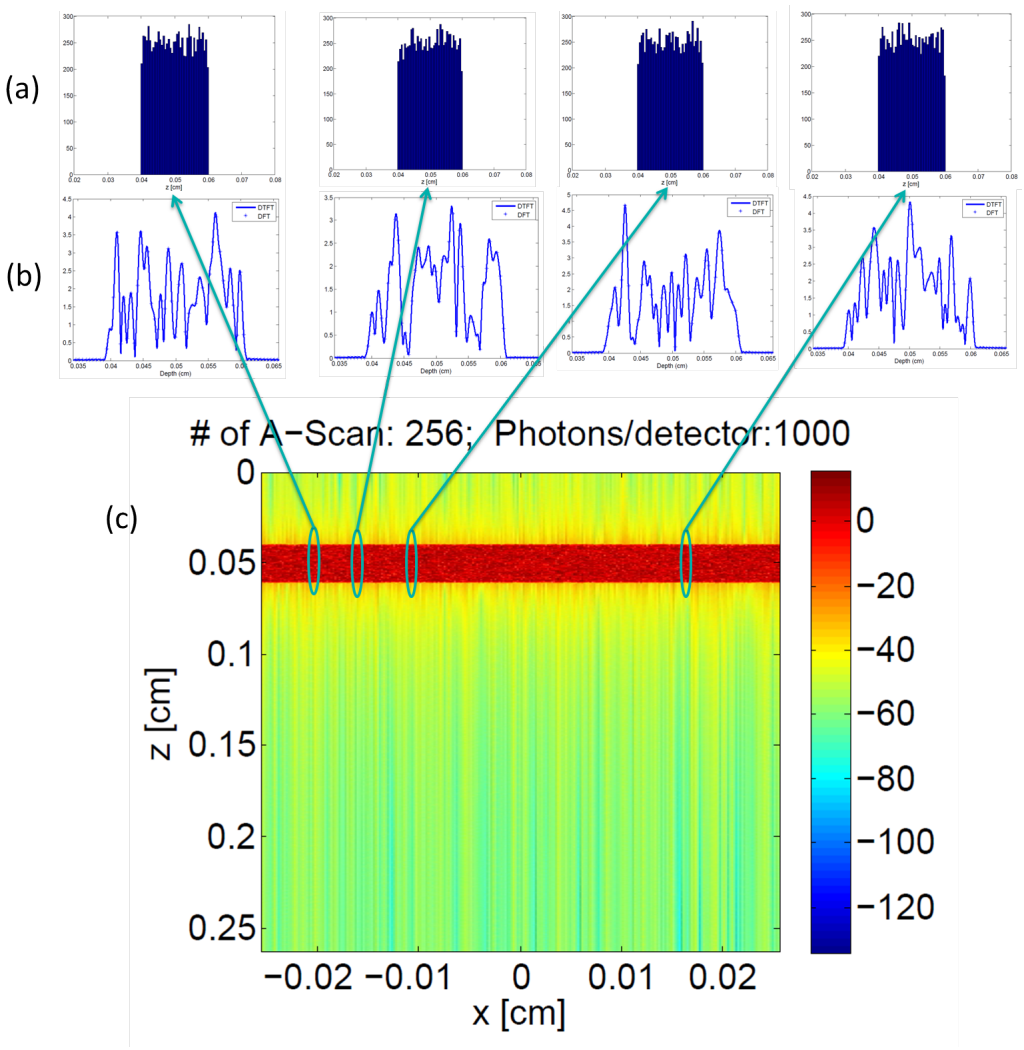


Figure 2.10. OCT Image formed by stitching 512 A-scans from the example with one layer of biological tissue as the target. (a) Top row: histograms of the photon half pathlength; (b) middle row: A-scans at different positions; (c) bottom row: the full OCT image formed by 512 A-scans.

corresponding to the same ground-truth, the same layered tissue in this case, will appear very much differently. This is one of the reasons why interpreting an OCT image is difficult, as the same anatomical structure can generate different-looking A-scans.

2.4 Reasons for the Need of Simulation

Hopefully be now you have a good qualitative understanding of the formation process of OCT signals. From the previous sections, it is easy to conclude that in order to generate an OCT image, all we need to do is construct the A-scans; and in order to construct an A-scan, all we need to know is the path length distribution of the contributing photons. But this is where we get stuck because there is no way we could know it without conducting an experiment or a computer simulation.

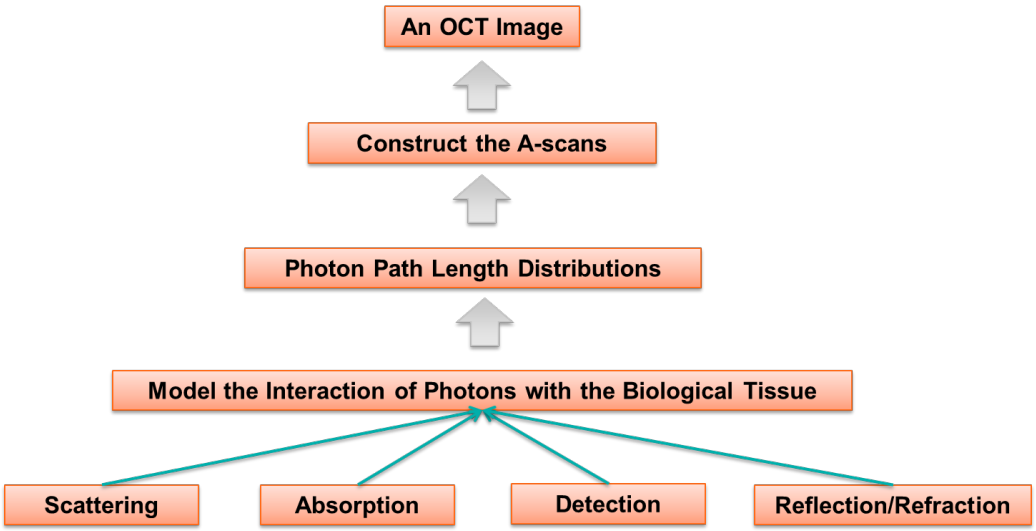


Figure 2.11. Flow chart of the modeling process.

In order to get a realistic distribution we need to model how the photons interact with the biological tissue, namely, how the photons get scattered, absorbed, reflected, refracted, and detected. In other words, we shoot the photons one by one and track its trajectory. Record its path length if the photon ends up getting detected. We repeat this process until we have enough detected photons. We then construct an OCT image bottom-up, as shown in Figure 2.11. Of course, this process of modeling is called Monte Carlo, which is the topic of next chapter.

Chapter 3

Monte Carlo Simulation of OCT

3.1 Chapter Overview

This chapter is intended to be a review of the standard Monte Carlo simulation procedure. By the end of the last chapter we already understand why we need a Monte Carlo simulation in oder to generate OCT images. In this chapter we begin with a general discussion of the principles of Monte Carlo simulation of light propagation in biological tissues, the big picture, important concepts and building blocks, as well as how can we use these to construct a platform to simulate OCT images. We then dig deeper into the issues behind conventional Monte Carlo simulation to gain some insight as to why it takes days to simulate a single OCT image previously. A thorough understanding of these issues is absolutely a prerequisite to come up with ways to speed up the simulation, which will be the topic of Chapter 4.

3.2 Introduction

As discussed in the previous chapter, the goal of simulation is to model the series of events that a photon experiences once it enters a biological tissue. This process, namely light propagation in random media, is illustrated in Figure 3.1 [13]. To describe this process, it is necessary to introduce the optical properties of tissues that governs the laws of light propagation inside the tissue.

The optical properties of a tissue are described in terms of the absorption coeffi-

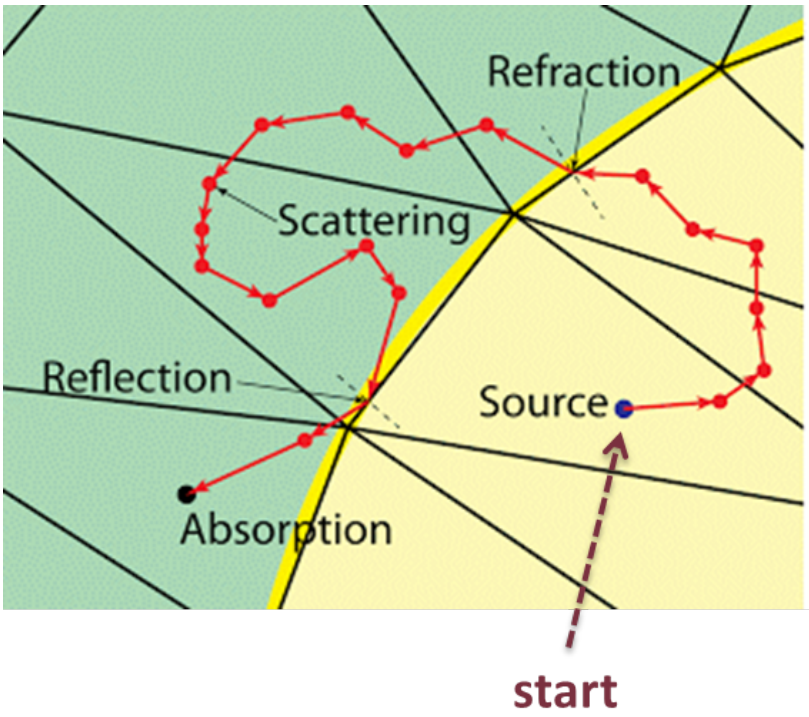


Figure 3.1. Illustration of the series of events that a photon experiences once it enters a biological tissue.

Absorption	μ_a	[cm^{-1}]
Scattering	μ_s	[cm^{-1}]
Scattering function	$p(\theta,\psi)$	[sr^{-1}]
Anisotropy	$g = \langle\cos\theta\rangle$	[-]
Real refractive index	n'	[-]
Reduced scattering	$\mu_s' = \mu_s(1-g)$	[cm^{-1}]

Figure 3.2. The optical properties of tissues.

cient, μ_a (cm^{-1}), the scattering coefficient μ_s (cm^{-1}), the scattering function $p(\theta, \psi)$ (sr^{-1}) where θ is the deflection angle of scatter and ψ is the azimuthal angle of scatter, and the real refractive index of the tissue, n'. An introduction to these properties is

presented elsewhere [14–16].

The $p(\theta, \psi)$ is appropriate when discussing only a single or few scattering events, such as during transmission microscopy of thin tissue sections or during confocal reflectance microscopy, which includes optical coherence tomography. In thicker tissues where multiple scattering occurs and the orientations of scattering structures in the tissue are randomly oriented, the ψ dependence of scattering is averaged and hence ignored, and the multiple scattering averages the θ such that an average parameter, $g = \langle \cos\theta \rangle$, called the anisotropy of scatter, characterizes tissue scattering in terms of the relative forward versus backward direction of scatter. Figure 3.2 summarizes these properties and their inter-relationships. For a thorough review of these properties, readers are encouraged to read the review article by Steven Jacques [17].

3.3 Basic Principles and the Monte Carlo Engine

Principles of stochastic Monte Carlo (MC) method for numerical calculation of radiation intensity scattered within a randomly inhomogeneous turbid medium are widely described in the literature [18–27]. MC is based on the consequent simulation of a random photon trajectory within the medium between the point where the photon enters the medium, and the point where it leaves the medium. Simulation of the photon trajectories consists of the following key stages: injection of the photon in the medium, generation of the photon path-length, generation of a scattering event, definition of reflection/refraction at the medium boundaries, definition of detection and accounting for the absorption.

3.3.1 Overview of the Simulation Steps

Photon packets begin with an initial weight, W, equal to 1. Once a photon packet is launched, the step length is selected randomly from an exponential distribution, the scale of which is defined by the optical properties of the tissue. The photon packet is then advanced along this direction orthogonal to the first surface until it either hits a boundary between regions, or it reaches the step length. If the photon packet

reaches the step length, its weight is decreased by the ratio of the medium absorption to total interaction coefficient. The remaining photon packet is then scattered with a deflection angle, θ , and an azimuthal angle, ψ, which are statistically sampled from the Henyey-Greenstein scattering phase function and between 0 and 2π, respectively. If the photon packet intersects a region boundary along its advancement before reaching the step length, the packet is moved to the boundary. Here it is internally reflected or transmitted according a random number generator with probabilities matched to the Fresnels equation (including regimes for total internal reflection). After a transmission or an internal reflection, the photon packet completes the remaining dimensionless step distance divided by the new region absorption and scattering coefficients (if transmitted). A photon packet is discarded if it crosses an outside boundary or by a Russian Roulette random process once weight values fall below a predefined threshold limit. In the method of [28], regions are defined by z-axis locations and it is trivial to identify boundary transections.

3.3.2 Mathematical Details of Implementation

For a good starting point to understand the details of this entire business I would recommend the Monte Carlo modeling of light transport in multi-layered tissues (MCML) with a C-language software package that is available for download from the web site of the Oregon Medical Laser Center [28]. MCML allows the simulation of an ensemble of photon packets launched in a steady-state pencil beam, normal to the surface of the topmost layer. Each photon packet produces a random walk whose step size is determined by an exponentially distributed random variable defined by the interaction coefficient μ_i, which is equal to the sum of the absorption μ_a and the scattering μ_s coefficients.

In other words, the photon free path s between the two successive elastic scattering events is determined by the Poisson probability density function [29]:

$$f(s) = \mu_i \exp(-\mu_i s), \tag{3.1}$$

where μ_i is the interaction coefficient. Note that the parameter$\bar{s} = 1/\mu_i$ which is the average scattering length depends on the size distribution of scatters, their concentration and relative refractive index in respect to the surrounding medium. The probability that the photon free path exceeds s is defined as:

$$\xi = \int_s^\infty f(s')ds'. \tag{3.2}$$

Given the probability density function in Equation 3.1 it is easy to express the random magnitude s via the probability ξ:

$$s = -\frac{\ln \xi}{\mu_i}. \tag{3.3}$$

This is the key element of MC technique, which is obtaining photon free path-length that consists of the computer generation of a random number ξ uniformly distributed in the interval $[0, 1]$.

The scattering events, which take place at the end of the random steps, are produced by two random angles that determine the future direction of the photon packet scattering in three-dimensional space. To account for the photon packet scattering with arbitrary anisotropy factor, g, we use the same Henyey-Greenstein probability density function used in the MCML software package, which is the most widely used model phase function for biotissues describes non-isotropic scattering depending on the anisotropy parameter $g = \overline{\cos\theta}$ [30]. The Henyey-Greenstein scattering phase function, is defined as

$$f_{HG}(\cos\theta_s) = \frac{1 - g^2}{2\left(1 + g^2 - 2g\cos\theta_s\right)} \tag{3.4}$$

where θ_s is the angle between the photon packet propagation direction $\hat{\mathbf{u}}$ prior to the scattering and the new scattering direction $\hat{\mathbf{u}}'$. After rotating away from the previous propagation direction $\hat{\mathbf{u}}$ by an angle θ_s, so that $\cos\theta_s = \hat{\mathbf{u}} \cdot \hat{\mathbf{u}}'$, the scattering direction $\hat{\mathbf{u}}'$ is rotated around the previous propagation direction $\hat{\mathbf{u}}$ by an angle ϕ that is randomly picked from a uniform probability density function from 0 to 2π. At each

scattering event, where the light-matter interaction is modeled, the weight W of the photon packet is decreased by an amount determined by the absorption coefficient μ_a at the scattering location. The weight W, which is initialized at 1, is an estimate of the residual number of photons left in the packet. When the packet weight reaches $W_{th} = 10^{-4}$, the photon packet is either eliminated with probability $1/m$ or is left to continue propagating with probability $1 - 1/m$ and weight equal to mW. In this work we use $m = 10$. This elimination process, called a Russian roulette technique, is an unbiased way to remove from the simulation the photon packets that have a negligible contribution to the scattering and absorption in the tissue, so that a new photon packet can be simulated.

Described steps are repeated till the photon is detected arriving at the detector with the given area and acceptance angle, or leaves the scattering medium, or its total path exceeds the maximum allowed path. The details of the reflection and refraction at the medium boundary and at the interface between layers are given in [31]. For an example of applying Monte Carlo method for simulation of 2D optical coherence tomography (OCT) images of skin-like model, where layer boundaries in skin model feature curved shape which agrees with physiological structure of human skin, please refer to the article by Mikhail Kirillin [32].

3.3.3 More Complex Geometry

To support a more complex geometry, the sample is subdivided into cuboidal voxels. We note that because these cuboidal voxels have boundaries that are always aligned to x/y axes, this method would not accurately estimate the specular (i.e., Fresnel) reflection from a tilted surface. In this approach, each voxel is assigned a specific tissue type with corresponding optical properties. The resolution of the geometry is defined by the voxel size, and can be reduced arbitrarily (at a computational cost) to simulate samples with higher resolution. For example, in microvascular networks, the smallest vessels of which are on the order of 4-9 μm, the recommended approach is to adopt a voxel sizes of 3 μm (27 μm^3), as shown in [1].

A boundary intersection algorithm needs to be implemented to first identify the voxel boundaries that are intersected by an advancing photon packet. If there are no cube boundary intersections or if there are intersections but each intersection occurs between voxels associated with the same region (and therefore identical optical properties), the photon packet is transported the full step length. If there are cube boundary intersections within the photon packet path and one of these intersections occurs between two voxels of different regions, the photon packet is transported to the first intersection between differing voxels where it is internally reflected or transmitted. Analogously to the multilayered approach, after a transmission or an internal reflection, the photon packet continues to travel as described above the step distance adjusted by the new region total interaction coefficient.

3.3.4 OCT Transverse and Axial Signal Localization

The aforementioned methods allow MC-based calculation of light transport in arbitrary three-dimensional structures. To extend this model to simulate OCT imaging, it is necessary to include methods for providing both transverse and axial signal localization (e.g., the imaging component of OCT). Transverse resolution can be incorporated as has been done in prior models [23, 33]. Briefly, photon packets are launched with a location defined by a uniform circular distribution. The angular position of photon packets is sampled uniformly between 0 and 2π while the radial position is sampled between 0 and a 10-μm beam radius with a square root distribution. On collection, exiting photon packets are included only if they meet the spatial and angular constraints of a detection aperture. In our model, a spatial filter with a 10-μm radius (uniform intensity) and an angular filter limited to a $\pm 5^\circ$ detection angle [23] was adopted. The location of the launch and collection apertures is translated across the imaging surface to simulate beam scanning.

For FD-OCT, we can model directly the generation of wavelength-dependent fringe signals, and then use Fourier analysis to convert these fringes to depth-resolved A-lines [1]. The simulation tool records the square root of the diffuse reflectance (weight)

and the total path length of each photon packet that passes the detection aperture. At the conclusion of the MC simulation for a given beam location (e.g, the modeling of one A-line), we obtain a set of detected photons packets, each with a diffuse reflectance amplitude and a path length. We then construct the received sample signal (amplitude and phase) as a function of wavelength according to

$$S(k) = \sqrt{G(k)} \sum_{n=1}^{N} \sqrt{W_n} \exp(ikl_n) \tag{3.5}$$

where W_n and l_n respectively represent the weight and path length of the n-th detected photon packet. k corresponds to the wavenumber and $G(k)$, to the Gaussian light source spectrum defined as

$$G(k) \propto \exp\left[-\left(\frac{k-k_0}{dk}\right)^2\right] \tag{3.6}$$

where k_0 represents the center wavenumber of the light source and dk the spectral bandwidth. An important note is that for swept-source OCT, where the imaging light source is a swept-frequency laser, we can simply treat $G(k) = 1$ and ignore it. The same diffuse reflectance, R_n, and path length, l_n, arrays are used for each k. We calculate S across a discrete k array between k_{start} ($\lambda = 1220$nm) and k_{stop} ($\lambda = 1380$nm) with equal spacing in k.

The reference field is calculated as

$$R(k) = \sqrt{\alpha G(k)} \tag{3.7}$$

where α is a parameter chosen to scale the total sample arm power ($|S(k)^2|$) between $0.001 - 0.01\%$ of the reference arm power ($|R(k)|^2$). The value of the parameter α is constant across a simulation for all A-lines.

We interfere the sample and reference arm fields in a balanced detection configu-

ration to generate a fringe signal given by

$$I_D(k) = |S(k) + R(k)|^2 - |S(k) - R(k)|^2. \tag{3.8}$$

We then follow conventional Fourier-domain processing by applying a window (Hamming) and Fourier transform to give the OCT signal as a function of depth, as illustrated in the previous chapter.

3.3.5 OCT Angiography and Blood Flow Simulation

It is also possible to simulate moving part within the tissue using OCT, for example, a blood flow. This is named OCT angiography and is based on the acquisition of repeated measurements from the same location and the analysis of these measurements to identify time-varying signals that result from flowing blood. To simulate angiographic OCT images, it is therefore necessary to derive time-series measurements that are constant or modulated according to their degree of interaction with intravascular scatterers. In is reasonable to assume that the intravascular scatterers have translated by more than the larger of the axial or transverse resolutions. As a result there is no correlation between the OCT signals scattered from vessels across two time points. This accurately describes many of the angiographic OCT approaches that employ large time separations between measurements to achieve sensitivity for slower flowing vessels.

To model temporal decorrelation of intravascular scatterers, the following approach can be used [1]. First, during MC light transport, we flag each photon packet (using the vessel flag variable, F_n) that scatters within a voxel associated with the region defining the intravascular space. Then, during calculation of the OCT signal, we include in each photon packet that is flagged a random phase, ϕ_{1n}, between 0 and 2π within the exponential. To simulate a second measurement, at the same location, the same MC solution is used but a new set of random phases,ϕ_{2n}, are applied. In this way, the OCT signals derived from photon packets that have not interacted with intravascular scatterers are constant in time, but those signals that are derived, at

least in part, from photon packets scattered from intravascular scatterers are proportionally modulated.

3.3.6 Putting It Together

Now we understand how to run a Monte Carlo simulation to generate OCT images. The algorithmic view of this is illustrated in Figure 3.3 [1]. We call this the standard Monte Carlo routine since it represents the minimum one would need in order to simulate an OCT image correctly.

We could list the algorithmic steps involved as follows:

- Launch a photon packet;
- Set a step size;
- Try move the photon;
- Check cube boundaries;
- Move the photon to the boundary;
- Check photon detected/escaped;
- Complete the step size;
- Absorb (reduce weight);
- Scatter (change propagation direction);
- Check photon dead;
- Record photon.

Depending on the details of implementation, some of the steps might be slow or fast relative to others. And it is also possible that one step or two are significantly slower than the rest of them. Therefore identifying and improving such bottle-neck steps are of key importance in order to speed up the simulation.

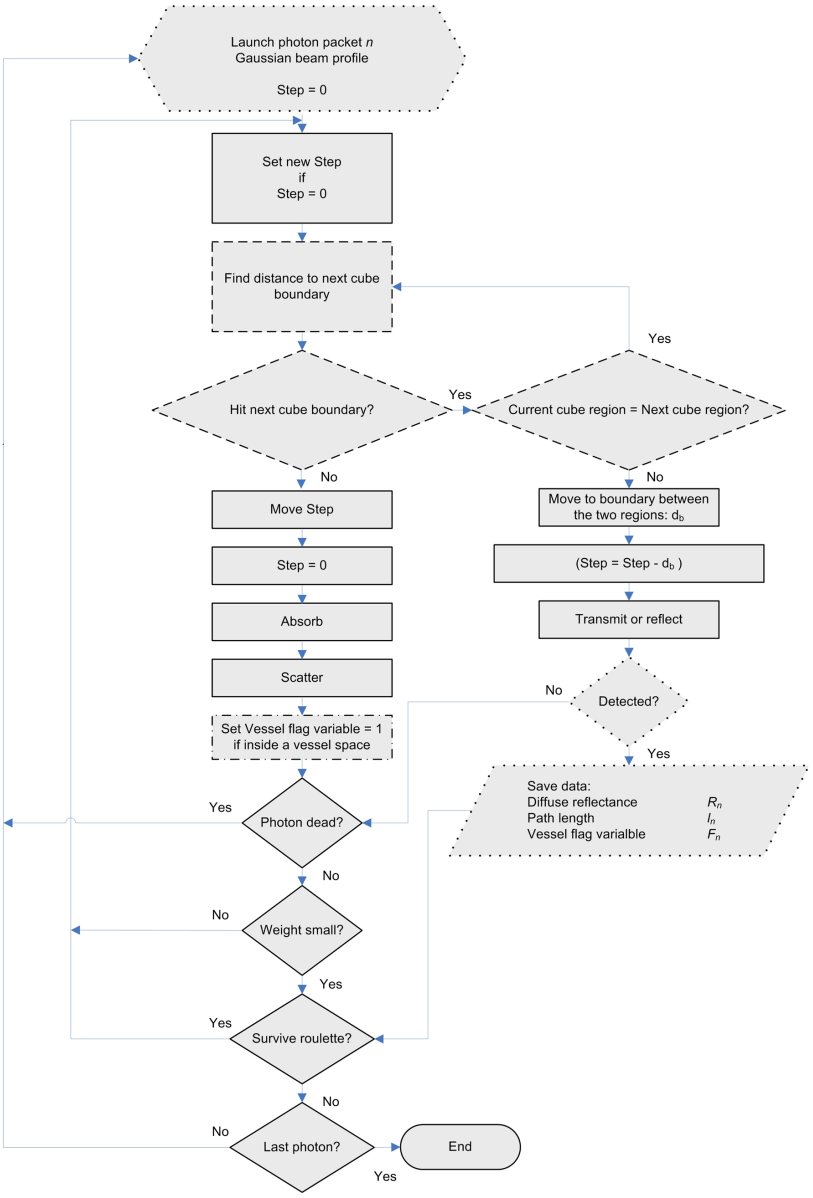


Figure 3.3. Flow chart of the MC algorithm.

3.4 Class I and Class II Photons

We introduce another fundamental concept before we move on to more advanced simulation techniques. OCT imaging considers the ballistic and quasi-ballistic photons reflected from a target layer inside the tissue, which we denote Class I diffuse reflectance, as the OCT signal that produces the image [23]. However multiple scattered photons reflected from the tissue, which we denote Class II diffuse reflectance,

do not carry useful information about the target layer and are considered a source of noise in OCT [34]. It has been demonstrated that Class II diffuse reflectance is the fundamental limitation in increasing the imaging depth of OCT in tissue [35]. As shown in Figure 3.4, the path length of the Class I photons matches the depth target layer, i.e., the maximum depth that the photon reached, z_{max} is close to half of the photon path length $l_n/2$. On the other hand, for Class II photons, there is a mismatch between the depth of the target layer and the photon path lengths. In other words, the Class II photons appear to image depth $l_n/2$, but it never reached there since $z_{max} < l_n/2$. Understanding the physical process governing both Class I and Class II diffusive reflectance is an important prerequisite to any effort to increase penetration depth and to enhance the quality of the images obtained with OCT. As this physical process is complicated, Monte Carlo simulation of light transport in tissue [28,36–38] has been used to obtain the TD-OCT signal [23], as well as the FD-OCT signal [1].

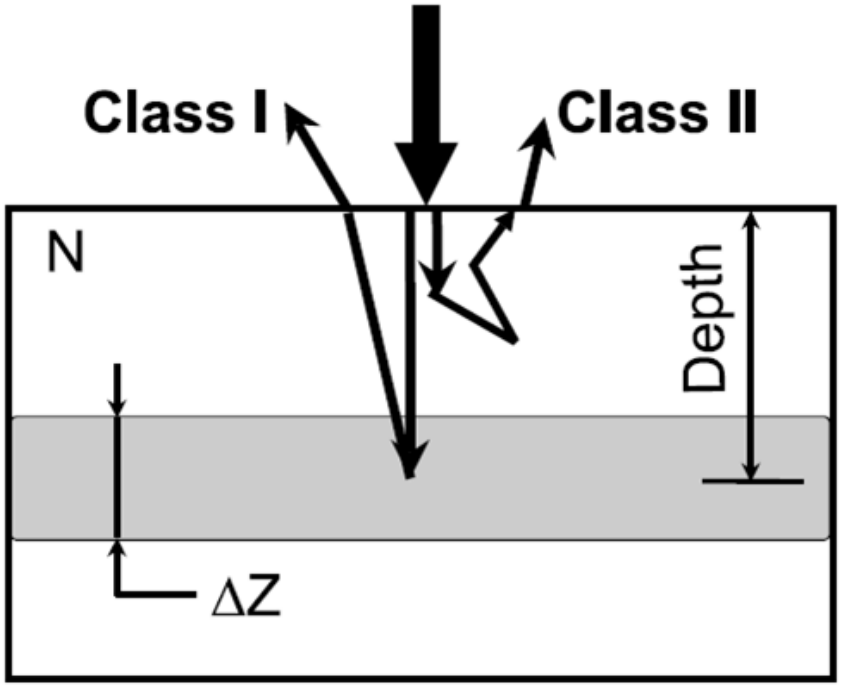


Figure 3.4. Class I and Class II photons.

3.5 Why the Simulation is Slow

It is not hard to conceive why simulating an OCT image using the standard Monte Carlo routine can take very long. Among the many reasons, the most prominent and simplest one is that *detecting a photon is an extremely rare event*. In order to record a single detected photon, you may need to launch 1000 to $10,000$ photons depending on the detection setup.

Intuitively this is very easy to understand: once you let loose of a launched photon (packet), it randomly changes propagation direction according to the scattering properties of the media each time it experiences a scattering event. Moreover, photons traveling in biological tissues exhibit much more forward scattering than backward scattering, i.e., the anisotropic factor g is close to 1, which further reduces the chances that the photon gets detected. Also, the mean free path length of photons is very short (on the order of 10μm-100μm), which means that each photon will go through many scattering events, which means that the simulation of each photon packet would require looping through the flow chart shown in Figure 3.3 over and over again. It is worth pointing out that using photon packets (with initial weight $W = 1$) instead of discrete photons has the effect of simulating many photons at the same time, effectively reducing the variance and the simulation time.

3.5.1 FD-OCT vs TD-OCT

It turns out that when it comes to simulate an OCT image, there is an inherent advantage of using FD-OCT as opposed to TD-OCT. To understand this subtle advantage, we need to understand how the reflectance is calculated in both FD-and TD-OCT. The former is already introduced in the previous section, and we take a look at the latter in what follows.

3.5.2 Calculation of the Reflectance in TD-OCT

In TD-OCT, to compute the reflectance at depth z, i.e., a particular point of an A-scan, we simply combine the Class I and the Class II reflectances.

The Class I and the Class II reflectances at depth z are obtained by calculating the mean value of the indicator functions I_1 and I_2 of the spatial and temporal filter of the Class I reflectance and the Class II reflectance, respectively, for all the photon packets (samples) in the ensemble. The indicator function I_1 and I_2 of the spatial and temporal filter for the i-th photon packet is defined as

$$I_1(z,i) = \begin{cases} 1, & l_c > |s_i - 2z_{max}|, r_i < d_{max}, \theta_{z,i} < \theta_{max}, |s_i - 2z| < l_c \\ 0, & \text{otherwise} \end{cases} \tag{3.9}$$

and

$$I_2(z,i) = \begin{cases} 1, & l_c < |s_i - 2z_{max}|, r_i < d_{max}, \theta_{z,i} < \theta_{max}, |s_i - 2z| < l_c \\ 0, & \text{otherwise} \end{cases}, \tag{3.10}$$

where z is the depth imaged, l_c is the coherence length of the source, r_i is the distance of the i-th reflected photon packet to the origin (where the detector/source pair is located) along the plane $z = 0$, where the collecting optical system is located, d_{max} and θ_{max} are the maximum collecting diameter and angle, respectively, $\theta_{z,i}$ is the angle with the z-axis, which is the axis normal to the tissue interface, s_i is the optical path, z_{max} is the maximum depth reached by the photon packet. A detected photon packet is considered a Class II photon packet if it does not reach a depth that is consistent with its optical path, so that it interferes constructively with corresponding detected Class I photons packets without bringing any information from the depth in which those Class I photons packets were reflected.

The estimated values of the Class I and Class II reflectances and their respective

variances are given by the following expressions

$$R_{1,2}(z) = \frac{1}{N}\sum_{i=1}^{N} I_{1,2}(z,i)W(i) \tag{3.11}$$

and

$$\sigma^2_{1,2}(z) = \frac{1}{N(N-1)}\sum_{i=1}^{N} \left[I_{1,2}(z,i)W(i) - R_{1,2}(z)\right]^2, \tag{3.12}$$

where $W(i)$ is the weight of the i-th photon packet in MCML, which is a quantity affected by the absorption coefficient at the scattering points.

3.5.3 Discussion

For TD-OCT, as shown in Equations 3.11 and 3.12, in order to generate an A-scan, we need to collect enough photon packets for each and every depth z to reach a certain degree of convergence. We are talking about 10^{11} launched photons per A-scan [39].

The implication is that despite the structure of the object being imaged, you need to look for photons reflected at each depth position z and make sure you have enough photons to guarantee the $R(z)$ has converged. For deeper positions, i.e., bigger z, you have fewer photons to begin with (since fewer photons reached there), and therefore it is much harder (in fact exponentially harder) for the reflectance at deeper positions to converge. To maintain a uniform convergence level, all you can do is to increase the overall launched photons, which is a huge overhead since most of the photons collected does not help the convergence at deeper regions.

On the other hand, FD-OCT make use of the collected photons in a way that is proportional to the complexity of the imaged object. For example, to image a single, mirror-like target, we could collect just a few photons and still construct the correct OCT image. In other words, FD-OCT has a global view of the object structure, and thus is able to efficiently determine a minimum number of collected photons that are needed to construct an A-scan, while TD-OCT only has access to local information by looking at each depth z. More specifically, in FD-OCT a single collected photon could give you the entire $R(z)$ at all values of z, but in FD-OCT you may need many

collected photons to obtain $R(z)$ at a single z. This particular advantage of FD-OCT allows the programmer to decide how many photons need to be collected for the problem at hand, which will make a big difference as we will see in the next chapter.

Chapter 4

Speeding up the Monte Carlo

4.1 Chapter Overview

In this chapter we will show in detail how we managed to bring down the simulation time of an OCT image from 360 hours to a single minute. We begin by identifying three main challenges that prevent us from fast Monte Carlo simulation of OCT images and then address them accordingly. To make this chapter less technical, most of the source code listings are left in the appendix and interested readers are welcome to visit in order to fully appreciate the simulation procedure.

4.2 Three Challenges

In this section we identify the three challenges. In the previous chapter we have already seen that one major challenge is the fact that *detecting a photon is an extremely rare event*. The other two are more technical and dig into the details of implementation.

- Detecting a photon packet is an extremely rare event, which only happens 1 out of 1000-10, 000 times when you launch a photon packet. This means that most of your Monte Carlo simulation is completely useless since more than 99.9% of the photon packets simulated does not contribute to your OCT signal.

- During the lifetime of a simulated photon packet, the fact that its mean free path

length is short (on the order of 10μm - 100μm), means that each photon will loop through the standard Monte Carlo routine (as shown in Figure 3.3) over and over again. A photon packet will travel a distance equal to the mean free path after which it undergoes absorption and scattering. The photon packet rates of absorption and scattering depend on the absorption and scattering coefficients of the regions where this packet is traveling, respectively. If a traveling photon packet enters a region with a different refractive index, then it will undergo both specular reflection and refraction at the boundary between the two regions. Therefore, at each step, we have to check if the packet path and the enclosing mesh intersect. This process of identifying photon-mesh intersection is the most computationally demanding task in tracing a photon path through complicated heterogeneous structures [13].

- An OCT image is consisted of multiple (e.g. 512) A-scans, where each A-scan launches and collects photon packets at a different lateral position. Before you can start simulating an A-scan, there is a lot of preprocessing that needs to be done. For example, you need to load the mesh profile, set up the variables, allocate the memories, and so on. That's a lot of computational overhead. Repeating these preprocessing for all the 512 A-scans is a complete waste and ideally you'd like to do it only once.

As advertised earlier, addressing these three challenges will be the theme of this chapter.

4.3 Importance Sampling and Photon Splitting

We will use Monte-Carlo simulation with *importance sampling* to calculate the OCT signal because the probability that a photon propagating in typical biological tissue will undergo single-scattered back-reflection is very low. This very low probability of events of interest would require an unacceptably large computational time if standard Monte Carlo simulation were used.

4.3.1 Introduction

Importance Sampling is an advanced statistical method [40] that consists of biasing random events in such a way that the events of interest, which are often rare, appear more often in Monte Carlo simulations [41–47]. To produce the correct statistical result in Monte Carlo simulation with importance sampling, each biased sample is weighted by the likelihood in which this sample would have been observed in the unbiased simulation. This procedure, which has to be tailored to each particular application, can reduce the computational time of Monte Carlo simulations by several orders of magnitude.

Importance sampling, tailored to each particular application, has been applied in optical communications [41–44], confocal microscopy [45], atmospheric optics [46] diffuse optical tomography (DOT) [38] and TD-OCT [23]. In the applications involving Monte Carlo simulation of light transport in three-dimensional space, one limitation that can be addressed is the statistical bias that increases with depth as the photons are preferentially scattered in the backward direction by importance sampling based implementations.

4.3.2 Mathematical Details

A photon packet in turbid tissue will scatter according to the Henyey–Greenstein probability density function that is defined as

$$f_{HG}(\cos\theta_s) = \frac{1-g^2}{2\left(1+g^2-2g\cos\theta_s\right)} \tag{4.1}$$

where θ_s is the angle between the photon packet propagation direction $\hat{\mathbf{u}} = (u_x, u_y, u_z)$ prior to the scattering and the new scattering direction $\hat{\mathbf{u}}'$. After rotating away from the previous propagation direction $\hat{\mathbf{u}}$ by an angle θ_s, so that $\cos\theta_s = \hat{\mathbf{u}} \cdot \hat{\mathbf{u}}'$, the scattering direction$\hat{\mathbf{u}}'$ is rotated around the previous propagation direction $\hat{\mathbf{u}}$ by an angle ϕ that is randomly picked from a uniform probability density function from 0 to 2π.

Since most biological tissues are generally highly forward scattering, the value of their anisotropy factor is close to 1. Thus there is a very small probability that a simulated photon packet at a given depth undergoes backscattering towards the tip of the collecting fiber. This already small probability of collecting photons decreases rapidly with the depth from which the photon is backscattered.

To speed up the simulation of collected photons, we developed a novel importance sampling method for Monte Carlo simulations that bias the scattered photon packet direction $\hat{\mathbf{u}}'$ preferentially towards the actual position of the center tip of the collecting optical system $\hat{\mathbf{v}}$, once the photon packet is traveling away from the probe ($u_z > 0$), to increase the probability of its detection.

Following an approach similar to the one described in Lima et al [39], we use a biased probability density function for the first biased scattering that produces random scattering with an angle no lesser than 90 degrees away from the direction to the actual position of the collecting optics. This biased probability density function, which was based on the Henyey-Greenstein probability density function in Equation 4.1, is given by

$$f_B(\cos\theta_B) = \begin{cases} \left(1 - \frac{1-a}{\sqrt{a^2+1}}\right)^{-1} \frac{a(1-a)}{(1+a^2-2a\cos\theta_B)^{3/2}}, & \cos\theta_B \in [0,1] \\ 0, & \text{otherwise} \end{cases} \tag{4.2}$$

where a is a bias coefficient in the range $(0, 1)$. After randomly picking a biased angle θ_B away from the direction of the actual position of the collecting optics, the biased direction $\hat{\mathbf{v}}$, so that $\cos\theta_B = \hat{\mathbf{v}} \cdot \hat{\mathbf{u}}'$, the resultant biased scattering direction $\hat{\mathbf{u}}'$ is rotated around $\hat{\mathbf{v}}$ by an angle ϕ that is randomly picked from a uniform probability density function from 0 to 2π. This last procedure is equivalent to the one used in the MCML software package to enable a full three-dimensional scattering. Then, the scattered photon packet is associated with a quantity that is defined as the likelihood ratio [41, 43, 44], which ensure converge of the statistical result towards the expected value. The likelihood ratio of the photon packet using the biased probability density

function in Equation 4.2 is given by

$$L(\cos\theta_B) = \frac{f_{HG}(\cos\theta_S)}{f_B(\cos\theta_B)} = \frac{1-g^2}{2a(1-a)}\left(1-\frac{1-a}{\sqrt{a^2+1}}\right)\left(\frac{1+a^2-2a\cos\theta_B}{1+g^2-2g\cos\theta_S}\right)^{3/2} \tag{4.3}$$

where $\cos\theta_S = \hat{\mathbf{u}}\cdot\hat{\mathbf{u}}'$ is a function of $\cos\theta_B$, which is statistically drawn from the probability density function in Equation 4.2 that is used to define the new propagation direction $\hat{\mathbf{u}}'$ of the photon packet at the first biased scattering. A schematic representation of the angles and vectors used in the biased and in the unbiased scatterings is shown in Figure 4.1 [48].

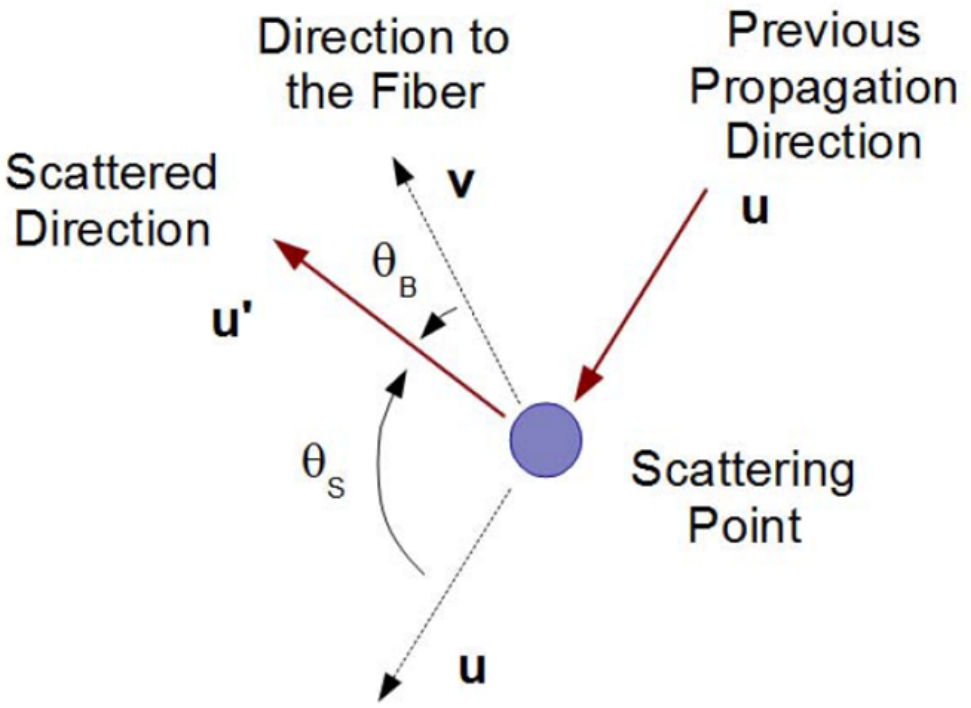


Figure 4.1. Schematic representation of the vectors and the angles used in the bias procedure.

Other probability density functions can also effectively speed up the calculation using this method, provided that they significantly increase the probability that the photon packet is scattered towards the apparent position of the collecting optics.

4.3.3 Additional biased scatterings

After the first biased scattering, the photon packet can undergo additional biased scatterings towards the actual direction of the collecting fiber $\hat{\mathbf{v}}$ with probability $0 \leq p \leq 1$, as opposed to $p = 1$ proposed in a previous method [48]. If a biased scattering is not applied at a given point in the tissue, the photon packet will undergo an unbiased scattering at that location according to the probability density function in Equation 4.1. The likelihood ratio of this scattering, whether the biased or the unbiased probability density function is randomly selected, is calculated according to the expression

$$L(\cos\theta_B) = \frac{f_{HG}(\cos\theta_S)}{pf_B(\cos\theta_B) + (1-p)f_{HG}(\cos\theta_S)} \tag{4.4}$$

If the biased function $f_B(\cos\theta_B)$ is selected to draw a random value of $\cos\theta_B$, which is an event with probability p, $\cos\theta_S = \hat{\mathbf{u}} \cdot \hat{\mathbf{u}}'$ is a function of $\cos\theta_B$ that is statistically drawn from the probability density function in Equation 4.2. Otherwise, which is a complementary event with probability $1 - p$, the unbiased function $f_{HG}(\cos\theta_S)$ is selected to draw a random value of is selected to draw a random value of $\cos\theta_S$ and $\cos\theta_B = \hat{\mathbf{v}} \cdot \hat{\mathbf{u}}'$ is a function of $\cos\theta_S$. Because each random scattering is independent from the other scatterings, the likelihood ratio of each photon packet is equal to the product of all the likelihood ratios of all the biased scatterings of that photon packet.

4.3.4 Photon Splitting

One limitation that results from the use of previously existing bias methods is the underestimation of the diffuse reflectance beyond the start of the target depth range in which the diffusive scattering is biased. The application of the first bias in the backward direction implies that there is a smaller probability that the photon packets will be forward scattered beyond the early part of the target depth range. This causes a systematic statistical bias that also affects the performance of the angle biasing procedure used in [23] and also the bias procedure used in [37, 38], which limits their application to a thin target layer. For these reasons, these bias procedures are limited to model systems in which only forward bias scattering is applied.

We ensure the correct statistics in the importance sampling method by splitting the photon packet that is biased towards the actual position of the collecting optics in two photon packets prior to the first biased scattering [19]. One of these two photon packets is the one biased towards the collecting optical system, which is associated with the likelihood ratio specified in Equation 4.3, as described earlier in this section, and we apply the successive biased scatterings until the photon packet is terminated. Then the other photon packet continues a forward propagation starting at the location of the biased backward scattering point, having the initial direction $\hat{\mathbf{u}}$ and scattered by the save procedure that is described in previous sections. We ensure that there is no systematic statistical bias in the statistical result of the forward propagating split photon packet by assigning to this second photon the likelihood ratio $L'(i)$, which is the complement of the likelihood ratio of the biased backward scattered photon packet $L(i)$, so that $L'(i) = 1 - L(i)$. This second photon packet, which is only created if $L(i) < 1$, is also allowed to undergo one biased backscattering, which may lead to another photon packet split, and successive additional biased scatterings towards the tip of the collecting optical system until the photon packet propagates beyond the simulation domain. In the cases that we studied, this procedure increased the computational time of each sample by a factor of 5 when compared with a sample computed using the standard Monte Carlo. The increase of computational time of importance sampling compared with the standard method depends on the average value of the mean free path and on the width of the target depth range. Once a photon packet exceeds the region within the depth target layer, it will no longer be biased and will likely be terminated after exceeding the boundary of the last layer while propagating in the forward direction. Then a new photon packet will be created at the origin as in the standard MCML method, and a new Monte Carlo sample will be simulated. Despite the higher computational cost per photon packet, the computation of the diffuse reflectance in the case that we studied using Monte Carlo simulations with importance sampling required as little as one-thousandth of the computational time required to achieve the same accuracy in the diffuse reflectance calculation using the standard Monte Carlo method.

4.3.5 Calculation of the reflectance

For TD-OCT, the estimated values of the Class I and Class II reflectances and their respective variances, taking into account for the likelihood ratios associated with importance sampling, are given by the following expressions

$$R_{1,2}(z) = \frac{1}{N}\sum_{i=1}^{N} I_1(z,i)L(i)W(i) \tag{4.5}$$

and

$$\sigma^2_{1,2}(z) = \frac{1}{N(N-1)}\sum_{i=1}^{N}\left[I_1(z,i)L(i)W(i) - R_{1,2}(z)\right]^2, \tag{4.6}$$

where $W(i)$ is the weight of the i-th photon packet in MCML, which is a quantity affected by the absorption coefficient at the scattering points, and $L(i)$ is the product of the likelihood ratios of all the biased scatterings that affected the i-th photon packet. Using the Monte Carlo method with the importance sampling implementation described in this section, the calculation of the reflectances in Equation 4.5 converge two to three orders of magnitude faster with the number of samples N than the standard Monte Carlo method used in MCML. It is important to point out that each split photon packet is not counted as an additional photon packet when determining the value of N in Equations 4.5 and 4.6, since the use of the likelihood ratio associated to each photon packet in these equations will produce the correct statistical result.

For FD-OCT, we then construct the received sample signal (amplitude and phase) as a function of wavelength according to

$$S(k) = \sqrt{G(k)}\sum_{n=1}^{N}\sqrt{W_n L_n}\exp(ikl_n) \tag{4.7}$$

where W_n, l_n, and L_n respectively represent the weight, path length and the likelihood ratio of the n-th detected photon packet.

4.3.6 Generation of random biased angles

To generate random angles according to the biased probability density function in Equation 4.2 we use the pseudo-random number generator of the Gnu Scientific Library [49]. That random number generator produces pseudo-random numbers uniformly distributed from 0 to 1, which we convert to the probability density function in Equation 4.2 according to the following conversion formula

$$\cos\theta_{B,i} = \frac{1}{2a}\left\{a^2 + 1 - \left[u_i\left(\frac{1}{1-a} - \frac{1}{\sqrt{a^2+1}}\right) + \frac{1}{\sqrt{a^2+1}}\right]^{-2}\right\} \tag{4.8}$$

where u_i is sampled from the random number generator of the Gnu Scientific Library. Equation 4.8 was derived based on the probability theory in [50].

4.4 Identifying Photon-Mesh Intersection

The most demanding task in tracing a photon path through complicated heterogeneous structures is to identify photon-mesh intersection. In order to perform this task more efficiently, we decided to use a voxel-based mesh as opposed to, for example, a tetrahedron-based mesh [11, 13].

As illustrated in Figure 4.2, once we decide that the current step size $s = |\mathrm{P}_{N+1} - \mathrm{P}_N|$ of the photon will drive the photon out of the current voxel, i.e., P_N and P_{N+1} are not in the same voxel, we then need to figure out the intersection points, C_1 and C_2 in this case, so that we could move the photon there. We introduce below a clever mathematical trick to accomplish this.

Assume the photon is currently located at $P_N = (x_1, y_1, z_1)$, and propagates along (u_x, u_y, u_z). In order to compute the distance to the nearest boundary, first we locate

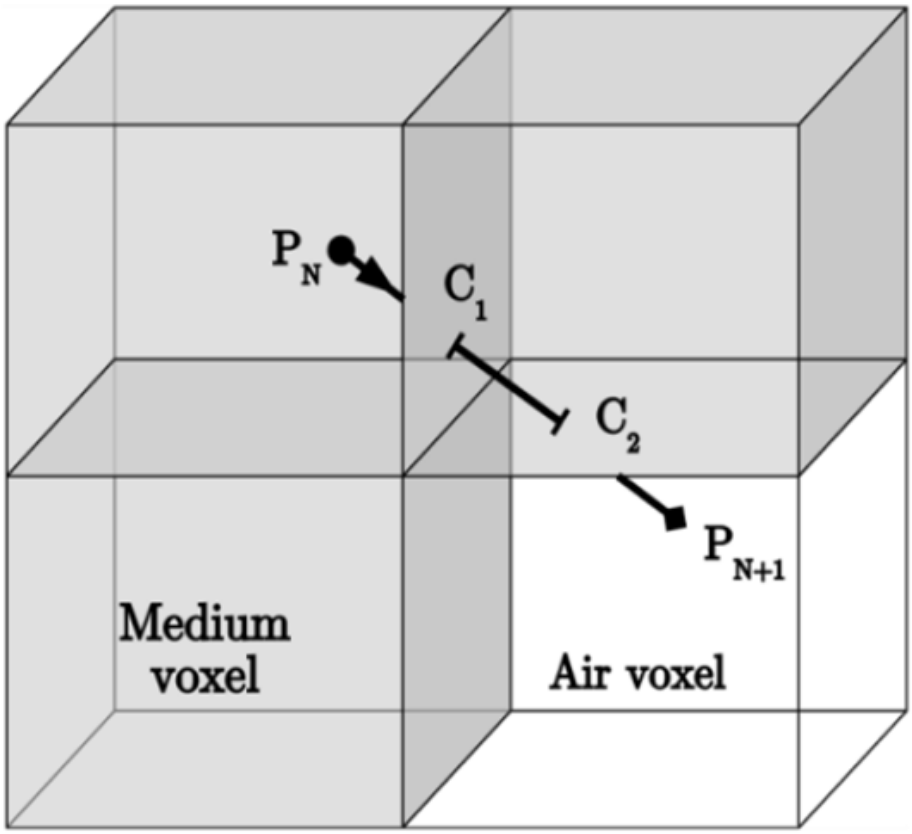


Figure 4.2. Schematic illustration of photon-mesh intersection.

the voxel faces ahead the photon's propagation direction:

$$ix_1 = \lfloor x_1/dx \rfloor \tag{4.9}$$

$$iy_1 = \lfloor y_1/dy \rfloor \tag{4.10}$$

$$iz_1 = \lfloor z_1/dz \rfloor \tag{4.11}$$

where ix_1, iy_1, iz_1 are the indices of the voxel where the photon is currently in. And

we then use

$$ix_2 = \begin{cases} ix_1 + 1, & u_x >= 0 \\ ix_1, & \text{otherwise} \end{cases} \tag{4.12}$$

$$iy_2 = \begin{cases} iy_1 + 1, & u_y >= 0 \\ iy_1, & \text{otherwise} \end{cases} \tag{4.13}$$

$$iz_2 = \begin{cases} iz_1 + 1, & u_z >= 0 \\ iz_1, & \text{otherwise} \end{cases} \tag{4.14}$$

where ix_2, iy_2, iz_2 are the indices of the voxel faces lying ahead of the photon's propagation path. Next we compute the distances from these voxel faces to the current position of the photon, utilizing its propagation directions:

$$x_s = (ix_2 \times dx - x_1)/u_x \tag{4.15}$$

$$y_s = (ix_2 \times dy - y_1)/u_y \tag{4.16}$$

$$z_s = (ix_2 \times dz - z_1)/u_z. \tag{4.17}$$

And the desired distance of the photon to its closest voxel face is thus given by:

$$s = \min(x_s, y_s, z_s). \tag{4.18}$$

4.5 Parallelization of A-Scans

We have learned that an OCT image consists of multiple (e.g. 512) A-scans, where each A-scan launches and collects photon packets at a different lateral position, and that is the only difference among A-scans. However, all the A-scans share the same preprocessing steps prior to their simulation. For example, you need to load the mesh profile, set up the variables, allocate the memories, and so on. That's a lot of computational overhead. Certainly you do not want repeat these preprocessing for

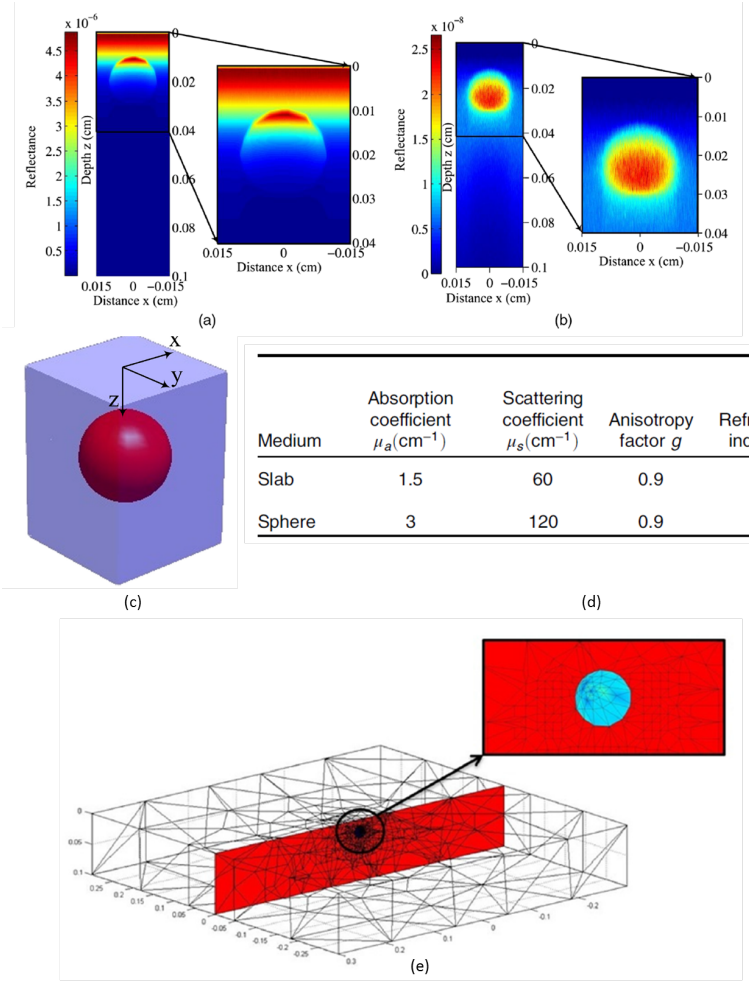
all the different A-scans, which is a complete waste. Ideally you'd like to perform the preprocessing only once and here is one way to do it.

- Use a line source instead of a point source. What I mean by a line source is that we now have an array of source/detector pairs, e.g., 512 of them.
- Each time a new photon packet is launched, we randomly pick one source/detector pair. Each pair has an equal chance of getting selected.
- Specify the initial state (starting position, propagation direction, etc.) of the launched photon packet according to the selected source.
- Apply the importance sampling & photon splitting techniques to the launched photon packet, and *always bias towards the selected detector*.
- If the photon packet is detected by a different detector other than the selected one, discard this photon packet.

It is worth pointing out that each time a photon splitting takes place (i.e., when a biased back-scattering happens), we save the information of the continuing photon packet (we call it the **continuing photon**, $\text{photon}_{continue}$), and keep simulating the back-scattered photon packet (we call it the **current photon**, $\text{photon}_{current}$). The current photon will then undergoes biased forward-scatterings (with probability p) as well as unbiased forward-scatterings (with probability $1-p$) until it is detected. Note that biased forward-scattering does not generate a photon split; only the biased backward-scatterings does (if the likelihood ratio is less than 1). Once the simulation of the current photon terminates, we load the saved continuing photon and it becomes the current photon. Therefore, at every moment you are only dealing with one photon packet, the current photon. And there could be at most one continuing photon saved.

4.6 Simulation Results

In this section we compare our simulation results (which is FD-OCT) against the start-of-the-art, of both TD-OCT and FD-OCT imaging.



Medium	Absorption coefficient $\mu_a(\text{cm}^{-1})$	Scattering coefficient $\mu_s(\text{cm}^{-1})$	Anisotropy factor g	Refractive index n
Slab	1.5	60	0.9	1
Sphere	3	120	0.9	1

Figure 4.3. State-of-the-art TD-OCT simulation results, which utilized importance sampling and is based on a tetrahedron mesh. The object represents a 3D arbitrary shape. The resulting OCT image consists of 512 A-scans, where each A-scan took 43 minutes to simulating, resulting in a simulation time of 360 hours for the entire OCT image. Simulated (a) Class I and (b) Class II reflectance-based B-scan OCT image of the non-layered object; (c) Spatial structure of the object; (d) Optical parameters of the non-layered object used in the tetrahedron-based OCT simulator; (e) A depiction of the tetrahedrons representing the non-layered object.

Figure 4.3 depicts the best TD-OCT results in the literature so far [11], which utilized importance sampling and is based on a tetrahedron mesh. The simulated

object represents a 3D arbitrary shape, while previous efforts have been focused on layered objects only. The resulting OCT image consists of 512 A-scans, where each A-scan took 43 minutes to simulating, resulting in a total simulation time of 360 hours for the entire OCT image.

There hasn't been much attention in the literature towards simulating FD-OCT images, and the only one available in the literature is illustrated in Figure 4.4, where the author simulated two cylindrical blood vessels surrounded by a homogeneous sample [1]. This simulation did not utilize importance sampling and its estimated simulation time is 36 seconds per A-scan, resulting in a total of 5 hours if their image consists of 512 A-scans. Despite the apparent faster simulation time compared to the TD-OCT result shown in Figure 4.3, it looks like this FD-OCT simulation hasn't converged yet, since the two blood vessels are hardly visible and the author needed to use yellow circles to let the reader know where they are. We can conclude that the existing FD-OCT simulation is far from satisfactory.

In our simulation, we attempt the same problem of simulating two cylindrical blood vessels as in [1]. We have used all the techniques described in the previous sections and the simulation result is shown in Figure 4.5. Our simulation only takes *one minute*. Clearly, the quality of our simulation result is much more satisfactory than those in [1], since the two blood vessels are perfectly visible and well defined in our case. The resulting OCT image has all the correct features, namely, the speckles due to coherent interference, as well as the typical shadowing effect. The structure two blood vessel is essentially the same as the sphere structure in the TD-OCT simulation as shown in Figure 4.3, in the regard that they both represent an arbitrary, circular shaped 3D structure. The quality of our simulated imaging is also similar to the TD-OCT result, but our simulation time is more than $10,000$ times faster. Lots of factors contribute to this drastic speed improvement, for example, the usage of voxel-based mesh combined with the mathematical trick of identifying photon-mesh intersections, the fact that we are implementing FD-OCT as opposed to TD-OCT in order to set a minimum number of collected photons, parallelization of A-scans, and so on. We will show the most notable implementation details in the section below and list the

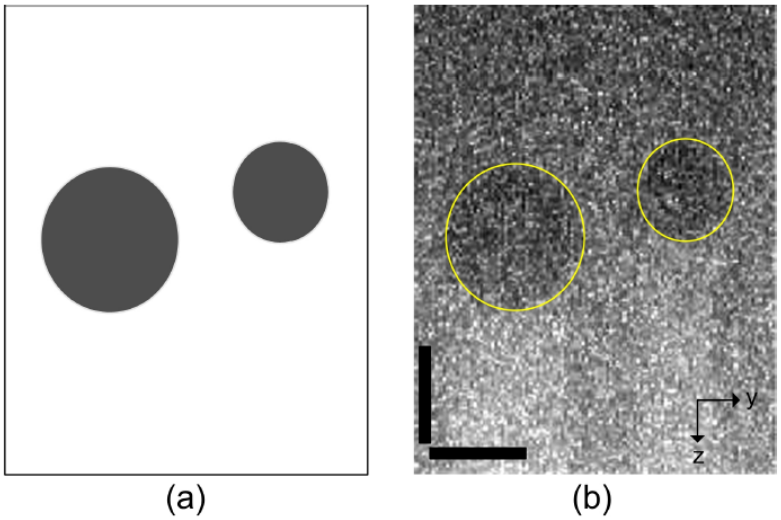


Figure 4.4. Simulated blood vessels. (a) Two blood vessels modeled at a depth of 250 μm and 200 μm. (b) Structural OCT image of two cylindrical blood vessels (outlined in yellow) surrounded by a homogeneous sample. The parameter μ_s for blood is fixed to 650 cm^{-1}, μ_a , to 5 cm^{-1}, g, to 0.9888, and n, to 1.37. The parameter μ_s for the sample is set to 10 cm^{-1}, μ_a, to 1 cm^{-1}, g, to 0.7, and n, to 1.37. Scale bars correspond to 100 μm.

more standard procedures in the Appendix.

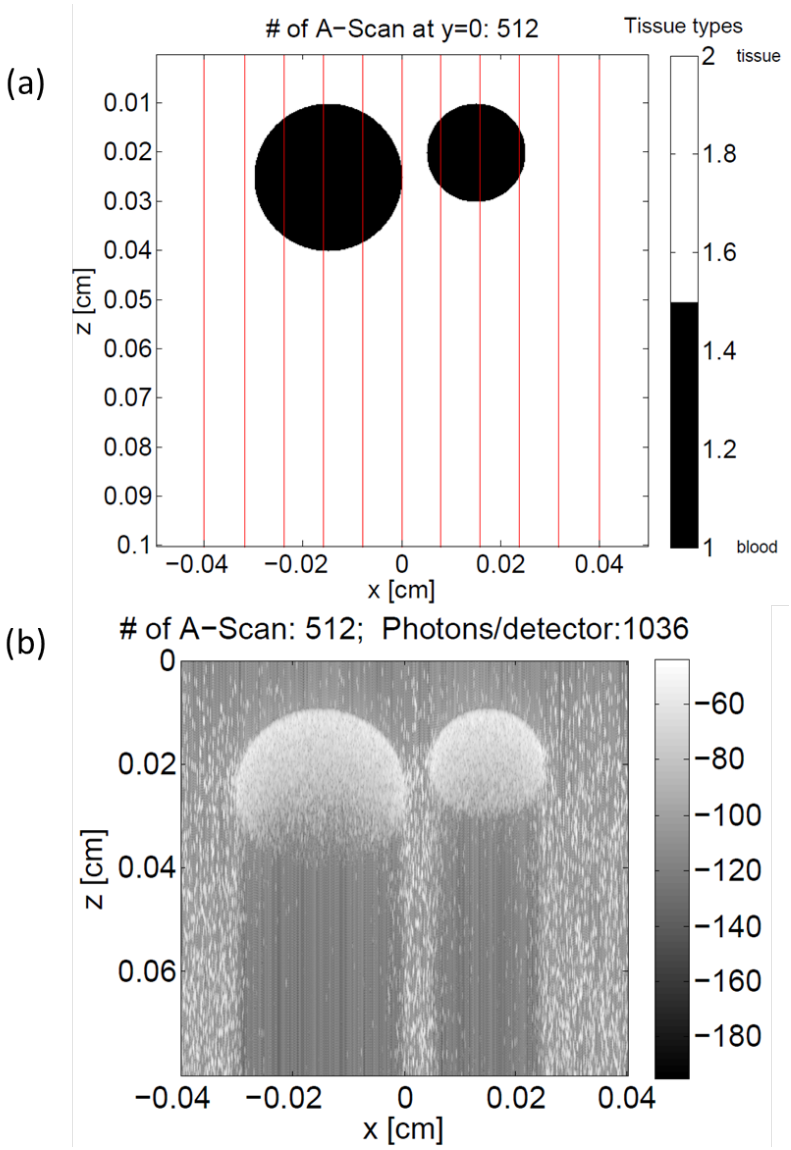


Figure 4.5. Same blood vessels simulated using our advanced Monte Carlo platform, consisting of 512 A-scans. The simulation only takes ***one minute***. (a) Two blood vessels modeled at a depth of 250 μm and 200 μm. The red lines represent the range covered by the A-scans. (b) Structural OCT image of two cylindrical blood vessels surrounded by a homogeneous sample. The simulation parameters are the same as in [1], where μ_s for blood is fixed to 650 cm^{-1}, μ_a , to 5 cm^{-1}, g, to 0.9888, and n, to 1.37. The parameter μ_s for the sample is set to 10 cm^{-1}, μ_a, to 1 cm^{-1}, g, to 0.7, and n, to 1.37. Scale bars correspond to 100 μm.

Chapter 5

The Learning Problem

5.1 Chapter Overview

The advanced Monte Carlo platform that we have developed opens up tremendous research opportunities: machine learning, compressed sensing, flood illumination, learning in the compressed domain, etc., just to name a few. In the rest of this thesis we will focus on the inverse problem: given an OCT image, predict its ground-truth structure on a pixel level. Solving this inverse problem would lead us to a completely new philosophy of medical imaging by handing the doctor and the patient precisely the anatomical structure, i.e., what you see is what you will get. No one has ever attempted at it because there is simply not enough annotated data in the OCT world, but our advanced Monte Carlo platform has completely solved this issue.

The fact that we can simulate OCT images in a minute or so, basically means that we could design and generate data sets at will for learning from them, and we put ourselves in a great position to solve the inverse problem using machine learning. It is always a good idea to start from examining simple structures in order to appreciate harder, more complex problems, which we will do next. This chapter is a warm-up towards solving the learning problem.

5.2 Layered Biological Tissues

We will begin by looking at OCT images of some simple structures to gain insight for the problem. And going back to visit the problem of imaging a layered biological tissue again is good start.

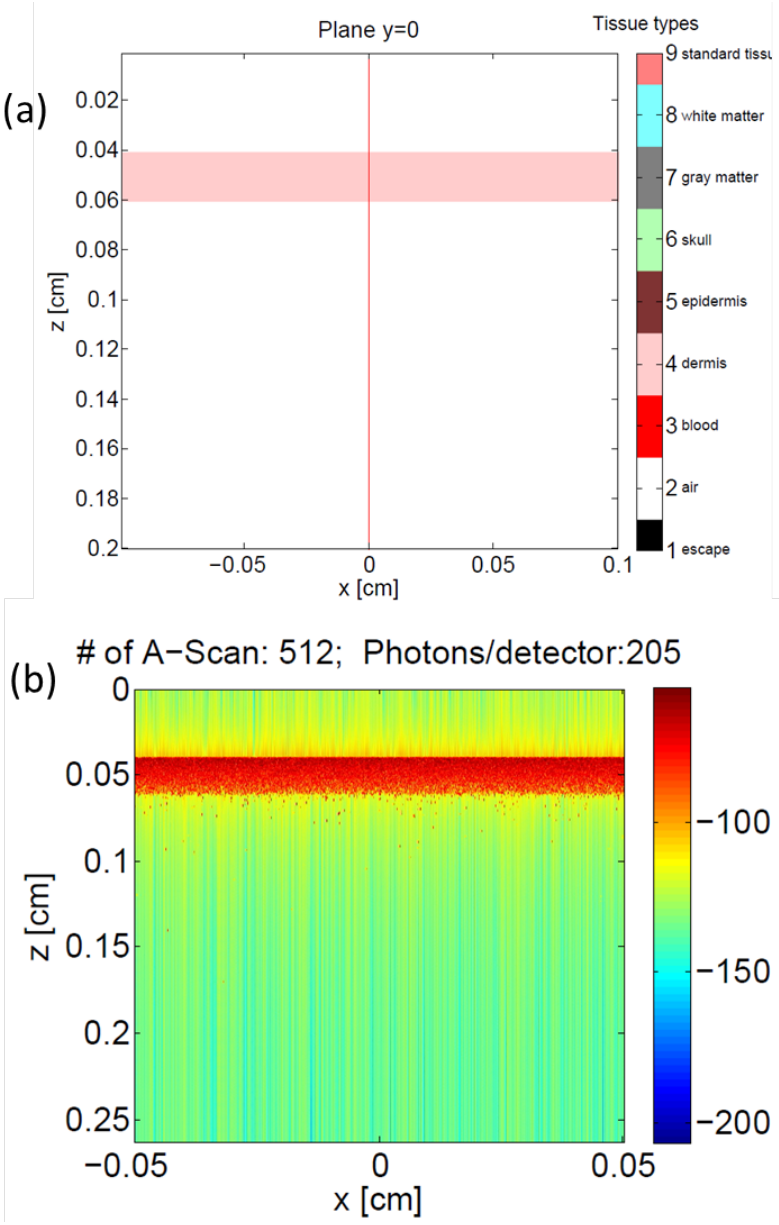


Figure 5.1. Monte Carlo simulation of a single layer biological tissue (dermis). The OCT image consists 512 A-scans and the simulation time is one minute. (a) Schematic drawing of the anatomical, ground-truth structure. (b) Simulated OCT image.

5.2.1 Single Layered Biological Tissues

As shown in Figure 5.1, the simulated image gives us what we expected to see: the speckle pattern, the decay of the signal strength from the surface of the layered tissue, and the differences from A-scan to A-scan. We can zoom in on the A-scan to see the decay of the signal strength, as shown in Figure 5.2. We also show the ground-truth structure indicated by the orange line. It can be seen that even though the bulk of the signal aligns well with the ground-truth, there is a small tail at the end of the signal, from 0.06 cm to 0.065 cm.

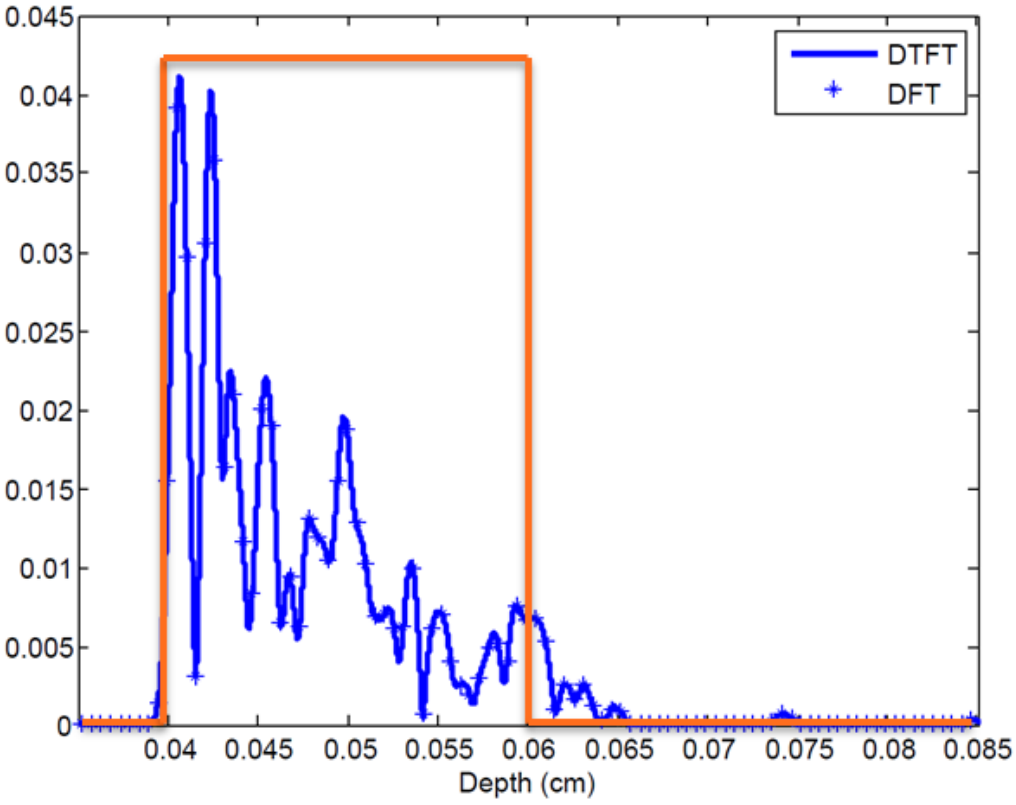


Figure 5.2. An example A-scan from the simulated OCT image of the one-layer biological tissue. The line in orange shows the corresponding ground-truth structure of this A-scan.

A look at the distribution of the photon path lengths reveals the why we see such a tail, as shown in Figure 5.3. It is worth pointing out that it has to be a weighted histogram since each photon carries a weight and a likelihood ratio and the

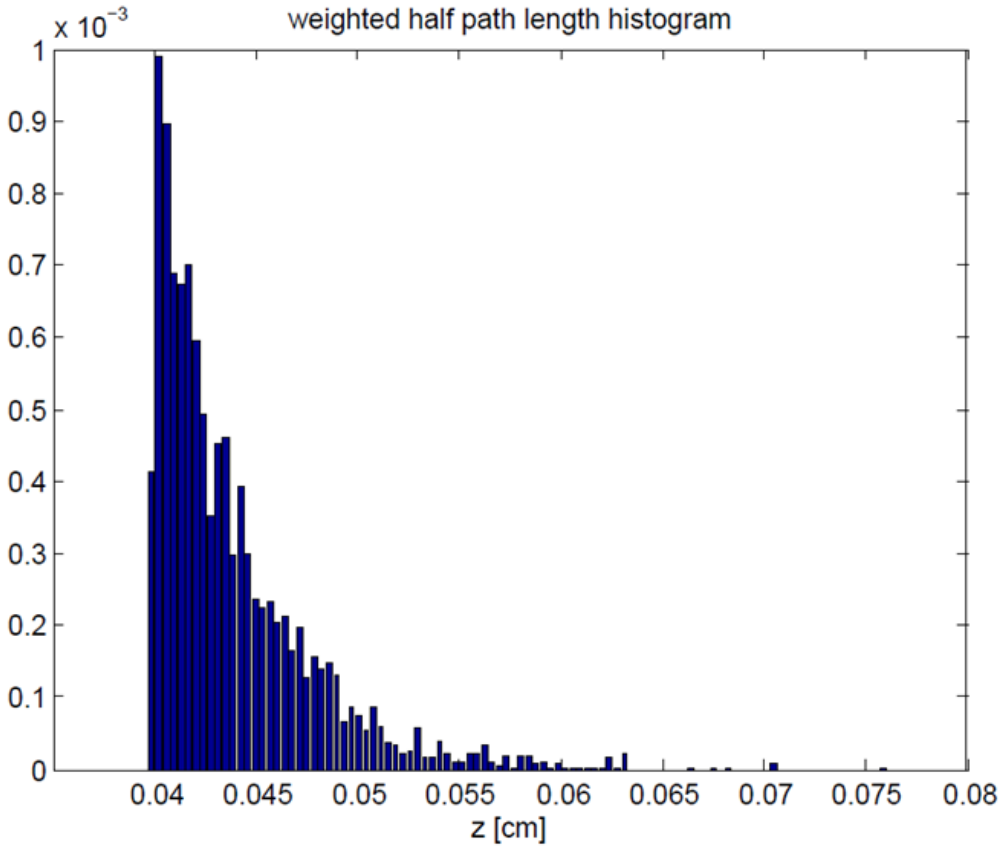


Figure 5.3. The weighted half path length histogram of the example A-scan from the simulated OCT image of the one-layer biological tissue. The photon half path lengths are weighted by the weight W_i and the likelihood ratio L_i they carry.

path lengths are weighted accordingly. Due to the importance sampling procedure, there is also filtering process associated with it to remove the outliers in likelihood ratio. Even though detecting a photon packet is an extremely rare event, we do get lucky sometimes if we launch $100,000$ photons. However, these "lucky" photons are not good for convergence since they will have a huge likelihood ratio associated with them compared to others, which would appear as outliers in the histogram and noisy spikes at the A-scan. A simple quantile function in MATLAB does the job as shown in Listing 5.1. This simple trick pays great dividends in that it allows us to further reduce the amount of collected photons to reach convergence.

```
%remove the outliers using .9 quantile of L
L_threshold = quantile(DetL,0.9);
ix = find(DetL<L_threshold);
DetL = DetL(ix);
DetS = DetS(ix);
DetW = DetW(ix);
DetID = DetID(ix);
DetZ = DetZ(ix);
Ncount = length(DetID);
% end of removing outliers
```

Listing 5.1: Filter the outliers among the detected photon packets with huge likelihood ratios.

From a prediction prospective, the small tail shown in Figure 5.2 should worry us because the machine learning model needs to figure out the tail is misleading and does not correspond to real tissues. The origin of this tail is due to the existence of Class II photons that experienced non-ballistic, multiple scattering, as can be see from the weighted histogram of the photon path lengths shown in Figure 5.3. This matter gets worse if we add more tissue layers or allow flood illumination.

5.2.2 Flood Illumination

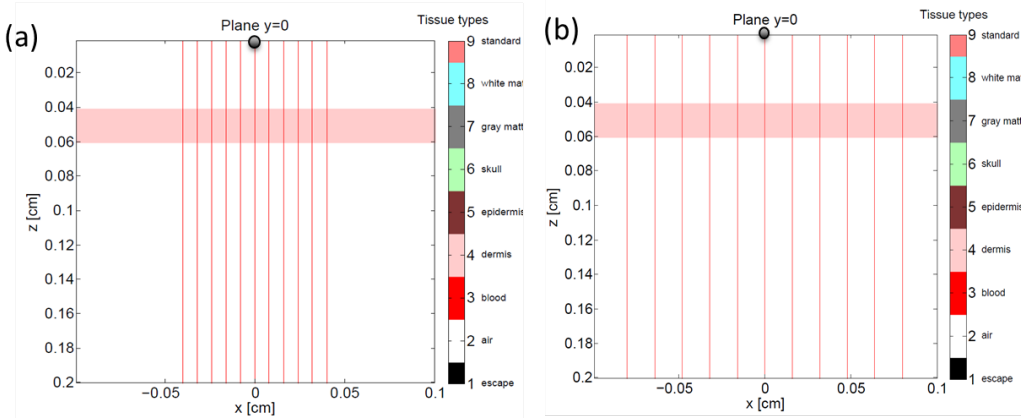


Figure 5.4. Schematic illustration of flood Illumination. (a) The radius of the illumination source equals 0.04 cm. (b) The radius of the illumination source equals 0.08 cm.

Flood illumination is always an interesting topic in OCT as it gives us the possibility of acquiring the entire OCT image at one shot as opposed to stitching it with A-scans, which is a slow process and requires moving parts. The idea of flood illumination is simply that, instead of using a point source, we use a disk source that could illuminate a larger region of the tissue. It is easy to imagine that the resulting image would be even harder to interpret, since the photons collected at a particular detector, as indicated by the blacks dots in Figure 5.4, may come from parts of the source other that the location of the detector. As a result, the path length of the imaging photons are further away from the truth depth of the region imaged, leading to even longer tails in the resulting the A-scans. Examples of A-scans under the condition of flood illumination can be found in Figure 5.5 in the next section, where we see the tails are much longer compared with A-scans under point illumination.

5.2.3 Sliding Window Prediction

In order to solve the inverse problem, essentially we need to give a label, i.e., the tissue type for each pixel in the OCT image. The standard approach in image processing and computer vision when it comes to labeling an image is sliding window prediction. The idea of sliding window prediction is using the neighboring pixels to predict the label of the centered pixel. As shown in Figure 5.5(a), in trying to predict the label around 0.05 cm pointed by the arrow, we can use all the pixels covered by the green window. This sliding window idea is nice because it converts the inverse problem into a standard classification problem in machine learning, where the input x_i is vector, whose dimension equals the window length, and the output y_i is an integer corresponding to the class label. We could label the "dermis" as class 1 and "air" as class 0 and we have our standard two-class classification problem. When there are more possible tissue types, it generalize to a multi-class classification problem by introducing more type labels.

However, this sliding window approach would fail badly because it couldn't handle the tails. From their physical origin, we know that the tails appears at the end of

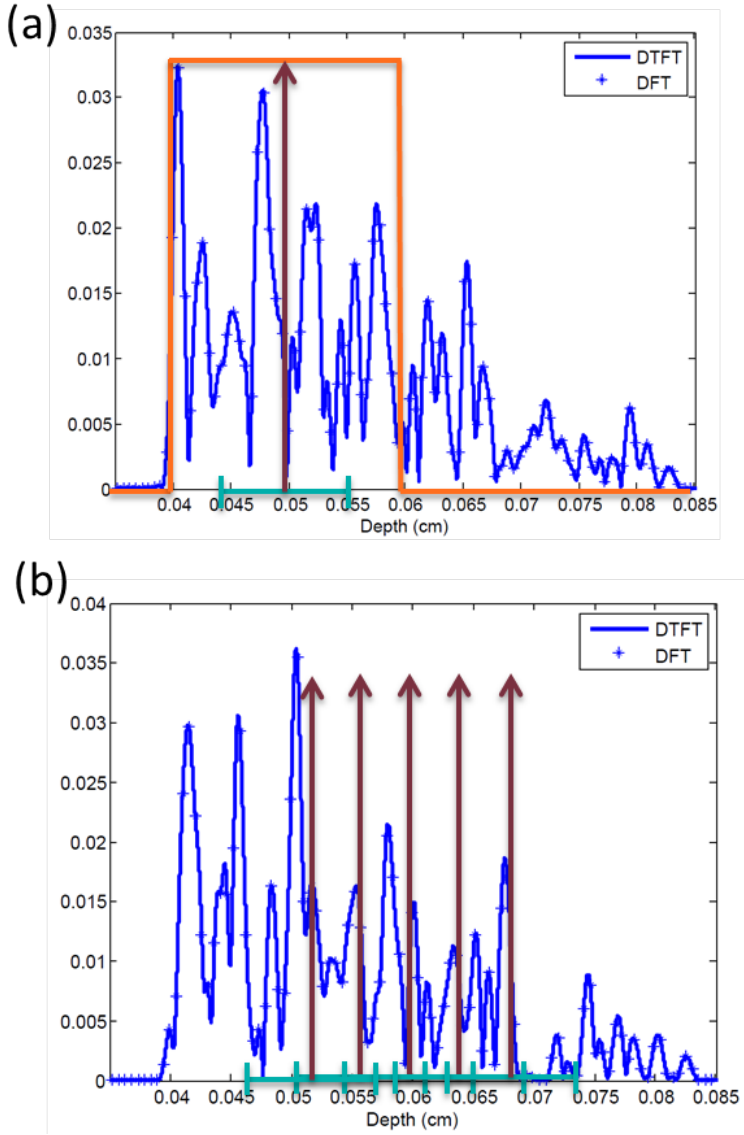


Figure 5.5. Example A-scans under the condition of flood illumination combined with the idea of sliding window prediction. (a) The resulting A-scan when the radius of the illumination source equals 0.04 cm. The orange line depicts the corresponding ground truth and the green window with an arrow illustrates the idea of sliding window prediction. (b) Schematic drawing of sliding window prediction, where the tissue type is predicted one at a time using a sliding window going from left to right.

a tissue layer since the presence of the tissue layer causes the photon paths to be prolonged and distorted, making them essentially the Class II photons. In order to

predict correctly in the tails region, the sliding window must know the existence of tissue layers prior to them which requires non-local information. In other words, the success of the sliding window approach depends on the fact that the problem is local. One might argue that the non-locality of the problem can be eased by increasing the length of the window, however, as will be shown in the next section, for multi-layered structures this non-locality extends to the entire A-scan.

5.2.4 Multiple Layered Biological Tissues

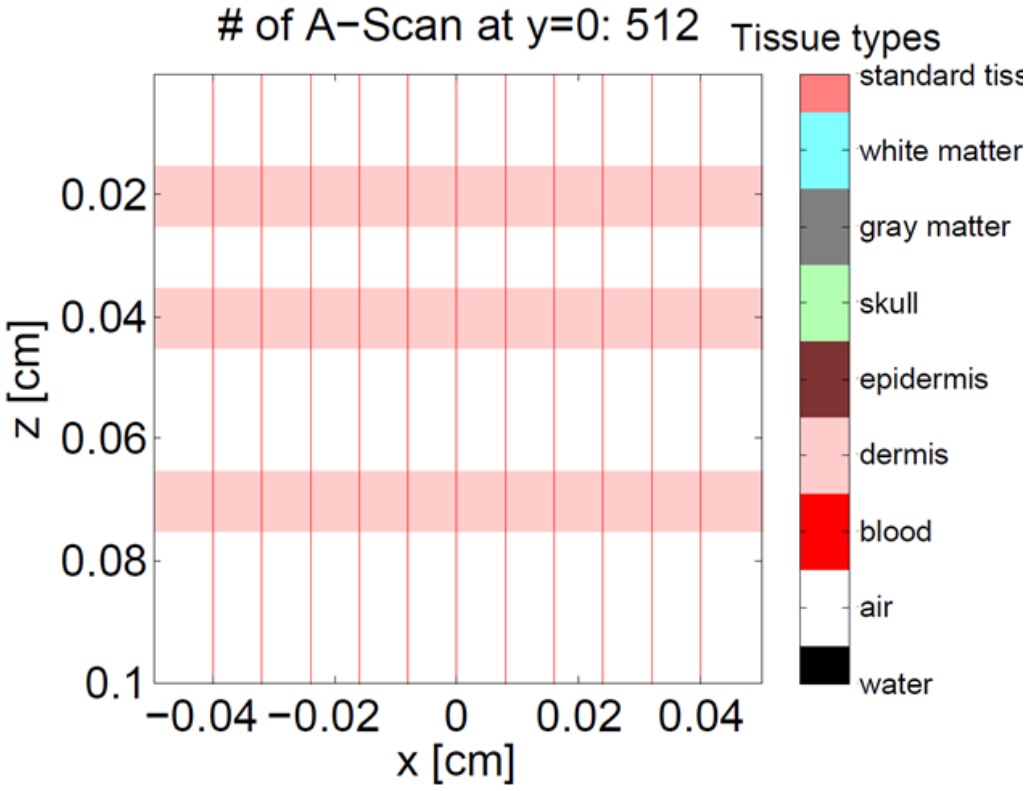


Figure 5.6. Schematic drawing of the anatomical, ground-truth structure of a multi-layered biological tissue (three layers of dermis), where the three dermis layers are centered at depth $z = 0.02$ cm, $z = 0.04$ cm, and $z = 0.07$ cm respectively, all with a thickness of 0.01 cm.

It would be interesting to extend from imaging single layered tissue to multiple layered tissues, which helps us to further understand the non-locality of the problem

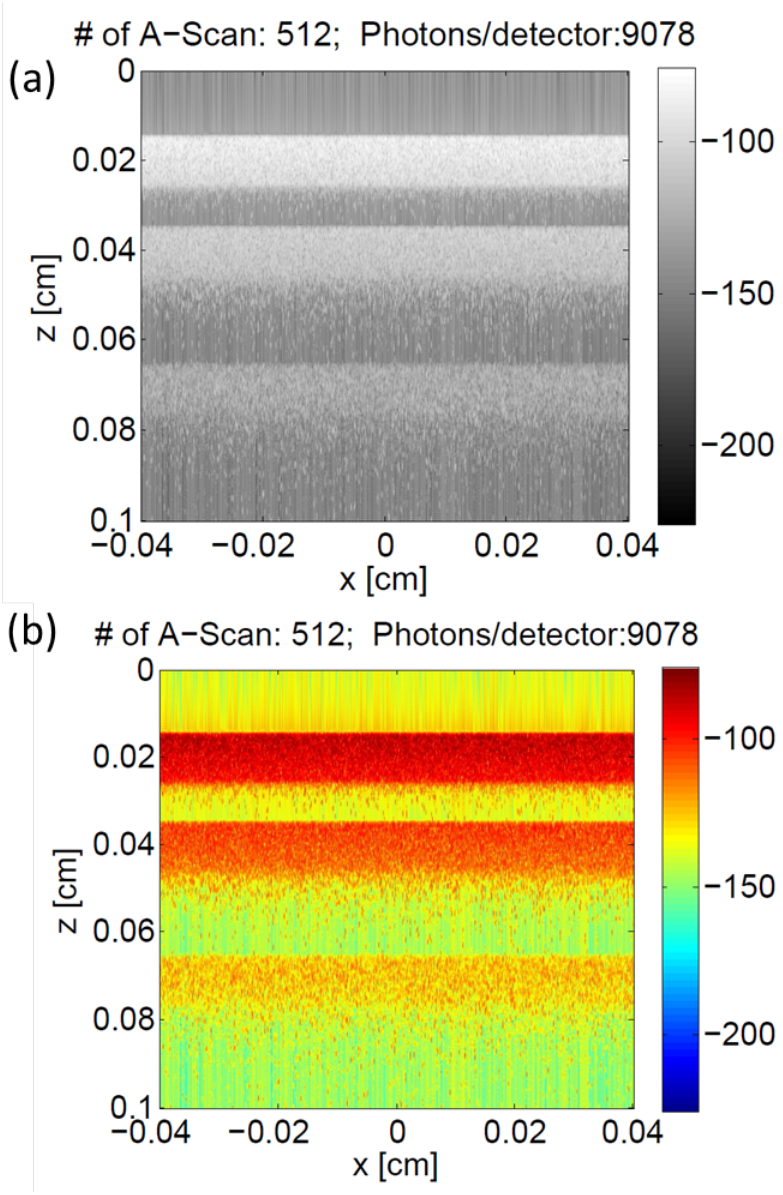


Figure 5.7. Monte Carlo simulation of a multi-layered biological tissue structure (three layers of dermis). The OCT image consists 512 A-scans and the simulation time is 32 minutes. (a)Simulated OCT image displayed in gray scale. (b) Simulated OCT image displayed in color scale.

in terms of how the presence of layers at the top affects the imaging of layers at the bottom.

In Figure 5.6 a three-layer tissue structure is depicted, where we see three layers of

dermis are centered at depth $z = 0.02$ cm, $z = 0.04$ cm, and $z = 0.07$ cm respectively, all with a thickness of 0.01 cm. It turns out that simulating such a structure takes 32 minutes, much longer than its single-layer counterpart. This is because the presence of the top layers slows down the imaging of the bottom (third) layer, as they prevents the photons from penetrating them to image the third layer. As a result, we need more photons per A-scans for the third layer in the OCT image to show up.

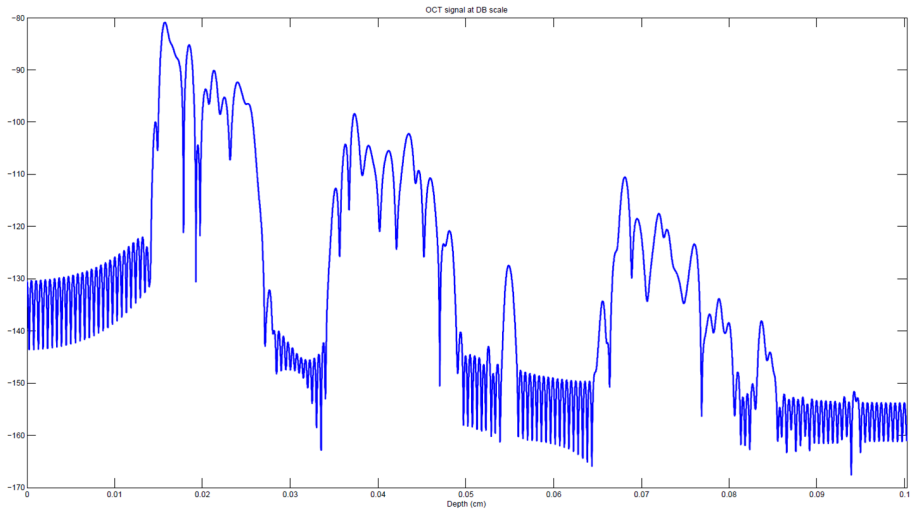


Figure 5.8. An example A-scan from the OCT image of the three-layered tissue structure, plotted at the dB scale.

In Figure 5.8 we show an example of an A-scan from the OCT image of the three-layered tissue structure. Clearly we see that even though we have three layers of dermis with identical thickness, the three corresponding peaks in the A-scan have wildly different widths. While the width of the first peak is close to the true width of the dermis layer, which is 0.01 cm, the widths of the other two peaks are distinctly greater than 0.01 cm. This is precisely the reason why OCT images are hard to interpret, as the apparent structure displayed in the image is a distorted version of the ground-truth.

Up till this point you can probably guess the answer of why the peaks in the OCT image are wider. As shown in Figure 5.9, the path length distribution of the collected photons spans an increasingly wider range as we go from the first peak, located at

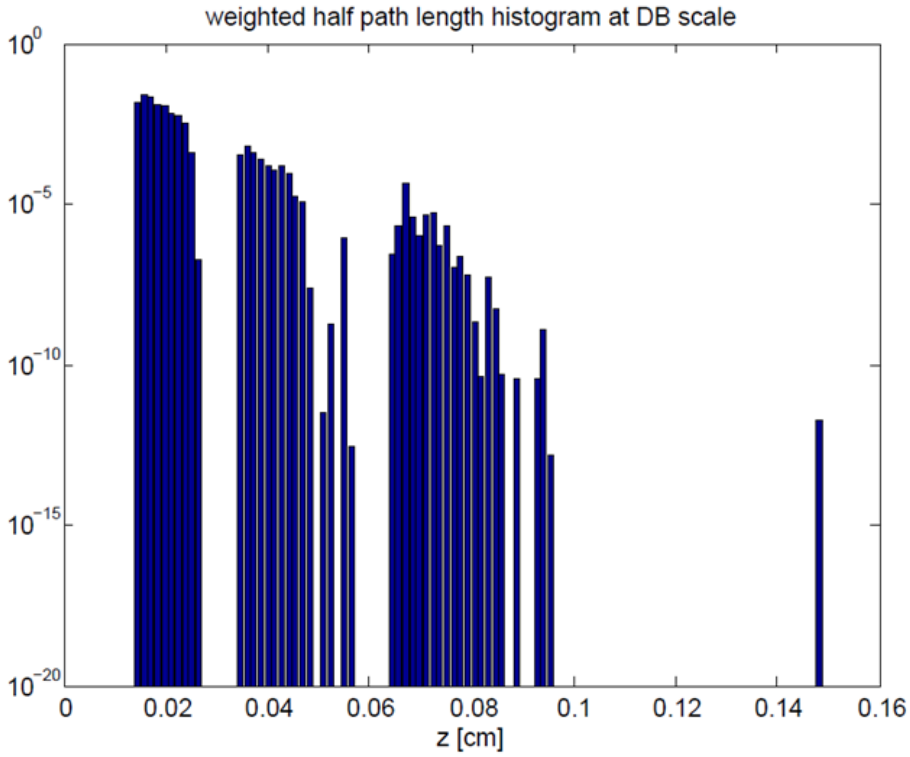


Figure 5.9. Weighted half path length histogram at dB scale for the three-layered tissue structure.

0.02 cm, to the third peak located at 0.07 cm. This increasingly wider range is a manifestation of the fact that the collected photons imaging the later layers have to penetrate and thus are scattered within the earlier layers, resulting in a wider range of path lengths. The deeper inside the tissue, the wider the path length range, which leads to more distortion.

Another challenge we can conclude is that, if you look at the magnitudes of the three peaks shown in Figure 5.9 and note that the scale is dB, the signals from the deeper regions in the tissue are much weaker. This is not necessarily a problem for machine learning algorithms, since you can always rescale the signal, but may cause significant trouble for the real-world detectors to sense them.

One last problem is the fact that, as shown in Figure 5.10, A-scans corresponding to the same underlying structure can and will be wildly different from one another. This posts another challenge to the learning algorithm as the algorithm needs to learn

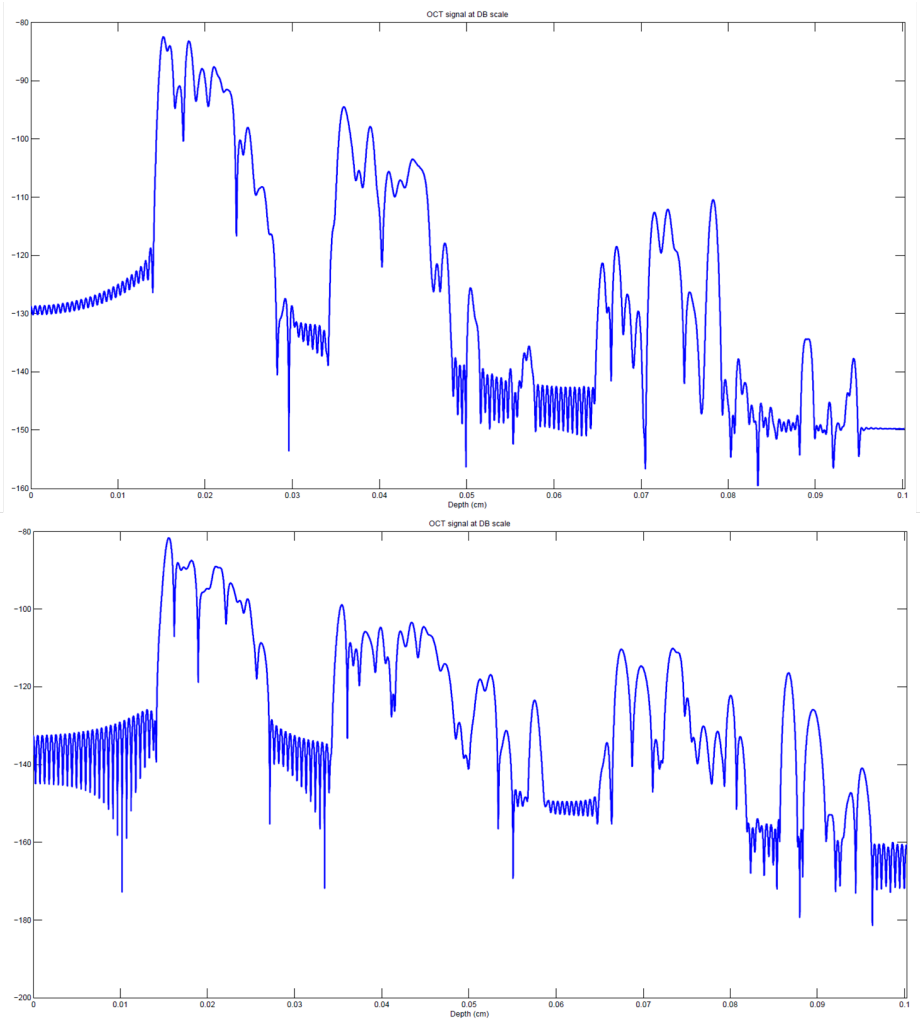


Figure 5.10. More example A-scans from the OCT image of the three-layered tissue structure, plotted at the dB scale. The fact that A-scans are different from one another even though they correspond to the same underlying structure is another challenge for the learning algorithm.

to be insensitive to this variance. We need to take this into account when designing the data set for training. For example, if the learning algorithm is experiencing a hard time trying to adapt to the variances among A-scans, we may want to include more A-scans for each OCT image.

5.3 Define the Learning Problem

Now it is time to formally introduce the learning problem, i.e., define the inverse problem from a machine learning point of view. While in Monte Carlo simulation the question being asked is "given the ground-truth structure, how will the OCT image look like", the question in machine learning is exactly the reverse, which is to reconstruct the ground-truth structure from an *unseen* OCT image. "Unseen" is the important word in the previous sentence since the machine can simply try to memorize all the previously seen data, in the form of $(image, truth)$ pairs, but this does us no good as it fails to generalize to new, unseen samples, which is the goal of machine learning.

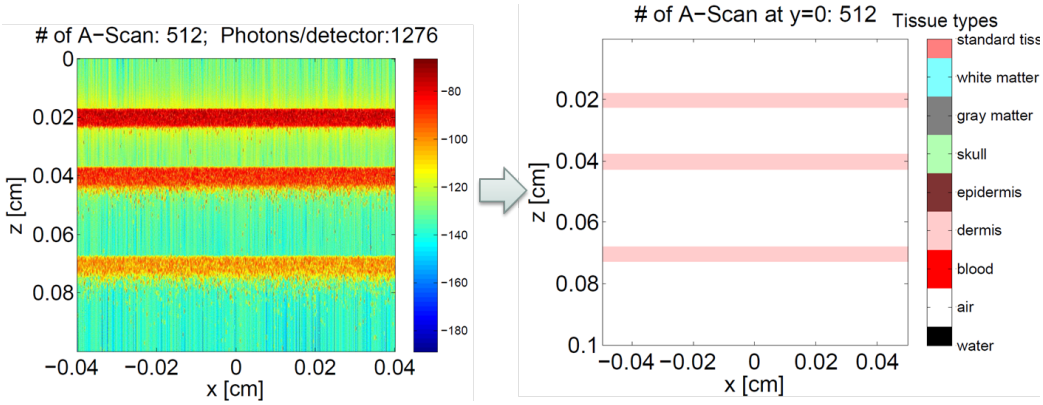


Figure 5.11. The machine learning problem, where we need to go from left to right, i.e., predicting the ground-truth structure from an unseen OCT image.

Figure 5.11 presents us the idea of learning. As the figure shows, we would like to predict the ground-truth structure (right) from an unseen OCT image (left). It is worth pointing out that from now on we set the maximum width of this kind of layered structures of dermis to be 0.005 cm, in order to generate the OCT images more quickly using our advanced Monte Carlo platform. The computer time for simulating the structure in Figure 5.11 is 4 minutes.

We decide to perform the prediction at the A-scan level first because the A-scans are the most natural way of decomposing an OCT image, as each A-scan is the signal collected at each detector. Figure 5.12 shows what the learning problem looks like on

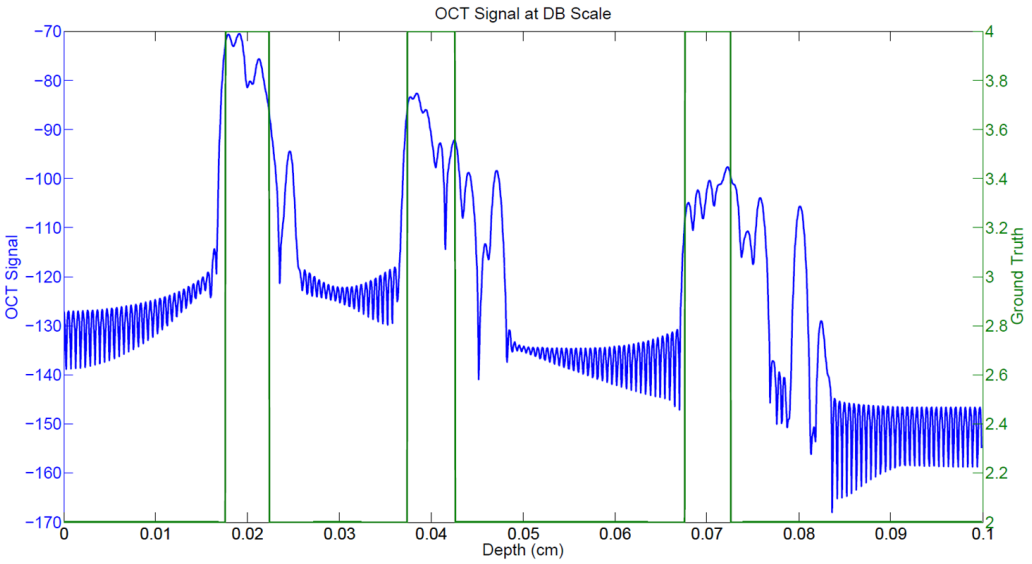


Figure 5.12. The machine learning problem at the A-scan level, where we need to predict the ground-truth structure (green line) from the transformed OCT signal, namely the A-scan (blue line).

the A-scan level, where the task is to predict the ground-truth structure (the green line) from the A-scan (the blue line).

5.4 Discussion

There are two other more fundamental reasons of choosing to predict at the A-scan level, as explained below:

- *The problem is global.* As we explained earlier when deciding against the sliding window approach, the problem is non-local. The three-layer Figure 5.12 is a perfect example to show that this problem is genuinely global. As shown in the plot, the reason why the second peak is wider than the ground-truth is due to the presence of the first peak, while the reason why the third peak is wider than its ground-truth is due to the presence of the first two peaks. Moreover, the width of the second peak is positively dependent on the width of the first peak, and similarly for the third peak. This is because as the photons penetrate thicker

tissues, their path lengths get distorted more, resulting in a wider range of path length distribution which corresponds to a wider peak in the A-scan. This means that you cannot solve the problem by predicting at a smaller scale, e.g., using something like half an A-scan as your input, since in that case you won't have enough information to perform the prediction correctly. Therefore A-scan is the smallest scale at which you can possibly solve the problem. You may elect to solve the prediction problem at a larger scale, for example, we can combine multiple A-scans and use them as the input to predict the output, which might be the ground-truth of the A-scan in the middle. This is a legitimate suggestion as we expect neighboring A-scans are informational. As we show in the next chapter, we can achieve the same effect, without the penalty of blowing up the input dimension, by performing a pooling step after prediction at the A-scan level is done.

- *The solution extends well to flood illumination.* We pointed earlier that flood-illumination is always an interest, therefore we would like to solve the non-flood-illumination version of the problem in a way that can generalize to flood illumination. Predicting at the A-scan level does just that. Under the condition of flood-illumination, the form of the problem is exactly the same except for longer tails at the end of each peaks.

In summary, we want to solve the machine learning problem at the A-scan level, which is not an arbitrary choice as there are good reasons behind this. In the machine learning language, this means that each sample is in the form (x_i, y_i) where x_i is a vector corresponding to an A-scan, and y_i is also vector corresponding the ground-truth and has the same dimension as x_i.

Chapter 6

Solving the Learning Problem

6.1 Chapter Overview

In this chapter we will solve the learning problem introduced in the previous chapter, using all sorts of machine learning techniques. We begin with some high-level discussion of the challenges for the problem as well as how we may solve it. The first step is to transform the output variable, y_i to reduce its dimension. Through this clever transformation, we arrive at a multi-output multi-class classification problem and a multi-output regression problem.

As we explore the data further, we pick up more and more insights as to how to do the job better. Our advanced Monte Carlo platform also comes in handy as it allows us to essentially generate data at will, so that we can intentionally design different data sets to train the machine learning models for better out-of-sample performance.

We then build a hierarchy architecture of machine learning models (committee of experts) based on extremely randomized trees (extra trees), and train different parts of the architecture with specifically designed data sets.

In prediction, an unseen OCT image first goes through a classification model to determine its structure (e.g., the number and the types of layers present in the image); then the image is handed to a regression model that is trained specifically for that particular structure to predict the length of the different layers and by doing so reconstruct the ground-truth of the image. We will also demonstrate in the next chapter that ideas from *Deep Learning* can be useful to further improve the performance.

6.2 First Impression and Initial Thoughts

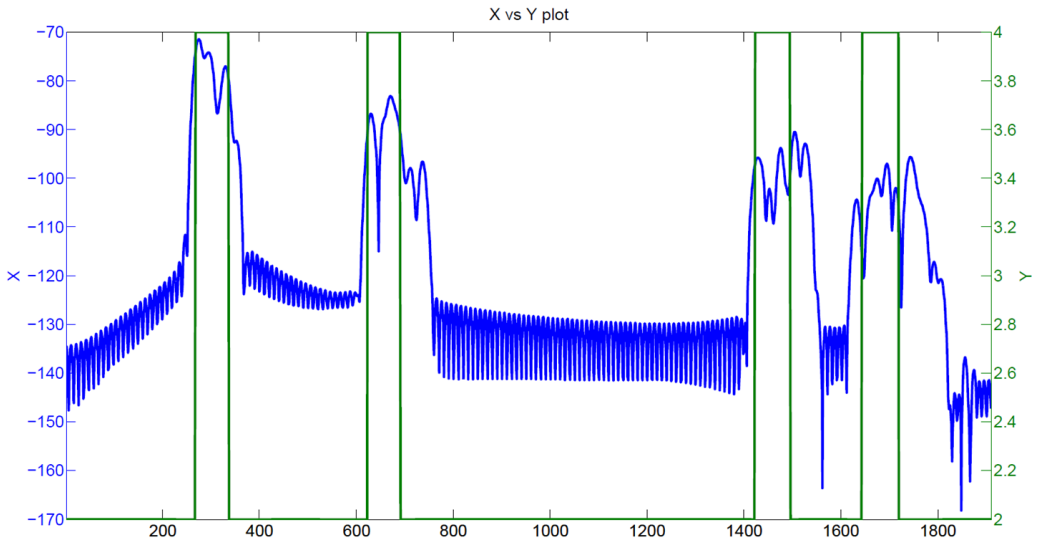


Figure 6.1. Reminder of the machine learning problem at the A-scan level, where the goal is to predict the ground-truth structure, a 1910-dimensional vector y_i(green line), from the input x_i (blue line), which is also a a 1910-dimensional vector.

We start by reminding the reader what the learning problem is. As shown in Figure 6.1, we need to use the input vector x_i, which is 1910 dimensional in our case, to predict the output y_i, which is also 1910 dimensional. As a machine learning practitioner or someone with a basic understanding of machine learning, you may start wondering in your mind the following questions.

- The first challenge that comes to mind is the fact that the dimension of y_i is too high, and predicting such a high dimensional output is simply a disaster in machine learning. However, if you think about it, the problem is not intrinsically high dimensional, especially in terms of representing y_i. You ask yourself, "do I really need as many as 1910 numbers to describe a layered structure", of course not. And pondering over this simple question leads to a clever transformation of the output variable y_i described in the next section.

- The next big question is, apparently, how do you prepare your data set. Even

though you have a powerful Monte Carlo code which is perhaps capable of generating OCT images of any object in a very efficient way, you still have to decide what data to generate. Of course, we can imagine that you would like to start with simple problems (i.e., simple structures and a small number of layers), that can be easily extended to more complex, harder problems. And more technically in OCT term, you also wonder how many images should I generate and how many A-scan should each OCT image have?

- Assume you have prepared some data sets and you could somehow magically transform the output y_i to a lower dimensional space, you still would have a multi-output problem. Then the next question is which machine learning models/paradigms would you use? Will you use the raw input x_i or transform it using some dimension reduction techniques in unsupervised learning? How would you deal with the multi-task nature of the problem? Do you prefer simple, linear models like ridge regression, or black-box models like random forests or neural networks? Perhaps you even would like to flirt with deep learning by trying ideas like unsupervised pre-training + supervised fine-tunning.

What I said above is by no means a comprehensive list but should cover most of the territory. This should give you a flavor of the type of questions to keep in mind as you start exploring a machine learning problem. We will address these questions/challenges in the next sections and we will reveal more insights as we dive deeper.

6.3 Transformation of the Output

As hinted in the previous section, the intrinsic dimension of the output variable y_i is not as high as 1910, and in fact far from it when considering realistic structures like human skins or brains. You ask yourself, "do I really need as many as 1910 numbers to describe a layered structure", of course not. How would you describe a layered biological tissue structure then? Probably you would answer by stating the type and

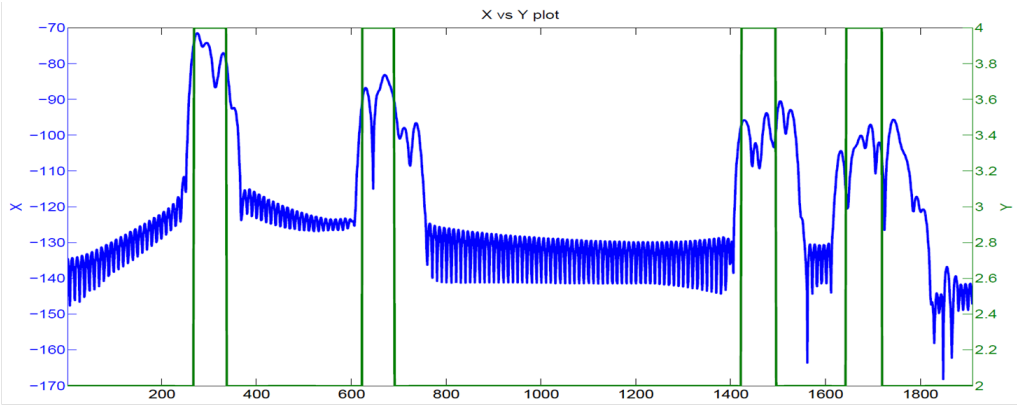


Figure 6.2. Transformation of the 1910-dimensional output variable y into two 20-dimensional variables y_{type} and y_{length}. y_{type} encodes the types of the segments of the output y into 20 class labels, and y_{length} describes the length of these segments in pixels. Here the code "2" means "air" and the code "4" means "dermis". There is no loss of information in this transformation as the new, lower dimensional variables y_{type} and y_{length} are enough to fully reconstruct the original output variable y.

length of the first layer, the second layer, and so on. You would have a short answer as there aren't many layers present in a natural, realistic tissue structure.

Based on exactly this simple idea, we transform the 1910-dimensional output variable y into two 20-dimensional variables y_{type} and y_{length}, as shown in Figure 6.2. As their names suggest, y_{type} encodes the types of the segments of the output y into 20 class labels, and y_{length} describes the length of these segments in pixels. The code "2" refers to the type "air" and the code "4" means the type "dermis".

The reader should be capable of writing a program to perform such a transformation, and the Listing 6.1 gives you an idea of how to do it. What you need to do is simply finding the locations of the discontinuities and then use them to figure out y_{type} and y_{length} respectively.

```
Y_Len = zeros(size(OCTY_all,1),N);
Y_Type = zeros(size(OCTY_all,1),N);
for ID = 1:size(OCTY_all,1)
    y = OCTY_all(ID,:);
    b = diff(y);
    c = find(b);
    d = diff([0,c,length(y)]);
    Y_Len(ID,1:length(d)) = d;
    Y_Type(ID,1:length(d)) =  y(cumsum(d));
end
```

Listing 6.1: MATLAB code for transforming the output variable y. The code simply locates the discontinuities and use the positions of them to figure out y_{type} and y_{length}.

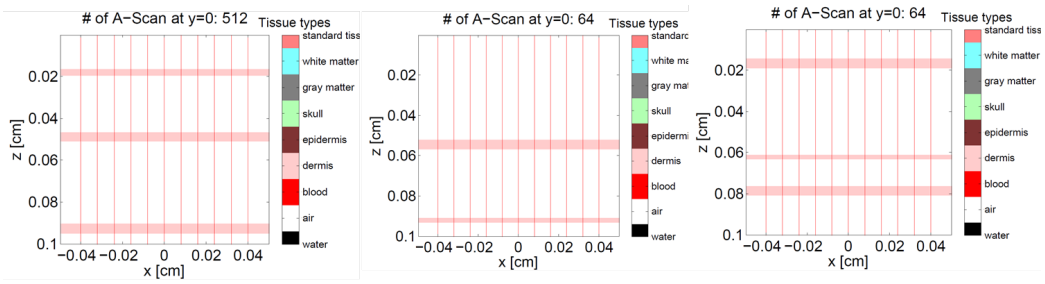
There is no loss of information in this transformation as the new, lower dimensional variables y_{type} and y_{length} have enough information to fully reconstruct the original output variable y. We choose the number 20 somewhat arbitrarily but you wouldn't expect a natural biological tissue to have more than 20 alterations of tissue types. The output variable y_i would be truly 1910-dimensional if you allow the types of the pixels to alter randomly from pixel to pixel, but this never happens in reality.

This transformation shares the same spirit of the fact that natural images are never random but lie in a lower dimensional manifold, much lower than the number of pixels present in an image.

6.4 Three Data Sets

A general principle for problem solving is to start with simple cases and work you way up to harder, more complex problems. We follow this advice and have prepared three data sets for the layered, air-dermis structure, as shown in Figure 6.3.

The first data set consists of 100 OCT images where each image is composed of 512 A-scans. The images may consist 1 to 5 layers of dermis, where the dermis layer will have a random position and a random thickness. We limit the thickness to less than 0.005 cm so that we could perform the Monte Carlo simulation within a few



Data Set No.1:
- 100 images;
- 512 A-Scans each;
- 1 to 5 layers of dermis;
- Random Thickness;
- Random Position;

Data Set No.2:
- 1000 images;
- 64 A-Scans each;
- 1 to 5 layers of dermis;
- Random Thickness;
- Random Position;

Data Set No.3:
- 1000 images;
- 64 A-Scans each;
- 3 layers of dermis;
- Random Thickness;
- Random Position;

Figure 6.3. Three data sets for the layered, air-dermis structure. The first data set consists of 100 OCT images where each image is composed of 512 A-scans. The images may consist 1 to 5 layers of dermis, where the dermis layer will have a random position and a random thickness. Data set No.2 is the same as data set No.1 except for this time there are 1000 images and each image is composed of 64 A-scans. Data set No.3 is the same as data set No.2 except that they all have a fixed, 3 layers of dermis instead of ranging from 1 to 5.

minutes. Data set No.2 is the same as data set No.1 except for this time there are 1000 images and each image is composed of 64 A-scans. Data set No.3 is the same as data set No.2 except that they all have a fixed, 3 layers of dermis instead of ranging from 1 to 5.

It is worth pointing out that even though this air-dermis structure does not sound like a realistic structure and thus does not make much biological sense, it makes the most sense from a machine learning point of view. It is the easiest problem for us to solve plus we have gained lots of insight for this type of structure in the previous chapters.

6.5 Decision Trees, Random Forest, Extra Trees

Now that we have our machine learning problem well defined, thanks to the clever transformation of the y variable, and also we have data for the problem. The natural question s to ask what kind of machine learning model(s) should we use to solve the problem at hand. In the minimum, the model should:

- be able to handle multi-output problems for both classification and regression;
- be able to handle high-dimensional input and not over-fit;
- be powerful enough to model arbitrary input-output dependence;
- be fast enough at prediction since once the models are trained they will be used in practice.

Given the above criteria, decision tree based models immediately come into mind [51]. We give below an introduction to them as they will become the main work-horse for the remaining of this chapter. If you aim to become a machine learning practitioner, tree models are a must-have in your arsenal.

6.5.1 Introduction to Decision Trees

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features.

Decision Trees are widely described in the literature [52–54], but the best way to introduce it is trough an example shown in Figure 6.4. As we can see from the plot, at each node of the decision tree, the model is trying to pick a feature and a threshold to split the data that arrives at this node so that the impurity of the data decreases in some sense after the split. The model repeats that process until the data at each node are pure, and we call those nodes leave nodes.

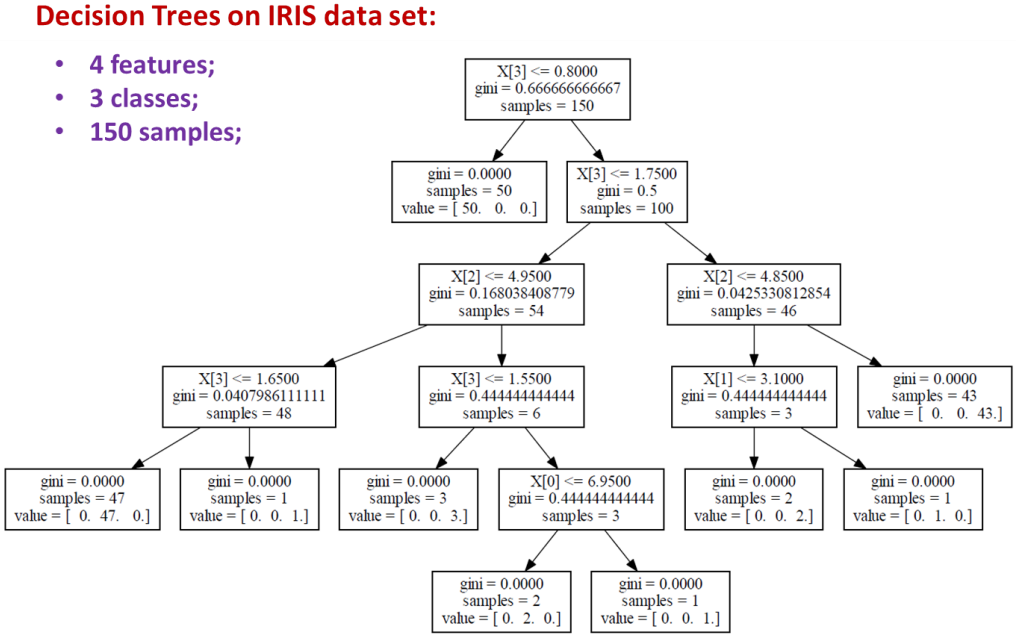


Figure 6.4. Decision Trees on IRIS data set [2], which consists of 150 samples and 3 class labels. Each sample has 4 features.

6.5.2 Mathematical Formulation of Decision Trees

Given training vectors $x_i \in R^n$, $i = 1, ..., I$ and a label vector $y \in R^l$, a decision tree recursively partitions the space such that the samples with the same labels are grouped together.

Let the data at node m be represented by Q. For each candidate split $\theta = (j, t_m)$ consisting of a feature j and threshold t_m, partition the data into $Q_{left}(\theta)$ and $Q_{right}(\theta)$ subsets:

$$Q_{left}(\theta) = \{(x, y)|x_j <= t_m\} \tag{6.1}$$

$$Q_{right}(\theta) = Q \setminus Q_{left}(\theta) \tag{6.2}$$

The impurity at m is computed using an impurity function $H()$, the choice of which depends on the task being solved (classification or regression):

$$G(Q, \theta) = \frac{n_{left}}{N_m} H(Q_{left}(\theta)) + \frac{n_{right}}{N_m} H(Q_{right}(\theta)). \tag{6.3}$$

Next we select the parameters that minimizes the impurity:

$$\theta^* = \text{argmin}_\theta \, G(Q, \theta). \tag{6.4}$$

Recurse for subsets $Q_{left}(\theta^*)$ and $Q_{right}(\theta^*)$ until the maximum allowable depth is reached, $N_m < \min_{samples}$ or $N_m = 1$.

If a target is a classification outcome taking on values $0, 1, ..., K-1$, for node m, representing a region R_m with N_m observations, let

$$p_{mk} = 1/N_m \sum_{x_i \in R_m} I(y_i = k) \tag{6.5}$$

be the proportion of class k observations in node m.

Common measures of impurity are Gini

$$H(X_m) = \sum_k p_{mk}(1 - p_{mk}) \tag{6.6}$$

Cross-Entropy

$$H(X_m) = \sum_k p_{mk} \log(p_{mk}) \tag{6.7}$$

and Misclassification

$$H(X_m) = 1 - \max(p_{mk}). \tag{6.8}$$

If the target is a continuous value, then for node m, representing a region R_m with N_m observations, a common criterion to minimize is the Mean Squared Error

$$c_m = \frac{1}{N_m} \sum_{i \in N_m} y_i, \tag{6.9}$$

$$H(X_m) = \frac{1}{N_m} \sum_{i \in N_m} (y_i - c_m)^2. \tag{6.10}$$

6.5.3 Ensemble methods

Random Forests and Extremely Randomized Trees (Extra Trees) are both examples of Ensemble Methods that support multi-output problems. The goal of ensemble methods is to combine the predictions of several base estimators built with a given learning algorithm in order to improve generalizability/robustness over a single estimator [54–57].

Two families of ensemble methods are usually distinguished:

- In averaging methods, the driving principle is to build several estimators independently and then to average their predictions. On average, the combined estimator is usually better than any of the single base estimator because its variance is reduced. Examples of this type are Bagging methods, Forests of randomized trees, etc.

- By contrast, in boosting methods, base estimators are built sequentially and one tries to reduce the bias of the combined estimator. The motivation is to combine several weak models to produce a powerful ensemble. Examples of this type are AdaBoost, Gradient Tree Boosting, etc.

The Random Forest algorithm and the Extra-Trees method are two averaging algorithms based on randomized decision trees. Both algorithms are perturb-and-combine techniques specifically designed for trees. This means a diverse set of classifiers is created by introducing randomness in the classifier construction. The prediction of the ensemble is given as the averaged prediction of the individual classifiers.

In random forests, each tree in the ensemble is built from a sample drawn with replacement (i.e., a bootstrap sample) from the training set. In addition, when splitting a node during the construction of the tree, the split that is chosen is no longer the best split among all features. Instead, the split that is picked is the best split among a random subset of the features. As a result of this randomness, the bias of the forest usually slightly increases (with respect to the bias of a single non-random tree) but, due to averaging, its variance also decreases, usually more than compensating for the increase in bias, hence yielding an overall better model. In contrast

to the original publication [55], the implementation we will use combines classifiers by averaging their probabilistic prediction, instead of letting each classifier vote for a single class.

In extremely randomized trees, randomness goes one step further in the way splits are computed. As in random forests, a random subset of candidate features is used, but instead of looking for the most discriminative thresholds, thresholds are drawn at random for each candidate feature and the best of these randomly-generated thresholds is picked as the splitting rule. This usually allows to reduce the variance of the model a bit more, at the expense of a slightly greater increase in bias.

The main parameters to adjust when using these methods is `n_estimators` and `max_features`. The former is the number of trees in the forest. The larger the better, but also the longer it will take to compute. In addition, note that results will stop getting significantly better beyond a critical number of trees. The latter is the size of the random subsets of features to consider when splitting a node. The lower the greater the reduction of variance, but also the greater the increase in bias. Empirical good default values are `max_features=n_features` for regression problems, and `max_features=sqrt(n_features)` for classification tasks (where `n_features` is the number of features in the data). Good results are often achieved when fully developing the trees. Bear in mind though that these values are usually not optimal, and might result in models that consume a lot of ram. The best parameter values should always be cross-validated.

Finally, both models also feature the parallel construction of the trees and the parallel computation of the predictions. The computations can be partitioned into k jobs, and run on k cores of the machine. Note that because of inter-process communication overhead, the speedup might not be linear (i.e., using k jobs will unfortunately not be k times as fast). Significant speedup can still be achieved though when building a large number of trees, or when building a single tree requires a fair amount of time (e.g., on large datasets).

6.6 Exploring and Shuffling the Data

With all the necessary tools in hand, we start our exploration by predicting y_{type}. We use the the zero-one error measure for this multi-output, multi-class classification problem. As shown in Figure 6.5, the zero-one error measure simply counts the number of class labels that are predicted incorrectly. In the example shown, there are 4 out of 20 class labels are predicted wrong, resulting in an error probability of 0.2.

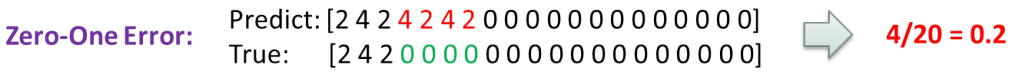


Figure 6.5. Zero-One Error for the prediction of y_{type}. In this case we get 4 class labels wrong out of 20, resulting in an error probability of 0.2.

6.6.1 Initial Exploration

Next we take a shot at data set No.1 where we have 100 images with 512 A-scans for each image, and each image is possible consisted of 1 to 5 layers of dermis. We randomly select 60 images as training set and the remaining 40 images as the test set. We fix the number of estimators to be 1000 and vary the parameter `max_features`.

Data Set No.1				
	Random Forest		Extra Trees	
`max_features`	Error Rate	Average Error	Error Rate	Average Error
`sqrt(n_features)`	0.0965	1.9305	0.0857	1.7146
16	0.0867	1.7345	0.0745	1.4894
8	0.0775	1.5506	0.0669	1.3394
4	0.0732	1.4644	0.0646	1.2927

Table 6.1: Prediction Results for y_{type} on Data Set No.1

As shown in Table 6.1, even the best model still get 1.2927 errors per sample (Average Error), which is far from satisfactory. We then tried to smooth the data using an exponential moving average, the prediction result improves to 1.2605 errors per sample, but still not good.

6.6.2 Shuffling the Data

It is worth pointing out that each image has to fully belong to either the training set or the test set, as we did in the previous section. We cannot assign, for example, half of the image (or 256 A-scans) to the training set and the other half to the test set, for obvious reasons.

However, we decide to cheat a little bit by breaking this rule and see what happens. More specifically, we shuffle all the A-scans before splitting the them into 60% training set and 40% test set, just like our previous trial. Surprisingly, we achieved zero error and 100% accuracy in prediction!

Data Set No.2				
	Extra Trees			
	Validation Set		Test Set	
`max_features`	Error Rate	Average Error	Error Rate	Average Error
1	0.032492	0.6498437		
2	0.030156	0.60312		
4	0.02903	0.580781		
8	0.02914062	0.582812		
16	0.0281562	0.563125	0.028062	0.56125
32	0.028664062	0.5732812		

Table 6.2: Prediction Results for y_{type} on Data Set No.2, which has 1000 images and 64 A-scans per image, using Extra Trees.

This tells us that the variance across different A-scans are easier to learn than variance across different images/structures. And obviously, the next step is to change the design of the data set as follows:

- Reduce the number of A-scans: $512 \rightarrow 64$;
- Increase the number of images: $100 \rightarrow 1000$.

In other words, we should try Data Set No.2. We reduce the number of A-scans for each image in order to keep the total size of the data set roughly the same. This only because the size of the data is close to the limit that we can handle on our hardware.

This time we will do it more carefully by splitting the data set into 60% training, 20% validation, and 20% test. The validation set is used to choose the best hyper

parameter `max_feature`. From now on we will only consider the Extra Trees model as it consistently out-performs the Random Forest model.

Data Set No.4				
Extra Trees				
	Validation Set		Test Set	
`max_features`	Error Rate	Average Error	Error Rate	Average Error
16	0.01906	0.20969		
32	0.01757	0.19328	0.01149	0.12641
64	0.01764	0.19406		

Table 6.3: Prediction Results for y_{type} on Data Set No.4, which has 4000 images and 16 A-scans per image, using Extra Trees.

As we see from Table 6.2, we get a huge improvement and reached about 0.5 in terms of Average Error. The obvious direction is simply to keep reducing the number of A-scans per image, and increasing the number of images. The power of the fast Monte Carlo Simulation pays dividends here as it allows us to design at generate OCT images at will. Thus we generated a new set, Data Set No.4, which has 4000 images and 16 A-scans per image.

As shown in Table 6.3, we get another improvement and we are looking at 0.12 errors per sample. We also checked the percentage of the test samples for which we predict them completely correctly, and we get a "Wrong Sample Rate" of 0.06313, in other words, we get more than 93% of the samples completely write. We can also reduce the output dimension by directly predicting the number of layers, which is an integer. And we achieved a "Wrong Sample Rate" of 0.06859, slightly worse than the multi-output prediction. One explanation is the fact that we are doing multi-task learning effectively increased the size of the data set.

6.6.3 Transfer Learning

It is also important to point out that the models we trained on one particular data set (eg., Data Set No.4) can generalize to other, unseen data sets (e.g., Data Set No.2), as indicated by the preservation of the ($> 93\%$) prediction accuracy. This is called *transfer learning*. It tells us that images formed by different number of A-scans are

of the same nature and allows us more flexibility in the design of OCT systems.

6.7 Restricted Boltzmann Machines

It turns out that we can borrow ideas from deep learning to further improve the y_{type} prediction performance. Deep Learning is a new area of Machine Learning research, which has been introduced with the objective of moving Machine Learning closer to one of its original goals: Artificial Intelligence. Deep Learning is about learning multiple levels of representation and abstraction that help to make sense of data such as images, sound, and text. Lots of deep learning models could be helpful for our case, namely:

- Multilayer perceptron;
- Deep Convolutional Network;
- Auto Encoders and Denoising Autoencoders;
- Restricted Boltzmann Machines.

We will utilize stacked Restricted Boltzmann machines to as a proof of concept that deep learning can indeed help. Restricted Boltzmann machines (RBM) are unsupervised nonlinear feature learners based on a probabilistic model. The features extracted by an RBM or a hierarchy of RBMs often give good results when fed into a linear classifier such as a linear SVM or a perceptron. The method gained popularity for initializing deep neural networks with the weights of independent RBMs. This method is known as unsupervised pre-training [58, 59].

We build a classification model that is composed of two layers of RBM with 512 components, followed by Logistic regression on top. This represents some flavor od deep learning as the first two layers are used to extract useful features from the raw, 1910-dimensional input. We apply this to Data Set No.4 and it turns out that we further improved the prediction accuracy to 97%, as shown in Table 6.4.

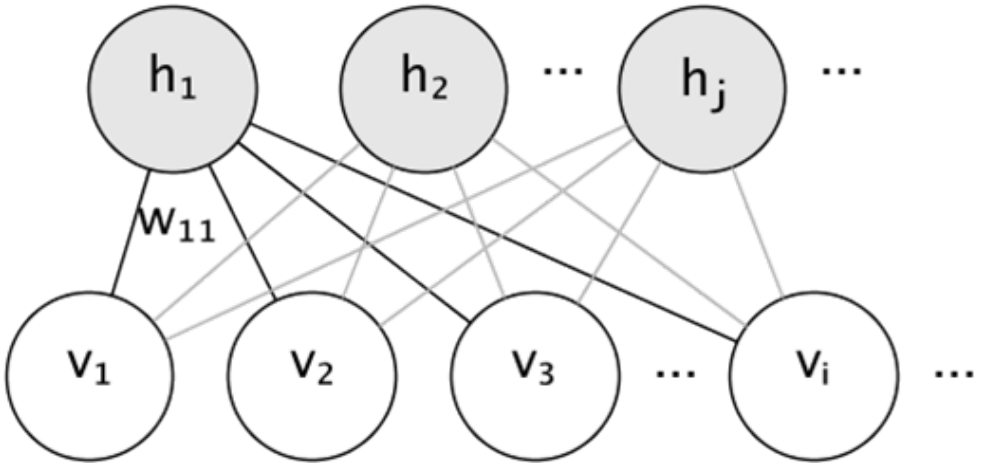


Figure 6.6. Schematic drawing of Restricted Boltzmann machines, where v_i are called visible nodes and h_j hidden nodes. The weights w_{ij} connects the two type of nodes.

Data Set No.4				
Logistic regression using RBM features				
number of layers	precision	recall	f1-score	support
3	0.99	1.00	0.99	305
5	0.98	0.97	0.97	311
7	0.94	0.96	0.95	351
9	0.95	0.94	0.95	313
avg/total	0.97	0.97	0.97	1280

Table 6.4: Prediction Results for y_{type} on Data Set No.4, which has 4000 images and 16 A-scans per image, using two layers of RBM with 512 components, followed by Logistic regression.

6.8 Hierarchical Architecture

We can continue to work on the y_{type}-prediction problem by fine-tuning the model parameters, trying more models, attempting different architecture of the models, as well as designing more data sets to further improve the prediction accuracy. It then becomes more of an engineering type pf problem and we believe the performance can be made very close to 100% accuracy. However, this only solves half of the problem as we still need to know the length of each segments besides their types. Therefore, it would be wise to switch gears and look at the other problem.

6.8.1 Predicting the Length of the Segments

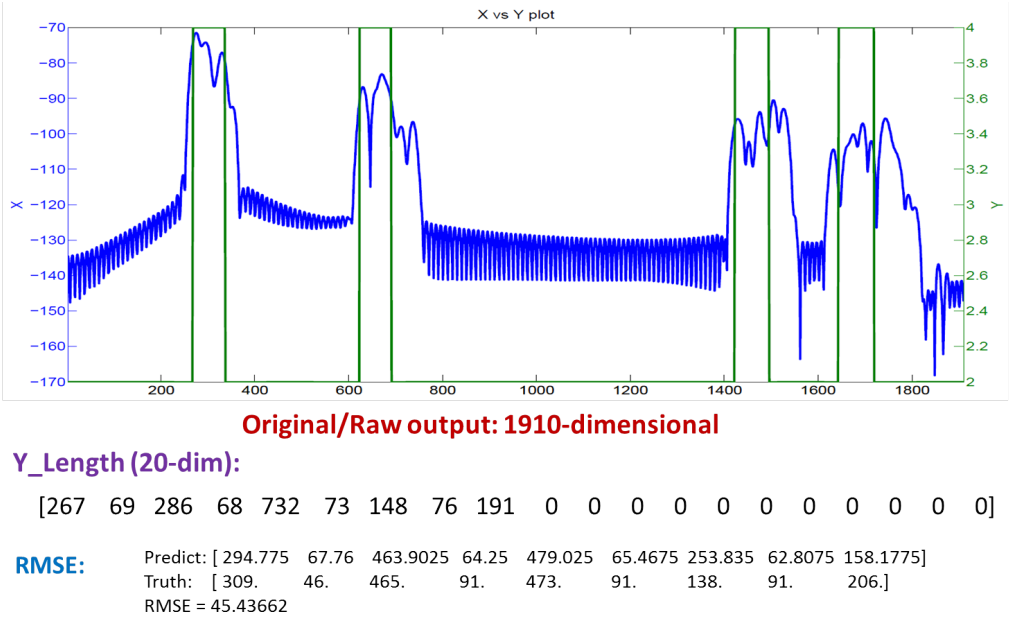


Figure 6.7. Schematic drawing of the problem of predicting the length of the segments. We use the Root Mean Squared Error (RMSE) for this task.

Data Set No.4		
	Extra Trees Regressor	
`max_features`	Average Validation RMSE	Average Test RMSE
128	63.83896	
256	63.62173	53.34265
512	64.11825	

Table 6.5: Prediction Results for y_{length} on Data Set No.4, which has 4000 images and 16 A-scans per image, using Extra Trees Regressor.

The y_{length} prediction problem is illustrated in Figure 6.7, where we use a Root Mean Squared Error (RMSE) as it is a multi-output regression problem. We will work with Data Set No.4 as it has given us the most success previously. As a reminder, Data Set No.4 has 4000 images and 16 A-scans per image, where each image may be composed of 1 to 4 layers of dermis. We split the data into 60/20/20 train/valid/test like before and use the Extra Trees Regressor as our model. We fix the number of estimators to be 400 and use the validation set to select the `max_features`. Once

we find the best value for `max_features`, we re-train the model on the combined train/valid set before we evaluate our final model using the test set.

6.8.2 Interpreting the RMSE

The prediction result is shown in Table 6.5, where we get an average RMSE of 53.34265. Here is one way to think of the RMSE. If our model is perfect and gets the length of every segment correctly, of course we should get an RMSE of zero. However, if we get every segment off by 1 pixel, note that it can either be $+1$ or -1, and since the total length of the segments are conserved (the conservation can be reinforced by re-normalization of the predictions), we should get roughly half $+1$'s and half -1's. Moreover, if we get a $+1$ and a -1 next to each other, one pixel is wrong for the two segments. Similarly, if we get two $+1$'s next to two -1's, we get two pixels wrong for the four segments. Following this line of thought, the expected number of wrong pixels of the reconstructed OCT image can be estimated as $\frac{RMSE \times n}{2}$, where n is the number of segments.

6.8.3 From Good to Bad

Good example:
[596.3525 64.44 739.685 69.5025 275.6675 43.385 99.005 6.1675 15.795]
[591. 61. 808. 73. 232. 73. 72. 0. 0.]
RMSE = 30.76857

Very good example:
[573.585 83.48 1252.935 0. 0. 0. 0. 0. 0.]
[576. 72. 1262. 0. 0. 0. 0. 0. 0.]
RMSE = 4.94185

Bad example:
[277.1125 60.7875 328.5225 63.835 511.8125 69.1275 390.4375 38.2625 170.1025]
[294. 50. 316. 76. 503. 76. 130. 69. 396.]
RMSE = 115.77440

Figure 6.8. Different test cases for the y_{length} prediction on Data Set No.4, where we see some very good examples as well as very bad examples.

A look at the different test cases for the y_{length} prediction on Data Set No.4 reveals lots of insight. As shown in Figure 6.8, there are good examples where we predicted the segment lengths close to perfect, as well as bad examples where we completely

missed it. Figuring out why there is such high variance in our prediction performance is vital for further improvement.

6.8.4 Committee of Experts

The reason why we saw high variance in the prediction is that the model was trying to do it all. It has to simultaneously handle the 1-layer, 2-layer, 3-layer and 4-layer cases and this would be a disaster since the underlying physics of these cases are fundamentally different. The reason is that in the presence of different number of layers, the photon path length distributions are perturbed in a totally different way. As a result, once the model detects a peak, for example, near end of the A-scan, it would have a hard time assigning the truth width of the corresponding layer since for different structures (in terms of the number of layers), the answer would be drastically different. More specifically, if it is a 1-layer structure, the true width of the layer would be close to the apparent width of the detected peak; on the other hand, had it been a 4-layer structure, the width of the layer would be much smaller than the apparent width of the detected peak, due to photon path length distortion.

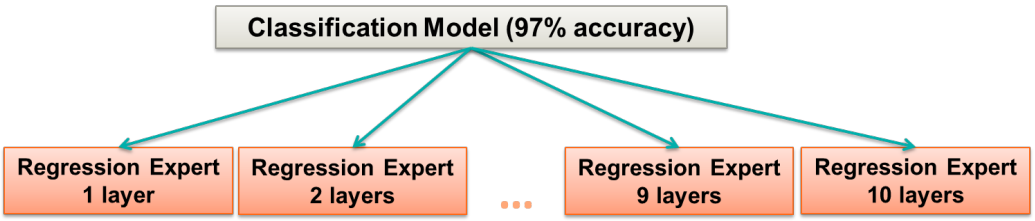


Figure 6.9. The hierarchical architecture in the form of a *committee of experts*, where the job of the Classification Model on the top layer to dictates the structure of the imaged object in terms of the number and type of the segments, and assign a Regression Expert to solve the segment length prediction problem corresponding to that particular structure.

The observation clearly calls for specialization of the prediction models and a hierarchy architecture in the form of a *committee of experts*. As shown in Figure 6.9,

the task for the Classification Model on the top layer is to dictates the structure of the object from its OCT image. Once that is done, we assign one Regression Expert from the bottom layer that is specialized in segment length prediction for that particular structure.

6.9 Prediction Results

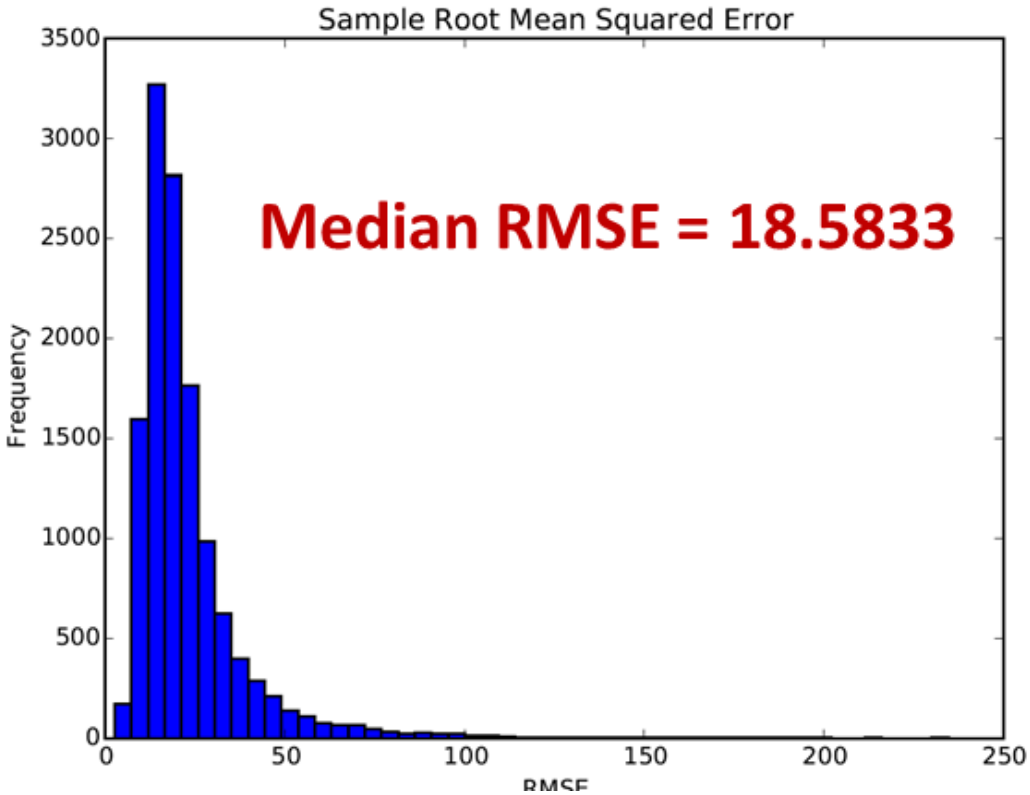


Figure 6.10. Histogram of the RMSE on Data Set No.5, where we see that the median RMSE is only 18.5833 compared to the average RMSE of 27.54875. This shows that there are a few bad examples driving the RMSE up, but they are a minority.

Of course, we need to train the different parts of the architecture using specifically designed data sets, we are able to do that thanks to our advanced Monte Carlo simulation platform. As an example, we can train the Regression Expert for the 3-layer dermis structure using data set No.3, which has 1000 images with each image

having 64 A-scans, and all the images have 3 layers of dermis. Once we do that the RMSE is reduced to 34.486 pixels. We can go ahead and design a Data Set No.5 which which has 4000 images with each image having 16 A-scans, and all the images have 3 layers of dermis. This would give us an average of RMSE at 27.54875 pixels.

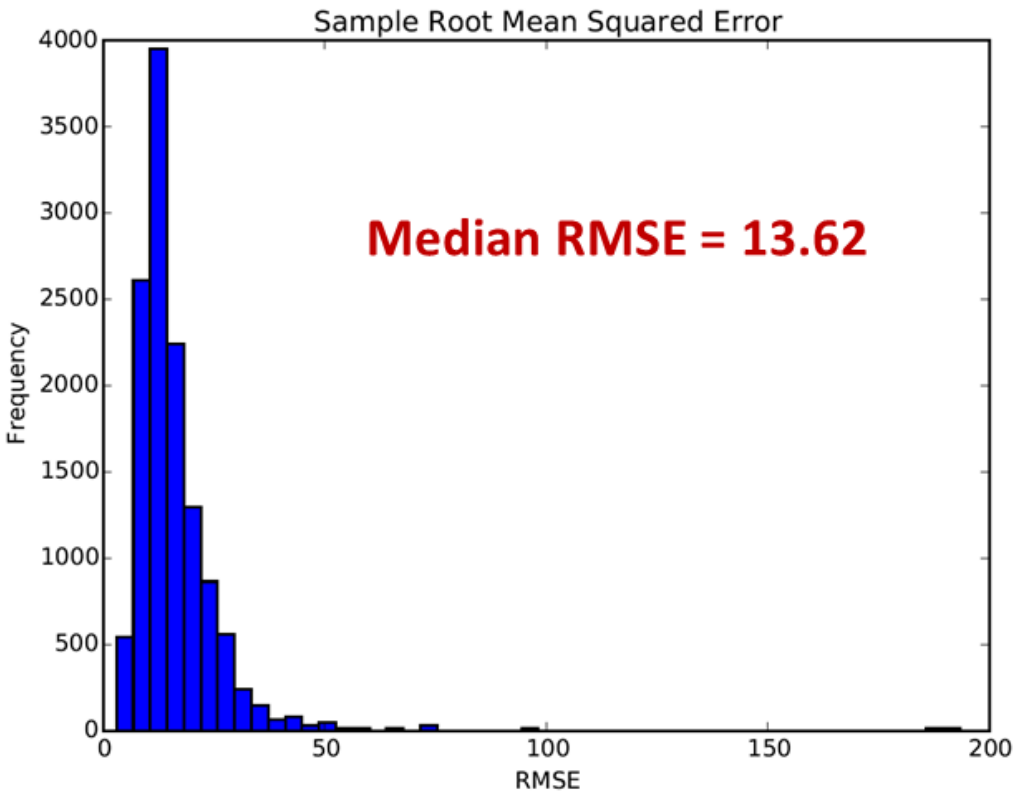


Figure 6.11. Histogram of the RMSE on newly generated Data Set No.5 without those tricky cases that have layers close to the boundary. We see that the average and median of RMSE are further reduced. However, there are still outliers present but significantly less.

If we look at the histogram of the RMSE, as shown in Figure 6.10, it tells us that the median RMSE is only 18.5833 compared to the average RMSE of 27.54875. This implies that there are a few bad, tricky examples driving the RMSE up, but they are a minority. It turns out that these tricky examples are those in which the dermis layers happen to be very close to the boundary, and these cases are rare in the population and that's why the learning algorithm, the regression expert was having

a hard time predicting them. This is expected since the performance of the learning algorithm highly depends of the distribution of the training data. For the cases that have a larger population in the training data, the learning algorithm would perform better out-of-sample, and vice versa for the less populated cases.

Once we removed those tricky cases that have layers close to the boundary, the average RMSE is further improved to 20.42273 pixels, and the median RMSE is done to 13.62 pixels, as shown in Figure 6.11. We can apply a pooling step after the prediction since we know the true structure is layered, this would give us another improvement of 5 pixels in the average RMSE. To more general structures, we could apply a local pooling since their exists a local continuity in natural biological tissues.

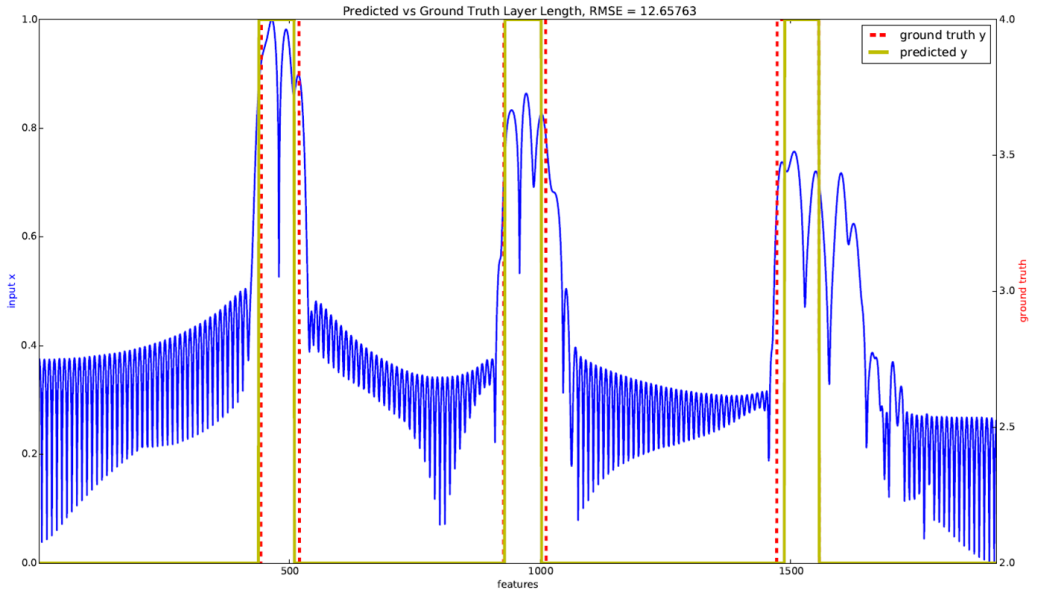


Figure 6.12. An example A-scan comparing our prediction result against the ground-truth. This represents a typical performance of our hierarchical model.

We conclude this chapter by showing three example A-scans comparing our prediction result against the ground-truth. The first two examples, as shown in Figure 6.12 and 6.13, represent a typical performance and a slightly worse-than-average performance of our hierarchical model. From the look of the two plots, we can safely claim that our model does the job close to perfect.

We also show in Figure 6.14 a third example that actually corresponds to the worst

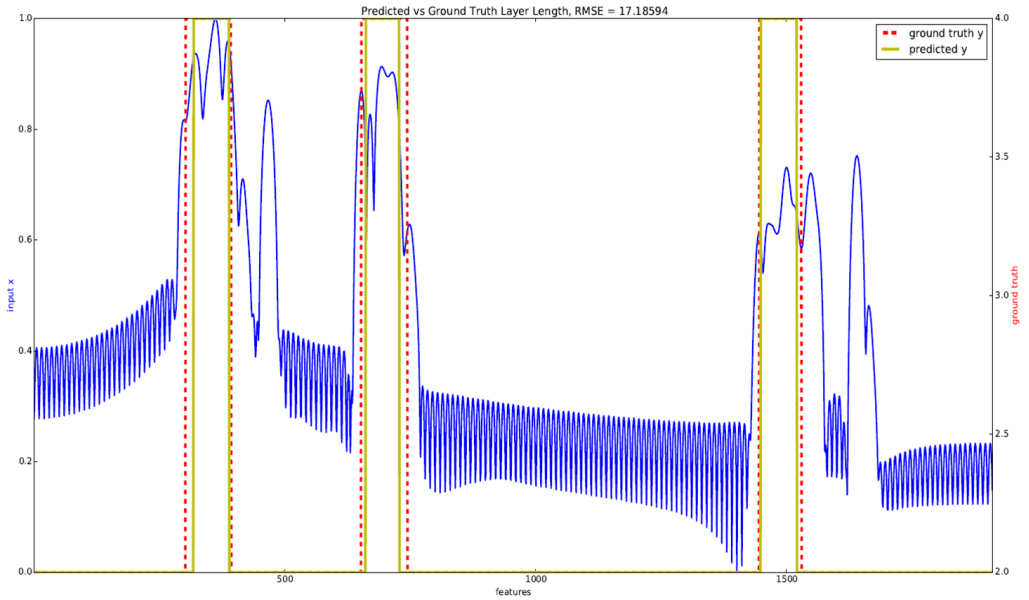


Figure 6.13. An example A-scan comparing our prediction result against the ground-truth. This represents a slightly worse-than-average performance of our hierarchical model.

performance of our model, and in fact represents the new type of outliers present in the newly generated data set. These outliers are the cases in which two layers of dermis are very close to each other. Again this corresponds a minority that drives the RMSE up. Of course we could either remove these outliers or intentionally generate more such outliers to further improve the performance of our hierarchical model. On the other hand, we can also see how our current model reacts to this tricky cases. It seems that our model was having a hard time trying to find the third layer, since it viewed the last two peaks as one (because in this data set most likely there is at most one peak near the right end of the A-scan), it has no choice but to make a guess and put the third layer in the middle, which is again the most likely case in this data set.

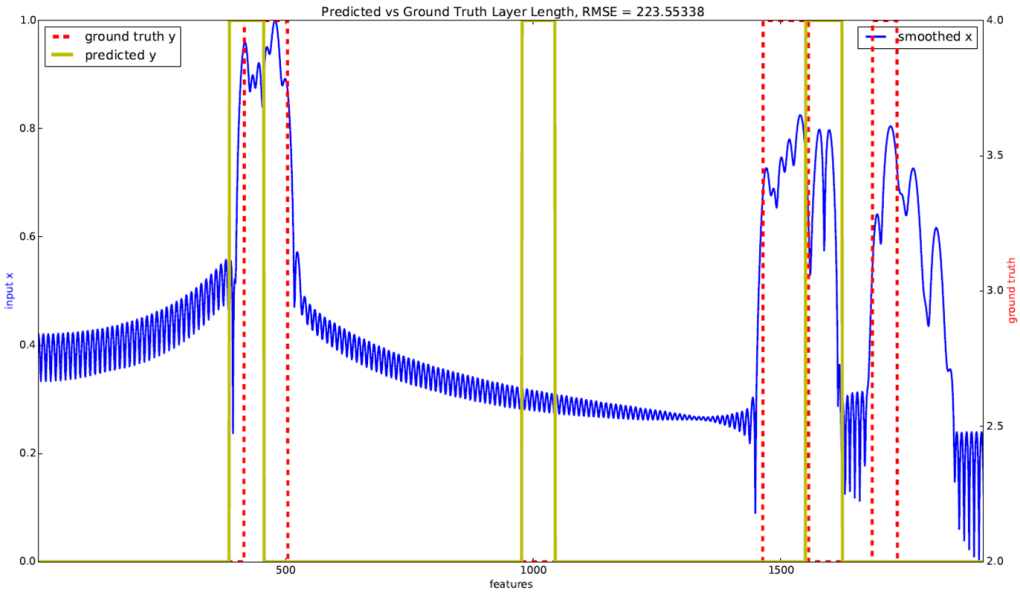


Figure 6.14. An example A-scan comparing our prediction result against the ground-truth. This actually corresponds to the worst performance of our model, and in fact represents the new type of outliers present in the newly generated data set. These outliers are the cases in which two layers of dermis are very close to each other. Again this corresponds a minority that drives the RMSE up.

Chapter 7

Conclusion and Outlooks

7.1 Chapter Overview

In this chapter we will first repeat what we have done in Chapter 6 to see if our hierarchical model is able to generalize to more complex, realistic problems. We will use the brain structure as our example, which is way more complicated than the air-dermis structure we have previously conquered. We then move on to summarize this thesis and discuss about the new philosophy that is made possible by this thesis when it comes to OCT imaging. We conclude this thesis by taking a tour of future research opportunities in the outlook section.

7.2 More Complex Structures

Conquering the air-dermis structure may not sound as convincing if the techniques could not generalize to more complex, real-world structures. In this section we address this concern by simulating and predicting the brain structure.

7.2.1 Simulating the Brain Structure

We start by drawing and simulating the brain structure with our advanced Monte Carlo platform, as shown in Figures 7.1 and 7.2, respectively. This simulation takes 8 minutes, which is understandably longer than the 1-minute simulation time of the

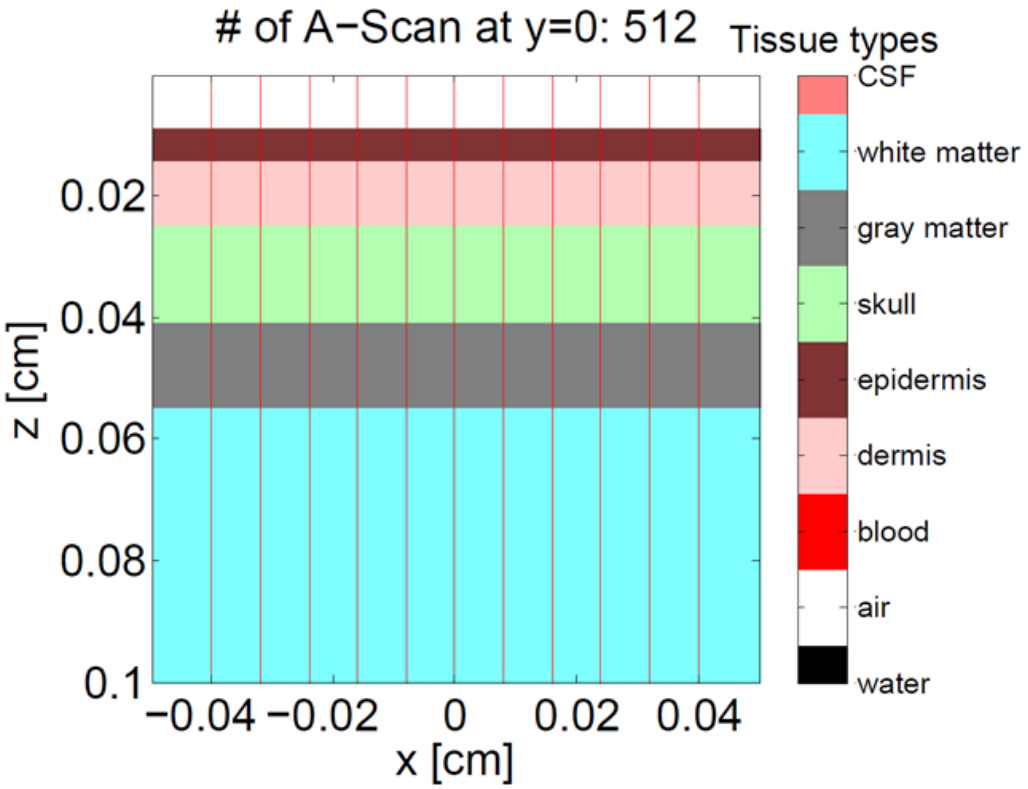


Figure 7.1. Schematic drawing of the anatomical, ground-truth structure of a multi-layered brain, which consists of an epidermis layer, a dermis layer, a skull layer, and both white and gray matters.

air-dermis structures. It demonstrates that complicated, real-world structures can be successfully simulated just like before.

7.2.2 Predicting the Brain Structure

We repeat the process discussed in Chapter 6 by training the regression expert on simulated Brain structures. We prepared a data set for training the same way as the Data Set No.5 in the previous chapter, which has 4000 images and 16 A-scans per image. We train our model (the regression expert) on this entire data set, and then we predict on newly generated images that have 512 A-scans. The prediction at the A-scan level is shown in Figure 7.3.

We then go ahead and stitch the A-scan level predictions to form a full, predicted

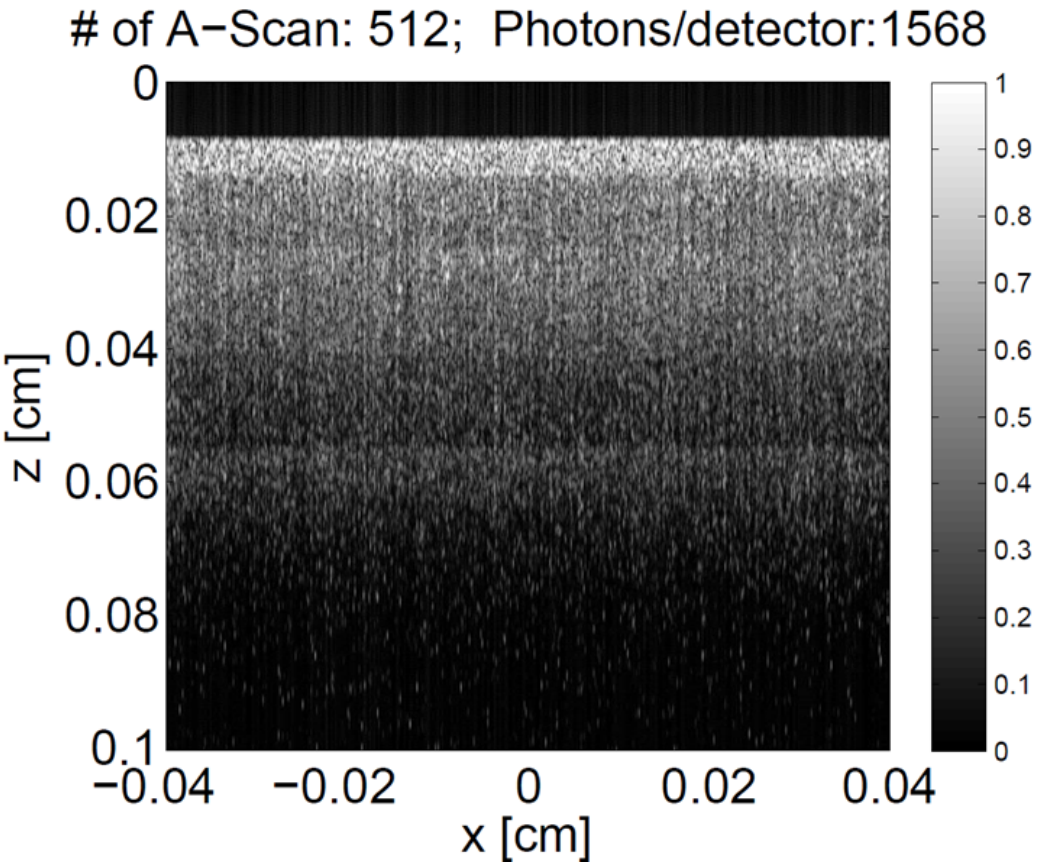


Figure 7.2. Simulated OCT images corresponding to the multi-layered brain structure, which consists of an epidermis layer, a dermis layer, a skull layer, and both white and gray matters.

ground-truth structure. And here is the story line: you get a conventional OCT image (Figure 7.4(a)), which is what you see. However, what you would like to see is the ground-truth structure (Figure 7.4(b)). Apparently there is a huge gap between the two since you could not tell whether the image is about brain or skin, as they are both layered structures. We then come to help and hand you Figure 7.5, which is what you get from our machine learning model. In our case, what you see is (almost) what you'll get.

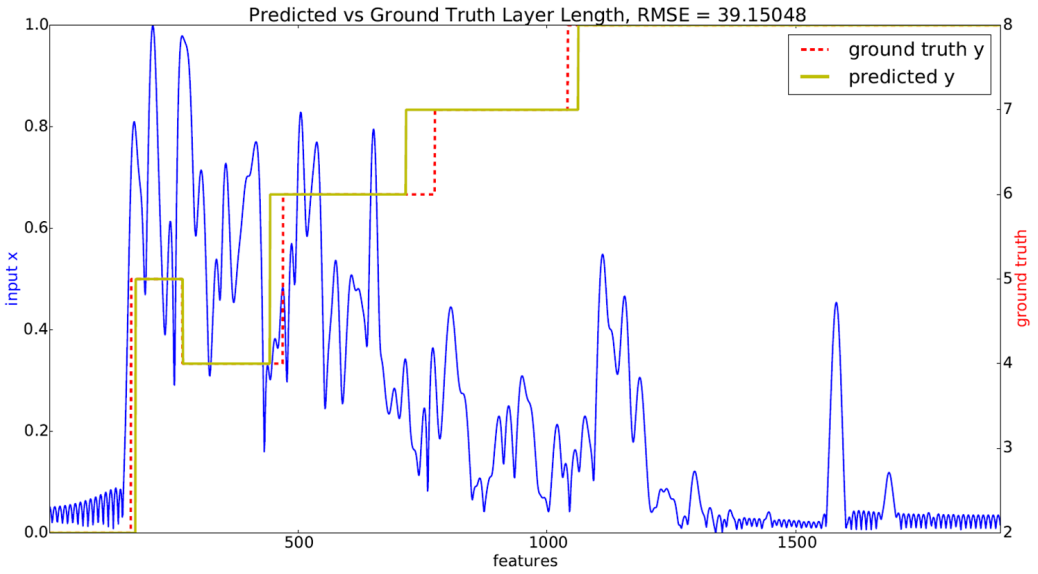


Figure 7.3. An example A-scan comparing our prediction result against the ground-truth, for the brain structure. This represents a typical performance of our hierarchical model.

7.2.3 The Curved Brain Structure

We also simulated a curved brain structure and performed the same prediction. The prediction performance is similar to the previous multi-layered case with an average RMSE of around 40 pixels.

7.3 Conclusion of the Thesis

Optical Coherence Tomography(OCT) is a popular, rapidly growing imaging technique due to its noninvasive nature, but not without its own challenges and limitations. In summary, obtaining an OCT image is not cheap, which takes either an experimental trial or expensive computer simulation. Interpreting an OCT image is also hard, i.e., we cannot recover the anatomical structure of the imaged object. This is due to the fact that the physics underlying the image formation process distorts and weakens the signal, as well as the lack of fully annotated OCT images for us to learn from. Moreover, because of the multiple scattering events that the photons

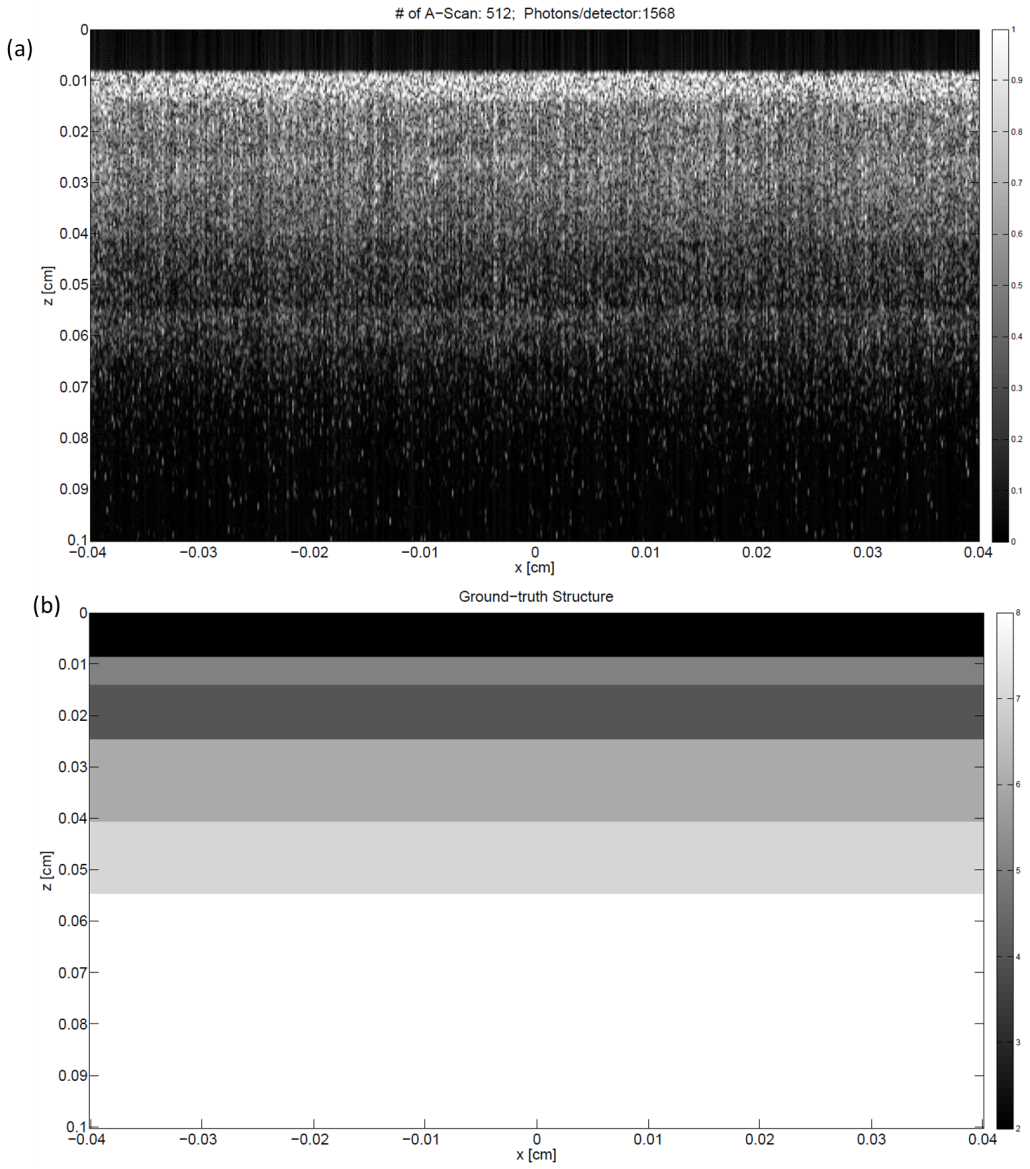


Figure 7.4. Part One of the story line. (a) An OCT image of the brain, this is what you see. (b) The ground-truth anatomical structure of the brain, this is what you would like to see.

need to go through, deeper regions of a biological tissue (beyond a few millimeters) are hard for the photons to reach, resulting in a limited imaging depth of OCT.

This thesis addresses the challenges described in the previous section by the successful development of an advanced Monte Carlo simulation platform, which is 10,000 times faster than the state-of-the-art simulator in the literature, bringing down the

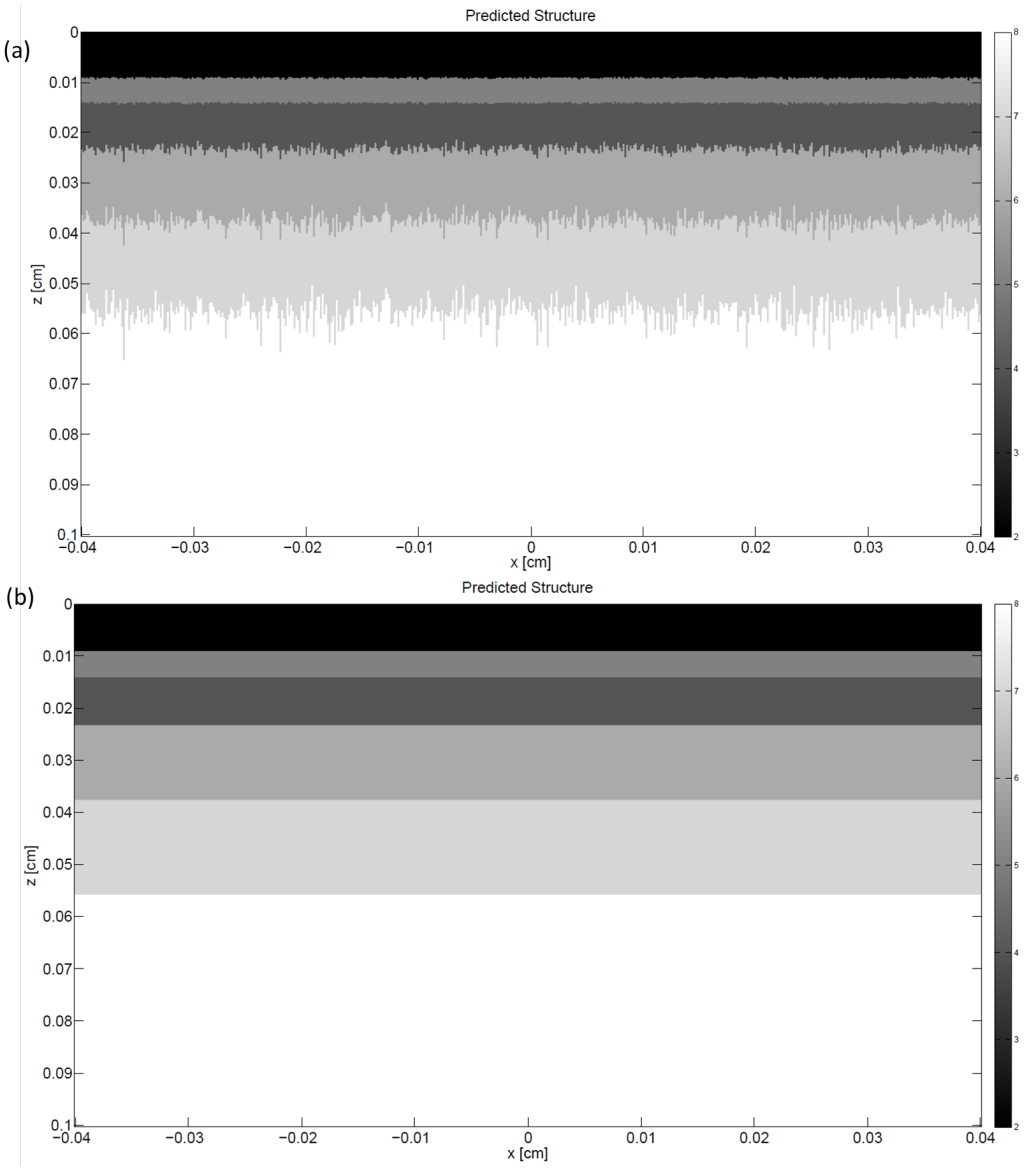


Figure 7.5. Part Two of the story line. This is what you get from our machine learning model. In our case, what you see is (almost) what you'll get. (a) The predicted ground-truth structure before pooling. (b)The predicted ground-truth structure after pooling.

simulation time of an OCT image from 360 hours to a single minute. This is one of the key contributions of this thesis, as this powerful simulation tool not only enables us to efficiently generate as many OCT images as we want, but it also provides us the underlying ground-truth of the simulated images at the same time because we

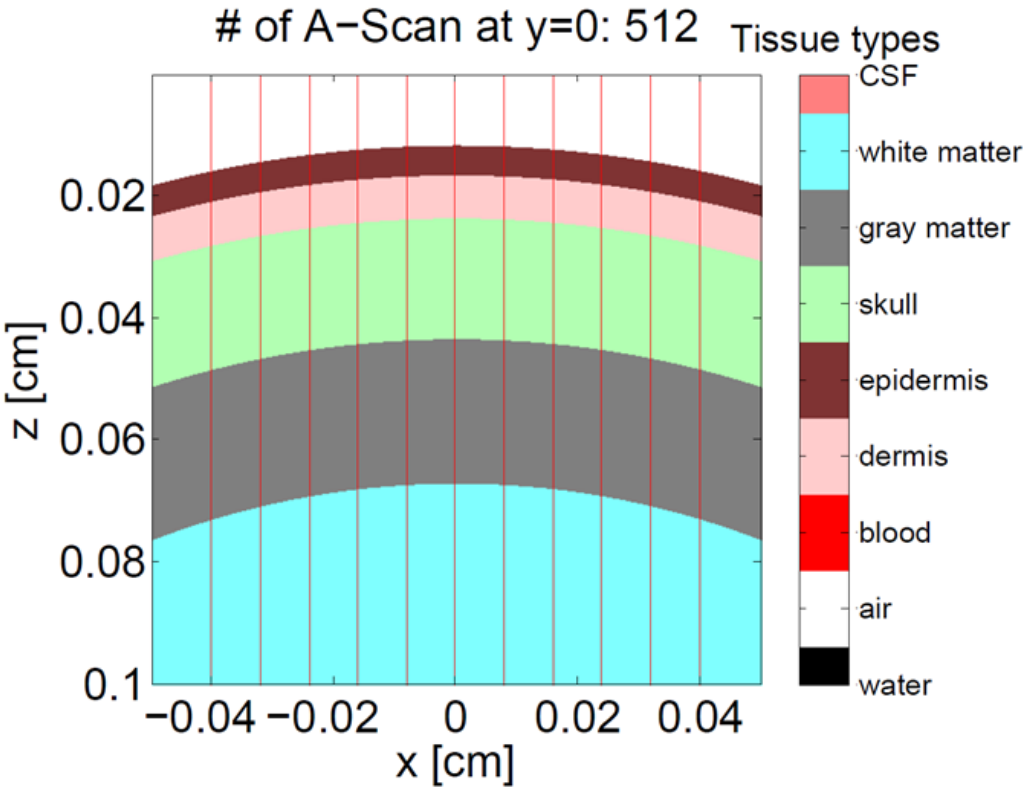


Figure 7.6. Schematic drawing of the anatomical, ground-truth structure of a curved brain, which consists of an epidermis layer, a dermis layer, a skull layer, and both white and gray matters.

dictate them at the beginning of the simulation.

What allowed us to build such a powerful simulation tool includes a thorough understanding of the signal formation process, of both time-and frequency-domain OCT, the correct implementation of the importance sampling/photon splitting procedure to FD-OCT systems, a clever and efficient use of the voxel-based mesh system in determining photon-mesh interception, and a parallel computation of different A-scans that consist a full OCT image, as well as knowing whether we have collected enough photons to stop the simulation.

Next we aimed at solving the inverse problem, which is given an OCT image, predict/reconstruct its ground-truth structure on a pixel level. By solving this problem we would be able to interpret an OCT image completely and precisely without

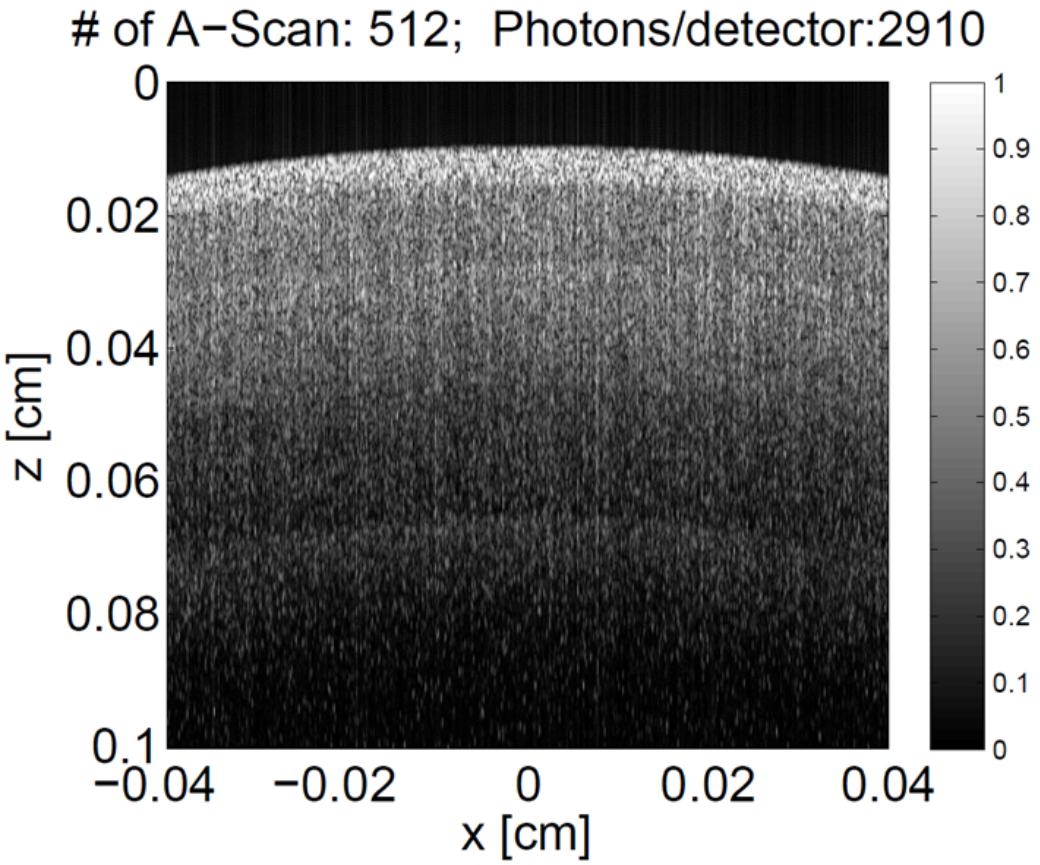


Figure 7.7. Simulated OCT images corresponding to the curved brain structure, which consists of an epidermis layer, a dermis layer, a skull layer, and both white and gray matters.

the help of a trained expert. It turns out that we can do much better. For simple structures we are able to reconstruct the ground-truth of an OCT image more than 98% correctly, and for more complicated structures (e.g., a multi-layered brain) we are looking at 93%. In other words, we managed to beat the hypothetical expert by a large margin. We achieved this through extensive uses of *Machine Learning*. The success of the Monte Carlo simulation already puts us in a great position by providing us lots of data (effectively unlimited), in the form of $(image, truth)$ pairs. These are the training examples for us or an machine learning algorithm to learn from.

The machine learning problem is not easy though, since given an OCT image, say with 512×512 pixels, we need to predict the same amount (which is 512×512) of

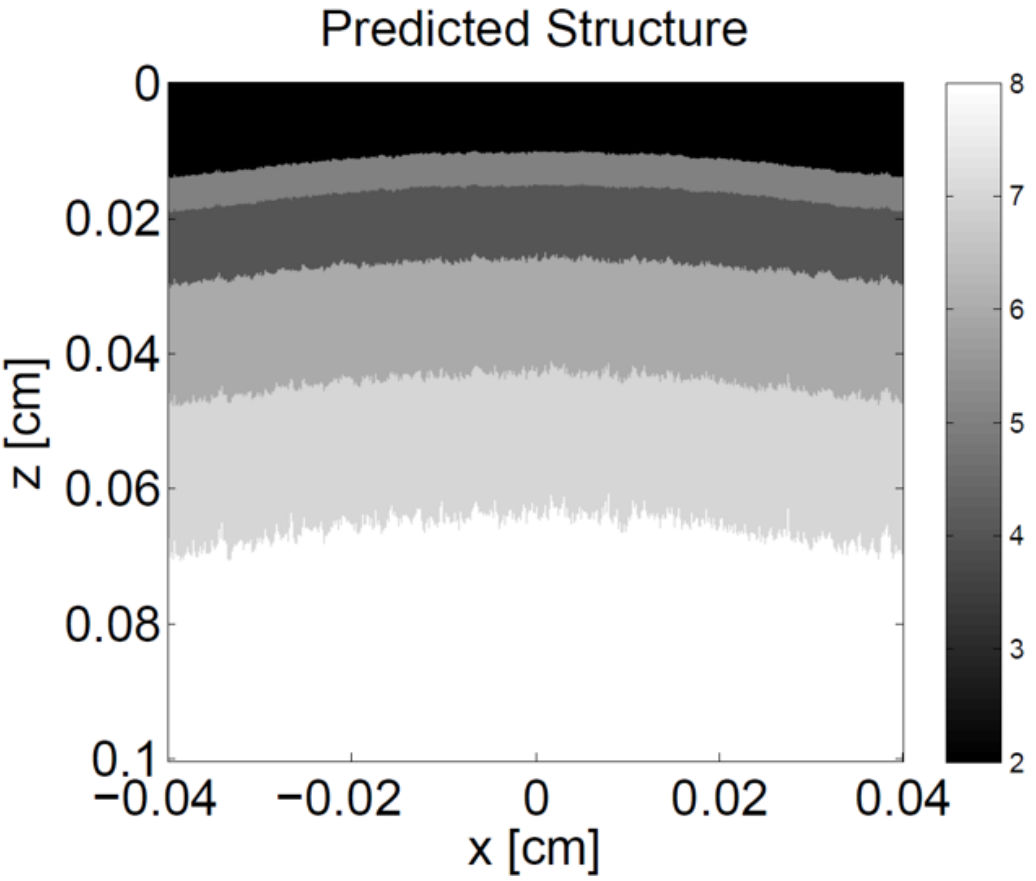


Figure 7.8. Predicted ground-truth structure (without pooling) for the curved brain example.

pixels in order to fully reconstruct the ground truth. In a typical computer vision problem, we are only required to predict a single output, say a classification label given an image. So it seems that our problem at hand is orders of magnitude harder. The good news is that we have effectively unlimited data to learn from, plus the data will be fully annotated on the pixel level, and more importantly, we have full control in terms of designing and generating the data set which we think will help to train machine learning models better. Clearly, it is the fast Monte Carlo simulation platform developed in this thesis that made such a scheme possible, since prior to us, simulating a single OCT images would take days or even weeks.

The prediction is done at the A-scan level. In our setup, each A-scan consists of 1910 points (pixels). In other words, the machine learning model would take a

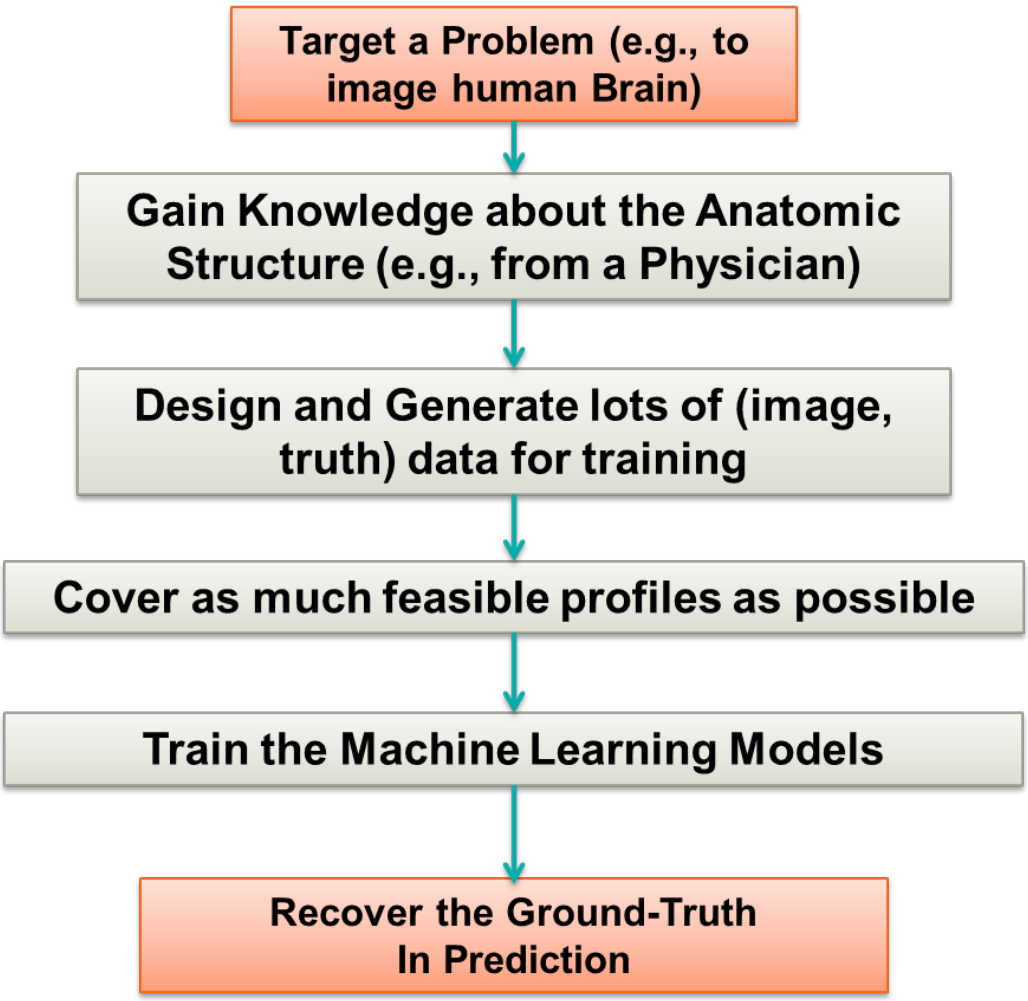


Figure 7.9. The New Philosophy for OCT.

1910-dimensional input, and need to produce a 1910-dimensional output. The first thing that comes into mind is dimension reduction, i.e., it is necessary to transform of the high-dimensional response variable into a smaller dimension, so that the problem is manageable. By recognizing that each realistic biological tissue displays a layered structure at the A-scan level, we convert the learning task into a multi-output multi-class classification problem and a multi-output regression problem through a clever transformation of the output variable. The goal of the multi-output multi-class classification problem is to predict the order and the types of the layers that the A-scan consists, while the goal multi-output regression problem is simply to predict the length of each layers. This is another key step towards solving the inverse problem,

since it reduces the dimension of the output from 1910 to less than 40. We then build a hierarchy architecture of machine learning models (a committee of experts) and train different parts of the architecture with specifically designed data sets. Again, we are able to do this by virtue of the fast Monte Carlo platform we developed. In prediction, an unseen OCT image is decomposed into A-scans, and each A-scan first goes through a classification model to determine its structure (e.g., the number and the types of layers present in the A-scan), and then it is handed to a regression model that is trained specifically for that particular structure to predict the length of the different layers and by doing so reconstruct the ground-truth of the A-scan. In the end we conduct a pooling step by averaging the prediction of adjacent A-scans before constructing our final answer for the ground-truth. This pooling step is based on the prior knowledge that there is a local continuity in the structure of a natural biological tissue, and it turns out that this gives us another boost in the prediction performance. We also demonstrate in the end of the thesis that ideas from *Deep Learning* can be useful to further improve the system.

It is worth pointing out that solving the inverse problem automatically improves the imaging depth, since previously the lower half of an OCT image (i.e., greater depth) can be hardly seen (as shown in Figure 1.2) but now can be fully resolved. Interestingly, although OCT signals coming from the lower half of the image are weak, messy, and uninterpretable to human eyes, they nevertheless carry enough information which, when fed into a well-trained machine learning model spits out precisely the true structure of the object being imaged. This is just another case where Artificial Intelligence (AI) outperforms humans. Clearly, with the development of computational power as well as advances in machine learning and AI, OCT systems will continue to become better. It is not hard to imagine that in the near future an OCT imaging system would present us the anatomical structure outright and possibly beyond a few millimeters. Hopefully this thesis represents an important first step towards this goal. To the best knowledge of the author, this thesis is not only a success but also the first attempt to reconstruct an OCT image at a pixel level. To even give a try on this kind of tasks, it would require fully annotated OCT images and

a lot of them (hundreds or even thousands), and this is clearly impossible without a powerful simulation tool like the one developed in this thesis.

This thesis has introduced a new philosophy when it comes to OCT imaging, as shown in Figure 7.9. First we target a particular imaging problem, e.g., to image human brains. We then go out and gain as much as knowledge about the anatomic structure (e.g., from a Physician, as well as the scientists that measures the optical properties of the relevant tissue types). Next we design and generate lots of $(image, truth)$ data for training, using own gained knowledge and our advanced Monte Carlo simulation platform, trying to cover as much feasible profiles as possible. We then train our machine learning models with the generated data and we are done. Then, once a new patient comes in with the target problem, we could give the patient directly the ground-truth, anatomical structure instead of a hard-to-interpret conventional OCT image.

7.4 Outlook

There is a lot of things we can do to further improve the learning models. For example, as suggested near the end of Chapter 6, there is a great chance that ideas from deep learning are likely to help. We already know that the output variable y_i lies in a much lower dimensional manifold, and it would be exciting to find out whether the input x_i, the A-scans, follow suit. Learning a sensible, lower dimensional representation of the A-scans would fundamentally improve our understanding of OCT signals.

We could also extend our Monte Carlo simulation and machine learning prediction to the flood-illumination case, which is a straightforward generalization within our framework. Flood illumination is always an interest in practice, since it would allow us to obtain an OCT image in one shot.

Also, since we have lots of data, performing all sorts of dictionary learning algorithms on the OCT signal prior to the FFT could possibly lead to the discovery of more efficient, data-driven sparse coding dictionaries that can be used for compressed sensing. Once the compressed sensing works, we could go a step further to conduct

machine learning in the compressed domain.

In short, the problems explored in thesis stand right at the intersections of the fields of Monte Carlo simulation, OCT Imaging, Machine Learning, Big Data, Computer Vision, Compressed Sensing, and Deep Learning. A breakthrough in one or a few of the above fields inevitably would lead to a breakthrough in others. And we are just getting started.

Appendix A

Implementation Details of the Monte Carlo Engine

A.1 Overview

Here we present some of the notable code listings that leads to the one-minute simulation. The object geometry is defined and the mesh files are prepared using MATLAB, while the core Monte Carlo engine is written in standard C. Once the Monte Carlo simulation is finished, the path lengths, weights, likelihood ratios, detector ID, and the maximum depth reached of the collected photon packets are saved. This saved data are then processed (i.e., compute the OCT signal and generate the A-scans via Fast Fourier Transform) to generate the OCT image.

A.2 Coding in MATLAB

In the two code listings below we show how to create the mesh for the tissue structure as well as how to process the collected photons to generate A-scans.

```
% CREATE TISSUE STRUCTURE T(y,x,z)
%   Create T(y,x,z) by specifying a tissue type (an integer)
%   for each voxel in T.
%   Note: one need not use every tissue type in the tissue list.
%   The tissue list is a library of possible tissue types.
T = double(zeros(Ny,Nx,Nz));
```

```
T = T + 2;      % fill background

zb1 = 250e-4;  % position of blood vessel 2
zb2 = 200e-4;  % position of blood vessel 2
rb1 = 150e-4;
rb2 = 100e-4;
xb1 = -150e-4;
xb2 = 150e-4;
for iz=1:Nz % for every depth z(iz)
    % blood vessel @ xc, zc, radius, oriented along y axis
    xc       = xb1;            % [cm], center of blood vessel
    zc       = zb1;      % [cm], center of blood vessel
    vesselradius  = rb1;        % blood vessel radius [cm]
    for ix=1:Nx
            xd = x(ix) - xc;  % vessel, x distance from vessel center
            zd = z(iz) - zc;    % vessel, z distance from vessel center
            r  = sqrt(xd^2 + zd^2); % r from vessel center
            if (r≤vesselradius)       % if r is within vessel
                T(:,ix,iz) = 1; % blood
            end
    end %ix
    % blood vessel @ xc, zc, radius, oriented along y axis
    xc       = xb2;            % [cm], center of blood vessel
    zc       = zb2;      % [cm], center of blood vessel
    vesselradius  = rb2;        % blood vessel radius [cm]
    for ix=1:Nx
            xd = x(ix) - xc;  % vessel, x distance from vessel center
            zd = z(iz) - zc;    % vessel, z distance from vessel center
            r  = sqrt(xd^2 + zd^2); % r from vessel center
            if (r≤vesselradius)       % if r is within vessel
                T(:,ix,iz) = 1; % blood
            end
    end %ix
end % iz
```

Listing A.1: Create Tissue Strcture.

In Listing A.1, we create the same two blood vessels are in [1], and then it is feed to the Monte Carlo engine written in standard C for simulation.

```
%% Load the saved photons
S = DetS(ix)'; % row vectors
W = DetW(ix)';
```

```
L = DetL(ix)';
%% Construct the signal
s = exp(1j*k*S)*sqrt(W.*L)';
s = s/length(ix);
I = (abs(0.01*s+1)).^2 - (abs(0.01*s-1)).^2;
y = I;
%% Apply the window
y = y.*hamming(M);
%% Zero padding and FFT
M10 = length(y)*10;
Y10 = fft(y,M10);
z10 = (0:(M10-1))*Fs/M10;
z10 = z10/2;
z10 = z10(1:floor(length(z10)/2));
Y10 = Y10(1:floor(length(Y10)/2));
out = db(Y10);
%% Save the A-scan
OCT(:,ID) = out;
```

Listing A.2: Computation of OCT signals and A-scans.

In Listing A.2, we process the saved photons and generate A-scans for each detector. The code starts by loading the saved photon path lengths, weights, likelihood ratio and then computes the OCT signal according to Equation 4.7. The A-scans are obtained by

A.3 Coding in Standard C

In this section we list the most important as well as the tricky parts of the Monte Carlo code written in standard C. It covers lots of subtleties not covered previously.

```
//Initialize photon position and trajectory.
i_photon += 1;        /* increment photon count */
W = 1.0;                    /* set photon weight to one */
photon_status = ALIVE;      /* Launch an ALIVE photon */
det_num = -1;         /* photon not detected yet*/
first_bias_done = 0;    /* photon not biased back-scattered yet */
cont_exist = 0;       /* no split generated yet */
L_current = 1;        /* photon's initial likelihood */
s_total = 0;        /* photon's initial path length */
```

```
10  z_max   = 0;        /* photon's initial depth reached */
11  /* pick the fiber that the current photon packet is biased towards to: [1, ...
        Ndetectors] */
12  while ((rnd = RandomGen(1,0,NULL)) ≥ 1.0); // avoids rnd = 1
13  Pick_det = floor(rnd * Ndetectors) + 1;
14  /* Set trajectory to mimic A-scan */
15  if(Ndetectors==1){
16    detx = 0;
17  }else{
18    detx = 2*radius*(Pick_det - 1)/(Ndetectors - 1) - radius;
19  }
20  x     = xs + detx;
21  y    = ys;
22  z    = zs;
23  // set trajectory toward focus
24  ux     = 0;
25  uy     = 0;
26  uz     = 1;
```

Listing A.3: Initialize photon position and trajectory

In Listing A.3, we initialize the position and trajectory of the launched photon packet, as well as set up the variables and flags for the standard Monte Carlo routine and importance sampling & photo splitting.

```
1  double FindVoxelFace2(double x1,double y1,double z1, int *det_num, int ...
       Pick_det, double detx, double det_radius, double det_z, double ...
       cos_accept, int Ndetectors, double dx,double dy,double dz, double ux, ...
       double uy, double uz) {
2    int ix1 = floor(x1/dx);
3    int iy1 = floor(y1/dy);
4    int iz1 = floor(z1/dz);
5    int izd = floor(det_z/dz);
6
7    int ix2,iy2,iz2;
8    if (ux≥0)
9    ix2=ix1+1;
10   else
11   ix2 = ix1;
12
13   if (uy≥0)
14   iy2=iy1+1;
15   else
```

```
16    iy2 = iy1;
17
18    if (uz≥0)
19    iz2=iz1+1;
20    else
21    iz2 = iz1;
22
23    double xs = fabs( (ix2*dx - x1)/ux);
24    double ys = fabs( (iy2*dy - y1)/uy);
25    double zs = fabs( (iz2*dz - z1)/uz);
26
27    double s = min3(xs,ys,zs);
28    //check detection
29    if( -uz ≥ cos_accept && izd == iz1 && s == zs && fabs(y1+s*uy) ≤ det_radius){
30      if(fabs(x1 + s*ux - detx)≤det_radius) *det_num = Pick_det;
31    }
32    return (s);
33  }
```

Listing A.4: Compute the step size the photon will take to get the first voxel crossing in one single long step. We also check whether the photon packet is detected by the assigned detector.

In Listing A.4, we show the function that performs the task of identifying the photon-mesh intersections. Calling this function automatically means that the step size would make the photon leave the current voxel, therefore we also need to check whether the photon gets detected by the collecting fiber.

```
1  /**** SPIN and SPLIT *****/
2  if(photon_status == ALIVE){
3    /* check whether the first biased back-scattering has been applied: 0 = not ...
       applied, 1 = applied */
4    if(first_bias_done == 0) { /* apply the first biased scattering */
5      /* Sample for costheta_B */
6      rnd = RandomNum;
7      double temp = 1/sqrt(1 + a*a) + rnd * (1/(1-a) - 1/sqrt(1 + a*a));
8      costheta_B = 0.5/a * (a*a + 1 - 1/(temp*temp));
9      sintheta_B = sqrt(1.0 - costheta_B*costheta_B);
10     /* Sample psi. */
11     psi = 2.0*PI*RandomNum;
12     cospsi = cos(psi);
```

```
13      if (psi < PI)
14      sinpsi = sqrt(1.0 - cospsi*cospsi);     /* sqrt() is faster than sin(). */
15      else
16      sinpsi = -sqrt(1.0 - cospsi*cospsi);
17      /* Compute the unit vector v towards the actual position of the detector, ...
        where detx is chosen uniformly along the centers of the collecting fiber ...
        array. */
18      if(Ndetectors==1){
19        detx = 0;
20      }else{
21        detx = 2*radius*(Pick_det - 1)/(Ndetectors - 1) - radius;
22      }
23      dety = 0;
24      detz = det_z;
25      temp  = sqrt((x - detx)*(x - detx) + (y - dety)*(y - dety) + (z - ...
        detz)*(z - detz));
26      vx    = -(x - detx)/temp;
27      vy    = -(y - dety)/temp;
28      vz    = -(z - detz)/temp;
29      /* New trajectory u' = (upx, upy, upz) */
30      if (1 - fabs(vz) ≤ ONE_MINUS_COSZERO) {      /* close to perpendicular. */
31        upx = sintheta_B * cospsi;
32        upy = sintheta_B * sinpsi;
33        upz = costheta_B * SIGN(vz);   /* SIGN() is faster than division. */
34      }
35      else {          /* usually use this option */
36        temp = sqrt(1.0 - vz * vz);
37        upx = sintheta_B * (vx * vz * cospsi - vy * sinpsi) / temp + vx * ...
        costheta_B;
38        upy = sintheta_B * (vy * vz * cospsi + vx * sinpsi) / temp + vy * ...
        costheta_B;
39        upz = -sintheta_B * cospsi * temp + vz * costheta_B;
40      }
41      /* Compute the likelihood ratio for this particular biased ...
        back-scattering */
42      costheta_S = upx*ux + upy*uy + upz*uz;
43      temp = (1 + a*a - 2*a*costheta_B)/(1 + g*g - 2*g*costheta_S);
44      double L_temp = (1 - g*g)/(2*a*(1-a)) * (1 - (1-a)/sqrt(1+a*a)) * ...
        sqrt(temp * temp * temp);
45
46      /* Check do we have a continuing photon packet or not? */
47      if (L_temp < (1-ls) ){ // yes, do the unbiased spin and save the ...
        trajectory for the continuing photon packet
48        L_cont = L_current * (1 - L_temp);
49        i_cont = i;
```

```
50        { /* the unbiased spin*/
51
52          /* Sample for costheta */
53          rnd = RandomNum;
54          if (g == 0.0)
55          costheta = 2.0*rnd - 1.0;
56          else {
57            double temp = (1.0 - g*g)/(1.0 - g + 2*g*rnd);
58            costheta = (1.0 + g*g - temp*temp)/(2.0*g);
59          }
60          sintheta = sqrt(1.0 - costheta*costheta); /* sqrt() is faster than ...
      sin(). */
61
62          /* Sample psi. */
63          psi = 2.0*PI*RandomNum;
64          cospsi = cos(psi);
65          if (psi < PI)
66          sinpsi = sqrt(1.0 - cospsi*cospsi);     /* sqrt() is faster than ...
      sin(). */
67          else
68          sinpsi = -sqrt(1.0 - cospsi*cospsi);
69
70          /* New trajectory. */
71          if (1 - fabs(uz) ≤ ONE_MINUS_COSZERO) {      /* close to ...
      perpendicular. */
72            uxx = sintheta * cospsi;
73            uyy = sintheta * sinpsi;
74            uzz = costheta * SIGN(uz);   /* SIGN() is faster than division. */
75          }
76          else {          /* usually use this option */
77            temp = sqrt(1.0 - uz * uz);
78            uxx = sintheta * (ux * uz * cospsi - uy * sinpsi) / temp + ux * ...
      costheta;
79            uyy = sintheta * (uy * uz * cospsi + ux * sinpsi) / temp + uy * ...
      costheta;
80            uzz = -sintheta * cospsi * temp + uz * costheta;
81          }
82        }
83        ux_cont = uxx;
84        uy_cont = uyy;
85        uz_cont = uzz;
86        x_cont  = x;
87        y_cont  = y;
88        z_cont  = z;
89        W_cont  = W;
```

```
90        s_total_cont = s_total;
91        z_max_cont   = z_max;
92        L_current *= L_temp;
93        cont_exist = 1;
94      }
95      else { // no continuing photon packet
96        L_current *= L_temp;
97        cont_exist = 0;
98      }
99      /* Update trajectory */
100     ux = upx;
101     uy = upy;
102     uz = upz;
103     first_bias_done = 1;
104   }
105   else {/* first biased back-scattering already done, apply additional biased ...
        forward-scattering */
106     if(RandomNum ≤ p){ // apply biased forward-scattering
107       /* Sample for costheta_B */
108       rnd = RandomNum;
109       double temp = 1/sqrt(1 + a*a) + rnd * (1/(1-a) - 1/sqrt(1 + a*a));
110       costheta_B = 0.5/a * (a*a + 1 - 1/(temp*temp));
111       sintheta_B = sqrt(1.0 - costheta_B*costheta_B);
112       /* Sample psi. */
113       psi = 2.0*PI*RandomNum;
114       cospsi = cos(psi);
115       if (psi < PI)
116       sinpsi = sqrt(1.0 - cospsi*cospsi);     /* sqrt() is faster than sin(). */
117       else
118       sinpsi = -sqrt(1.0 - cospsi*cospsi);
119
120       /* Compute the unit vector v towards the actual position of the ...
        detector, where detx is chosen uniformly along the centers of the ...
        collecting fiber array. */
121       if(Ndetectors==1){
122         detx = 0;
123       }else{
124         detx = 2*radius*(Pick_det - 1)/(Ndetectors - 1) - radius;
125       }
126       dety = 0;
127       detz = det_z;
128       temp  = sqrt((x - detx)*(x - detx) + (y - dety)*(y - dety) + (z - ...
        detz)*(z - detz));
129       vx    = -(x - detx)/temp;
130       vy    = -(y - dety)/temp;
```

```
131       vz    = -(z - detz)/temp;
132       /* New trajectory u' = (upx, upy, upz) */
133       if (1 - fabs(vz) ≤ ONE_MINUS_COSZERO) {      /* close to perpendicular. */
134         upx = sintheta_B * cospsi;
135         upy = sintheta_B * sinpsi;
136         upz = costheta_B * SIGN(vz);   /* SIGN() is faster than division. */
137       }
138       else {          /* usually use this option */
139         temp = sqrt(1.0 - vz * vz);
140         upx = sintheta_B * (vx * vz * cospsi - vy * sinpsi) / temp + vx * ...
      costheta_B;
141         upy = sintheta_B * (vy * vz * cospsi + vx * sinpsi) / temp + vy * ...
      costheta_B;
142         upz = -sintheta_B * cospsi * temp + vz * costheta_B;
143       }
144       /* Compute the likelihood ratio for this particular biased ...
      forward-scattering */
145       costheta_S = upx*ux + upy*uy + upz*uz;
146       temp = 1 + g*g - 2*g*costheta_S;
147       f_HG = (1 - g*g) * 0.5 / sqrt(temp*temp*temp);
148       temp = 1 + a*a - 2*a*costheta_B;
149       f_B = a*(1-a)/((sqrt(temp*temp*temp))*(1 - (1-a)/sqrt(1+a*a)));
150       double L_temp = f_HG/(p*f_B + (1-p)*f_HG);
151       L_current *= L_temp;
152       /* Update trajectory */
153       ux = upx;
154       uy = upy;
155       uz = upz;
156     } else {// apply unbiased scattering
157       /* Sample for costheta */
158       rnd = RandomNum;
159       if (g == 0.0)
160       costheta = 2.0*rnd - 1.0;
161       else {
162         double temp = (1.0 - g*g)/(1.0 - g + 2*g*rnd);
163         costheta = (1.0 + g*g - temp*temp)/(2.0*g);
164       }
165       sintheta = sqrt(1.0 - costheta*costheta); /* sqrt() is faster than ...
      sin(). */
166       /* Sample psi. */
167       psi = 2.0*PI*RandomNum;
168       cospsi = cos(psi);
169       if (psi < PI)
170       sinpsi = sqrt(1.0 - cospsi*cospsi);     /* sqrt() is faster than sin(). */
171       else
```

```
172         sinpsi = -sqrt(1.0 - cospsi*cospsi);
173         /* New trajectory. */
174         if (1 - fabs(uz) ≤ ONE_MINUS_COSZERO) {      /* close to perpendicular. */
175           uxx = sintheta * cospsi;
176           uyy = sintheta * sinpsi;
177           uzz = costheta * SIGN(uz);   /* SIGN() is faster than division. */
178         }
179         else {          /* usually use this option */
180           temp = sqrt(1.0 - uz * uz);
181           uxx = sintheta * (ux * uz * cospsi - uy * sinpsi) / temp + ux * costheta;
182           uyy = sintheta * (uy * uz * cospsi + ux * sinpsi) / temp + uy * costheta;
183           uzz = -sintheta * cospsi * temp + uz * costheta;
184         }
185         /* Compute the unit vector v towards the actual position of the ...
        detector, where detx is chosen uniformly along the centers of the ...
        collecting fiber array. */
186         if(Ndetectors==1){
187           detx = 0;
188         }else{
189           detx = 2*radius*(Pick_det - 1)/(Ndetectors - 1) - radius;
190         }
191         dety = 0;
192         detz = det_z;
193         temp  = sqrt((x - detx)*(x - detx) + (y - dety)*(y - dety) + (z - ...
        detz)*(z - detz));
194         vx    = -(x - detx)/temp;
195         vy    = -(y - dety)/temp;
196         vz    = -(z - detz)/temp;
197         /* Compute the likelihood ratio for this particular biased ...
        forward-scattering */
198         costheta_S = costheta;
199         costheta_B = uxx*vx + uyy*vy + uzz*vz;
200         temp = 1 + g*g - 2*g*costheta_S;
201         f_HG = (1 - g*g) * 0.5 / sqrt(temp*temp*temp);
202         temp = 1 + a*a - 2*a*costheta_B;
203         f_B = a*(1-a)/((sqrt(temp*temp*temp))*(1 - (1-a)/sqrt(1+a*a)));
204         double L_temp = f_HG/(p*f_B + (1-p)*f_HG);
205         L_current *= L_temp;
206         /* Update trajectory */
207         ux = uxx;
208         uy = uyy;
209         uz = uzz;
210       }
211     }
212   }
```

Listing A.5: Spin and Split. The Spin process is to scatter photon into new trajectory defined by θ and ψ. θ is specified by $\cos(\theta)$, which is determined based on the Henyey-Greenstein scattering function, and then convert θ and ψ into cosines u_x, u_y, u_z. Split follows exactly the procedure described in Section 4.3, where we apply biased backward-scatterings and biased forward-scattering, as well as unbiased scatterings. Once the first biased backward-scattering takes place, we split the photon packet into two if the likelihood ratio of the biased back-scattering is less than 1. We save the information of the continuing photon and continue to track the current photon, by applying biased and unbiased forward-scatterings.

The Listing A.5 is the bread and butter of the advanced Monte Carlo engine. The Spin process comes from the standard Monte Carlo routine to scatter photon into new trajectory based on the Henyey-Greenstein scattering function, and then update the directional cosines u_x, u_y, u_z of the photon packet. The Split part is the actual implementation of importance sampling & photon splitting and follows exactly the procedure described in Section 4.3, which is about when and how to apply biased backward- and forward-scatterings, as well as unbiased scatterings. Once a photon packet is about to experience a scattering event, we first determine whether the first biased backward scattering has been done. Once the first biased backward-scattering takes place, we split the photon packet into two if the likelihood ratio of the biased back-scattering is less than 1. We save the information of the continuing photon and continue to track the current photon, by applying biased and unbiased forward-scatterings.

Appendix B

Source Code in Standard C

B.1 Declaration of Functions

```
1 double RandomGen(char Type, long Seed, long *Status);
2 Boolean SameVoxel(double x1,double y1,double z1, double x2, double y2, double ...
      z2, double dx,double dy,double dz);
3 double max2(double a, double b);
4 double min2(double a, double b);
5 double min3(double a, double b, double c);
6 double FindVoxelFace2(double x1,double y1,double z1, int *det_num, int ...
      Pick_det, double detx, double det_radius, double det_z, double ...
      cos_accept, int Ndetectors, double dx,double dy,double dz, double ux, ...
      double uy, double uz);
7 /* How much step size will the photon take to get the first voxel crossing in ...
      one single long step? */
8 double RFresnel(double n1, double n2, double ca1, double *ca2_Ptr);
```

Listing B.1: Function Declarations

B.2 Major Cycle of the Monte Carlo Engine

```
1 * ========================= MAJOR CYCLE =====================*/
2 start_time = clock();
3 now = time(NULL);
4 printf("%s\n", ctime(&now));
5
```

```
6  /**** INITIALIZATIONS*****/
7  RandomGen(0, -(int)time(NULL)%(1<<15), NULL); /* initiate with seed = 1, or ...
       any long integer. */
8
9  /**** RUN
10 Launch N photons, initializing each one before progation.
11 *****/
12 printf("-------------- Begin Monte Carlo --------------\n");
13 printf("%s\n",myname);
14 printf("requesting %0.1f min\n",time_min);
15 Nphotons = 200; // will be updated to achieve desired run time, time_min.
16 i_photon = 0;
17 c_photon = 0;
18 //CNT = 0;
19 do {
20
21   /* Get tissue voxel properties of launchpoint.
22 * If photon beyond outer edge of defined voxels,
23 * the tissue equals properties of outermost voxels.
24 * Therefore, set outermost voxels to infinite background value.
25 */
26   ix = (int)(Nx/2 + x/dx);
27   iy = (int)(Ny/2 + y/dy);
28   iz = (int)(z/dz);
29   if (ix≥Nx) ix=Nx-1;
30   if (iy≥Ny) iy=Ny-1;
31   if (iz≥Nz) iz=Nz-1;
32   if (ix<0)   ix=0;
33   if (iy<0)   iy=0;
34   if (iz<0)   iz=0;
35   /* Get the tissue type of located voxel */
36   i    = (long)(iz*Ny*Nx + ix*Ny + iy);
37   type   = v[i];
38   mua    = muav[type];
39   mus    = musv[type];
40   g      = gv[type];
41
42   bflag = 1; // initialize as 1 = inside volume, but later check as photon ...
       propagates.
43
44   /* HOP_DROP_SPIN_CHECK
45 Propagate one photon until it dies as determined by ROULETTE.
46 At the beginning, check whether to load the continuing photon.
47 *******/
48   do { // while photon_status == ALIVE
```

```
49
50      if(photon_status == DEAD){ //load the continuing photon and update the flags
51
52        x     = x_cont;
53        y     = y_cont;
54        z     = z_cont;
55        ux    = ux_cont;
56        uy    = uy_cont;
57        uz    = uz_cont;
58        i     = i_cont;
59        s_total = s_total_cont;
60        z_max = z_max_cont;
61        type = v[i];
62        mua   = muav[type];
63        mus   = musv[type];
64        g     = gv[type];
65        W     = W_cont;
66        L_current = L_cont;
67        cont_exist = 0;
68        photon_status = ALIVE;
69        first_bias_done = 0;
70        det_num = -1;
71      }
72      /**** HOP and DETECTION
73    Take step to new position
74    s = dimensionless stepsize
75    ux, uy, uz are cosines of current photon trajectory.
76    **** Detection Check:
77    During the hop, photon may go out of simulation domain or be detected, thus ...
        changing the photon_status = DEAD
78    *****/
79      while ((rnd = RandomNum) ≤ 0.0);   /* yields 0 < rnd ≤ 1 */
80      sleft = -log(rnd);        /* dimensionless step */
81      //CNT += 1;
82
83      do{  // while sleft>0
84        s     = sleft/mus;        /* Step size [cm].*/
85        tempx = x + s*ux;       /* Update positions. [cm] */
86        tempy = y + s*uy;
87        tempz = z + s*uz;
88
89        sv = SameVoxel(x,y,z, tempx, tempy, tempz, dx,dy,dz);
90        if (sv) /* photon in same voxel */
91        {
92          x=tempx;          /* Update positions. */
```

```
93          y=tempy;
94          z=tempz;
95
96          /**** DROP
97        Drop photon weight (W) into local bin.
98        *****/
99          absorb = W*(1 - exp(-mua*s)); /* photon weight absorbed at this step */
100         W -= absorb;          /* decrement WEIGHT by amount absorbed */
101
102         /* Update sleft */
103         sleft = 0;    /* dimensionless step remaining */
104
105
106         /* Update total path length */
107         s_total += s;
108
109         /* Update maximum depth reached */
110         if(z > z_max) {z_max = z;}
111       }
112       else /* photon has crossed voxel boundary */
113       {
114
115         /*** collecting fiber array ranges:
116       along x: [-Ndetectors * det_radius, Ndetectors * det_radius]
117       along y: [-det_radius, det_radius]
118       along z: [det_z, det_z+dz]
119       ***/
120
121         /* step to voxel face + "littlest step" so just inside new voxel. */
122         s = ls + FindVoxelFace2(x,y,z, ...
      &det_num,Pick_det,detx,det_radius,det_z,cos_accept,Ndetectors, dx,dy,dz, ...
      ux,uy,uz);
123
124         /**** DROP
125       Drop photon weight (W) into local bin.
126       *****/
127         absorb = W*(1-exp(-mua*s));   /* photon weight absorbed at this step */
128         W -= absorb;                  /* decrement WEIGHT by amount absorbed */
129
130         if (det_num != -1){          /* check if the photon is detected. */
131
132           /* Update total path length */
133           s_total += s;
134
135           /* Save properties of interest */
```

```
136           if(L_current > 0 && det_num == Pick_det){ // avoid NAN and zero ...
      likelihood, and avoid cross-detection
137             DetID[c_photon] = det_num;
138             DetW[c_photon]  = W;
139             DetL[c_photon]  = L_current;
140             DetS[c_photon]  = s_total;
141             DetZ[c_photon]  = z_max;
142             /* increment collected photon count */
143             c_photon += 1;
144           }
145           //if(c_photon ==1) {printf("OK at 590;\n");}
146
147           photon_status = DEAD;
148           sleft = 0;
149
150
151         }
152         else{ // not detected
153
154           /* Update sleft */
155           sleft -= s*mus;  /* dimensionless step remaining */
156           if (sleft≤ls) sleft = 0;
157
158           /* Update positions. */
159           x += s*ux;
160           y += s*uy;
161           z += s*uz;
162
163           /* Update total path length */
164           s_total += s;
165
166           /* Update maximum depth reached */
167           if (z>z_max) z_max = z;
168
169           // pointers to voxel containing optical properties
170           ix = (int)(Nx/2 + x/dx);
171           iy = (int)(Ny/2 + y/dy);
172           iz = (int)(z/dz);
173
174           bflag = 1;  // Boundary flag. Initialize as 1 = inside volume, then ...
      check.
175           if (boundaryflag==0) { // Infinite medium.
176             // Check if photon has wandered outside volume.
177             // If so, set tissue type to boundary value, but let photon wander.
178             // Set bflag to zero, so DROP does not deposit energy.
```

```
179             if (iz≥Nz) {iz=Nz-1; bflag = 0;}
180             if (ix≥Nx) {ix=Nx-1; bflag = 0;}
181             if (iy≥Ny) {iy=Ny-1; bflag = 0;}
182             if (iz<0)   {iz=0;    bflag = 0;}
183             if (ix<0)   {ix=0;    bflag = 0;}
184             if (iy<0)   {iy=0;    bflag = 0;}
185           }
186           else if (boundaryflag==1) { // Escape at boundaries
187             if (iz≥Nz) {iz=Nz-1; photon_status = DEAD; sleft=0;}
188             if (ix≥Nx) {ix=Nx-1; photon_status = DEAD; sleft=0;}
189             if (iy≥Ny) {iy=Ny-1; photon_status = DEAD; sleft=0;}
190             if (iz<0)   {iz=0;    photon_status = DEAD; sleft=0;}
191             if (ix<0)   {ix=0;    photon_status = DEAD; sleft=0;}
192             if (iy<0)   {iy=0;    photon_status = DEAD; sleft=0;}
193           }
194           else if (boundaryflag==2) { // Escape at top surface, no x,y, ...
        bottom z boundaries
195             if (iz≥Nz) {iz=Nz-1; bflag = 0;}
196             if (ix≥Nx) {ix=Nx-1; bflag = 0;}
197             if (iy≥Ny) {iy=Ny-1; bflag = 0;}
198             if (iz<0)   {iz=0;    photon_status = DEAD; sleft=0;}
199             if (ix<0)   {ix=0;    bflag = 0;}
200             if (iy<0)   {iy=0;    bflag = 0;}
201           }
202           // update pointer to tissue type
203           i    = (long)(iz*Ny*Nx + ix*Ny + iy);
204           type = v[i];
205           mua  = muav[type];
206           mus  = musv[type];
207           g    = gv[type];
208         }
209       } //(sv) /* same voxel */
210     } while(sleft>0); //do...while
211
212
213     /**** CHECK ROULETTE
214   If photon weight below THRESHOLD, then terminate photon using Roulette ...
        technique.
215   Photon has CHANCE probability of having its weight increased by factor of ...
        1/CHANCE,
216   and 1-CHANCE probability of terminating.
217   *****/
218     if(photon_status == ALIVE){
219       if (W < THRESHOLD) {
220         if (RandomNum ≤ CHANCE)
```

```
221           W /= CHANCE;
222           else photon_status = DEAD;
223         }
224       }
225     } while (photon_status == ALIVE || cont_exist == 1);  /* end ...
        STEP_CHECK_HOP_SPIN */
226     /* if ALIVE, continue propagating the current photon */
227     /* If current photon DEAD, then load the continuing photon. */
228   } while (i_photon < Nphotons);  /* end RUN */
```

Listing B.2: Major Cycle of our Monte Carlo Engine, with the Spin and Split part and the Launch part removed.

Bibliography

[1] A. E. Hartinger, A. S. Nam, I. Chico-Calero, and B. J. Vakoc, "Monte carlo modeling of angiographic optical coherence tomography," *Biomedical optics express*, vol. 5, no. 12, pp. 4338–4349, 2014.

[2] J. Phillips, K. W. Bowyer, and P. J. Flynn, "Comments on the casia version 1.0 iris data set," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, no. 10, pp. 1869–1870, 2007.

[3] A. F. Fercher, W. Drexler, C. K. Hitzenberger, and T. Lasser, "Optical coherence tomography-principles and applications," *Reports on progress in physics*, vol. 66, no. 2, p. 239, 2003.

[4] P. H. Tomlins and R. Wang, "Theory, developments and applications of optical coherence tomography," *Journal of Physics D: Applied Physics*, vol. 38, no. 15, p. 2519, 2005.

[5] K. Bizheva, B. Považay, B. Hermann, H. Sattmann, W. Drexler, M. Mei, R. Holzwarth, T. Hoelzenbein, V. Wacheck, and H. Pehamberger, "Compact, broad-bandwidth fiber laser for sub-2-μm axial resolution optical coherence tomography in the 1300-nm wavelength region," *Optics letters*, vol. 28, no. 9, pp. 707–709, 2003.

[6] J. G. Fujimoto, "Optical coherence tomography for ultrahigh resolution in vivo imaging," *Nature biotechnology*, vol. 21, no. 11, pp. 1361–1367, 2003.

[7] C. Xu, C. Vinegoni, T. S. Ralston, W. Luo, W. Tan, and S. A. Boppart, "Spectroscopic spectral-domain optical coherence microscopy," *Optics letters*, vol. 31, no. 8, pp. 1079–1081, 2006.

[8] J. Serup, G. B. Jemec, and G. L. Grove, *Handbook of non-invasive methods and the skin*. CRC press, 2006.

[9] J. B. Thomsen, B. Sander, M. Mogensen, L. Thrane, T. M. Jørgensen, G. B. Jemec, and P. E. Andersen, "Optical coherence tomography: Technique and applications," in *Advanced Imaging in Biology and medicine*, pp. 103–129, Springer, 2009.

[10] M. Mogensen, L. Thrane, T. M. Jørgensen, P. E. Andersen, and G. Jemec, "Optical coherence tomography for imaging of skin and skin diseases," in *Seminars in cutaneous medicine and surgery*, vol. 28, pp. 196–202, WB Saunders, 2009.

[11] S. Malektaji, I. T. Lima, and S. S. Sherif, "Monte carlo simulation of optical coherence tomography for turbid media with arbitrary spatial distributions," *Journal of biomedical optics*, vol. 19, no. 4, pp. 046001–046001, 2014.

[12] A. Vasilyev, *The optoelectronic swept-frequency laser and its applications in ranging, three-dimensional imaging, and coherent beam combining of chirped-seed amplifiers*. PhD thesis, Citeseer, 2013.

[13] H. Shen and G. Wang, "A tetrahedron-based inhomogeneous monte carlo optical simulator," *Physics in medicine and biology*, vol. 55, no. 4, p. 947, 2010.

[14] A. J. Welch and M. J. Van Gemert, *Optical-thermal response of laser-irradiated tissue*, vol. 1. Springer, 1995.

[15] S. L. Jacques and B. W. Pogue, "Tutorial on diffuse light transport," *Journal of Biomedical Optics*, vol. 13, no. 4, pp. 041302–041302, 2008.

[16] A. J. Welch and M. J. van Gemert, "Overview of optical and thermal laser-tissue interaction and nomenclature," in *Optical-Thermal Response of Laser-Irradiated Tissue*, pp. 3–11, Springer, 2011.

[17] S. L. Jacques, "Optical properties of biological tissues: a review," *Physics in medicine and biology*, vol. 58, no. 11, p. R37, 2013.

[18] I. Meglinski, "Modeling the reflectance spectra of the optical radiation for random inhomogeneous multi-layered highly scattering and absorbing media by the monte carlo technique," *Quantum Electron*, vol. 31, no. 12, pp. 1101–1107, 2001.

[19] M. Y. Kirillin, A. V. Priezzhev, and R. A. Myllylä, "Role of multiple scattering in formation of oct skin images," *Quantum Electronics*, vol. 38, no. 6, pp. 570–575, 2008.

[20] R. Meier, J.-S. Lee, and D. Anderson, "Atmospheric scattering of middle uv radiation from an internal source," *Applied optics*, vol. 17, no. 20, pp. 3216–3225, 1978.

[21] E. Berrocal, D. L. Sedarsky, M. E. Paciaroni, I. V. Meglinski, and M. A. Linne, "Laser light scattering in turbid media part i: Experimental and simulated results for the spatial intensity distribution," *Optics express*, vol. 15, no. 17, pp. 10649–10665, 2007.

[22] E. Berrocal, I. Meglinski, D. Greenhalgh, and M. Linne, "Image transfer through the complex scattering turbid media," *Laser Physics Letters*, vol. 3, no. 9, pp. 464–467, 2006.

[23] G. Yao and L. V. Wang, "Monte carlo simulation of an optical coherence tomography signal in homogeneous turbid media," *Physics in medicine and biology*, vol. 44, no. 9, p. 2307, 1999.

[24] M. Kirillin, M. Shirmanova, M. Sirotkina, M. Bugrova, B. Khlebtsov, and E. Zagaynova, "Contrasting properties of gold nanoshells and titanium dioxide

nanoparticles for optical coherence tomography imaging of skin: Monte carlo simulations and in vivo study," *Journal of biomedical optics*, vol. 14, no. 2, pp. 021017–021017, 2009.

[25] B. Karamata, P. Lambelet, M. Laubscher, M. Leutenegger, S. Bourquin, and T. Lasser, "Multiple scattering in optical coherence tomography. i. investigation and modeling," *JOSA A*, vol. 22, no. 7, pp. 1369–1379, 2005.

[26] M. Y. Kirillin, I. Meglinskii, and A. V. Priezzhev, "Effect of photons of different scattering orders on the formation of a signal in optical low-coherence tomography of highly scattering media," *Quantum Electronics*, vol. 36, no. 3, p. 247, 2006.

[27] V. Kuzmin and I. Meglinski, "Multiple scattering and intensity fluctuations in optical coherent tomography of randomly inhomogeneous media," *Journal of Experimental and Theoretical Physics*, vol. 105, no. 2, pp. 285–291, 2007.

[28] L. Wang, S. L. Jacques, and L. Zheng, "Mcmlmonte carlo modeling of light transport in multi-layered tissues," *Computer methods and programs in biomedicine*, vol. 47, no. 2, pp. 131–146, 1995.

[29] I. M. Sobol, *A primer for the Monte Carlo method*. CRC press, 1994.

[30] A. Ishimaru, *Wave propagation and scattering in random media*, vol. 2. Academic press New York, 1978.

[31] D. Churmakov, I. Meglinski, and D. Greenhalgh, "Influence of refractive index matching on the photon diffuse reflectance," *Physics in medicine and biology*, vol. 47, no. 23, p. 4271, 2002.

[32] M. Kirillin, I. Meglinski, V. Kuzmin, E. Sergeeva, and R. Myllylä, "Simulation of optical coherence tomography images by monte carlo modeling based on polarization vector approach," *Optics express*, vol. 18, no. 21, pp. 21714–21724, 2010.

[33] M. Y. Kirillin, G. Farhat, E. A. Sergeeva, M. C. Kolios, and A. Vitkin, "Speckle statistics in oct images: Monte carlo simulations and experimental studies," *Optics letters*, vol. 39, no. 12, pp. 3472–3475, 2014.

[34] S. S. Sherif, C. C. Rosa, C. Flueraru, S. Chang, Y. Mao, and A. G. Podoleanu, "Statistics of the depth-scan photocurrent in time-domain optical coherence tomography," *JOSA A*, vol. 25, no. 1, pp. 16–20, 2008.

[35] M. Yadlowsky, J. Schmitt, and R. Bonner, "Multiple scattering in optical coherence microscopy," *Applied optics*, vol. 34, no. 25, pp. 5699–5707, 1995.

[36] B. Wilson and G. Adam, "A monte carlo model for the absorption and flux distributions of light in tissue," *Medical physics*, vol. 10, no. 6, pp. 824–830, 1983.

[37] N. G. Chen and J. Bai, "Estimation of quasi-straightforward propagating light in tissues," *Physics in medicine and biology*, vol. 44, no. 7, p. 1669, 1999.

[38] N. Chen, "Controlled monte carlo method for light propagation in tissue of semi-infinite geometry," *Applied optics*, vol. 46, no. 10, pp. 1597–1603, 2007.

[39] I. T. Lima, A. Kalra, H. E. Hernández-Figueroa, and S. S. Sherif, "Fast calculation of multipath diffusive reflectance in optical coherence tomography," *Biomedical optics express*, vol. 3, no. 4, pp. 692–700, 2012.

[40] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo method*, vol. 707. John Wiley & Sons, 2011.

[41] G. Biondini, W. L. Kath, and C. R. Menyuk, "Importance sampling for polarization-mode dispersion," *Photonics Technology Letters, IEEE*, vol. 14, no. 3, pp. 310–312, 2002.

[42] S. L. Fogal, G. Biondini, and W. L. Kath, "Multiple importance sampling for first-and second-order polarization-mode dispersion," *Photonics Technology Letters, IEEE*, vol. 14, no. 9, pp. 1273–1275, 2002.

[43] I. T. Lima Jr, A. O. Lima, J. Zweck, and C. R. Menyuk, "Efficient computation of outage probabilities due to polarization effects in a wdm system using a reduced stokes model and importance sampling," *Photonics Technology Letters, IEEE*, vol. 15, no. 1, pp. 45–47, 2003.

[44] I. T. Lima, A. O. Lima, G. Biondini, C. R. Menyuk, and W. L. Kath, "A comparative study of single-section polarization-mode dispersion compensators," *Journal of lightwave technology*, vol. 22, no. 4, p. 1023, 2004.

[45] J. Schmitt and K. Ben-Letaief, "Efficient monte carlo simulation of confocal microscopy in biological tissue," *JOSA A*, vol. 13, no. 5, pp. 952–961, 1996.

[46] H. Iwabuchi, "Efficient monte carlo methods for radiative transfer modeling," *Journal of the atmospheric sciences*, vol. 63, no. 9, pp. 2324–2339, 2006.

[47] I. Lima Jr, "Advanced monte carlo methods applied to optical coherence tomography," *invited), presented at the*, 2009.

[48] I. T. Lima, A. Kalra, and S. S. Sherif, "Improved importance sampling for monte carlo simulation of time-domain optical coherence tomography," *Biomedical optics express*, vol. 2, no. 5, pp. 1069–1081, 2011.

[49] G. Contributors, "Gsl-gnu scientific library-gnu project-free software foundation (fsf)," 2010.

[50] A. Papoulis and S. U. Pillai, *Probability, random variables, and stochastic processes*. Tata McGraw-Hill Education, 2002.

[51] M. Dumont, R. Marée, L. Wehenkel, and P. Geurts, "Fast multi-class image annotation with random subwindows and multiple output randomized trees," in *International Conference on Computer Vision Theory and Applications (VISAPP)*, vol. 2, 2009.

[52] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, "Classification and regression trees. wadsworth," *Belmont, CA*, 1984.

[53] J. R. Quinlan, *C4. 5: programs for machine learning*. Elsevier, 2014.

[54] T. J. Hastie, R. J. Tibshirani, and J. H. Friedman, *The elements of statistical learning: data mining, inference, and prediction*. Springer, 2009.

[55] J. H. Friedman, "Greedy function approximation: a gradient boosting machine," *Annals of statistics*, pp. 1189–1232, 2001.

[56] J. H. Friedman, "Stochastic gradient boosting," *Computational Statistics & Data Analysis*, vol. 38, no. 4, pp. 367–378, 2002.

[57] G. Ridgeway, "Generalized boosted models: A guide to the gbm package," *Update*, vol. 1, no. 1, p. 2007, 2007.

[58] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[59] T. Tieleman, "Training restricted boltzmann machines using approximations to the likelihood gradient," in *Proceedings of the 25th international conference on Machine learning*, pp. 1064–1071, ACM, 2008.