Balancing Fidelity and Performance in Iridal Light Transport Simulations Aimed at Interactive Applications

by

Boris Kravchenko

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2016

Abstract

Specific light transport models based on first-principles approaches have been proposed for complex organic materials such as human skin and blood. The driving force behind these efforts has been the high-fidelity reproduction of material appearance attributes without one having to rely on the manipulation of ad hoc parameters. These models, however, are usually considered excessively time consuming for rendering applications requiring interactive rates. In this thesis, we address this open problem with respect to one of the most challenging of these organic materials, namely the human iris. More specifically, we present a framework that consists in the careful configuration of algorithms employed by a biophysically-based iridal light transport model on the CUDA (Compute Unified Device Architecture) parallel computing platform. We then investigate the sensitivity of iridal appearance attributes to key model running parameters, namely spectral resolution and number of sample rays, in order to obtain a practical balance between appearance fidelity and performance on this platform. The results of our investigation indicate that predictive light transport simulations can be effectively employed in the generation of iridal images that are not only believable, but also controlled by biophysically meaningful parameters. Although our investigation is centered at the human iris, we believe that it can be viewed as a proof of concept, and the proposed configuration strategies and parameter space explorations can be employed to obtain similar results for other organic materials.

Acknowledgements

I am very grateful to my supervisor Dr. Gladimir Baranoski for his guidance, help and for the countless readings and revisions of this work. He has taught me to take the most thorough approach when in pursuit of the scientific method as well as the importance of always polishing my work to perfection. After all, “We never know who is sitting in the audience”.

Dedication

This is dedicated to my family, I would not have achieved this without their love and support.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

The synthesis of realistic images of organic materials has always been a challenge for the computer graphics community. One of the main research avenues in this area involves the high-fidelity[1] modeling [23] of appearance attributes [16]. Another avenue involves the use of specialized GPUs (Graphics Processing Units) to enable the fast generation of believable images for interactive applications (*e.g.*, [11, 14, 28, 56]). As the advances in this area continue, the demand for higher correctness to cost ratios in the rendering process has been increasing accordingly.

The key aspect in this context is predictability, which is instrumental to increase the fidelity of the image synthesis process [22, 23] and to make it less dependent on manual tweaks [19]. To achieve predictable rendering results controlled by biophysically meaningful parameters, one often needs to resort to light transport models developed using first-

[1]In this thesis, we consider the following definition of fidelity as stated by Gross [23]: "The degree to which a model or simulation reproduces the state and behavior of a real world object or the perception of a real world object, feature, condition, or chosen standard in a measurable or perceivable manner."

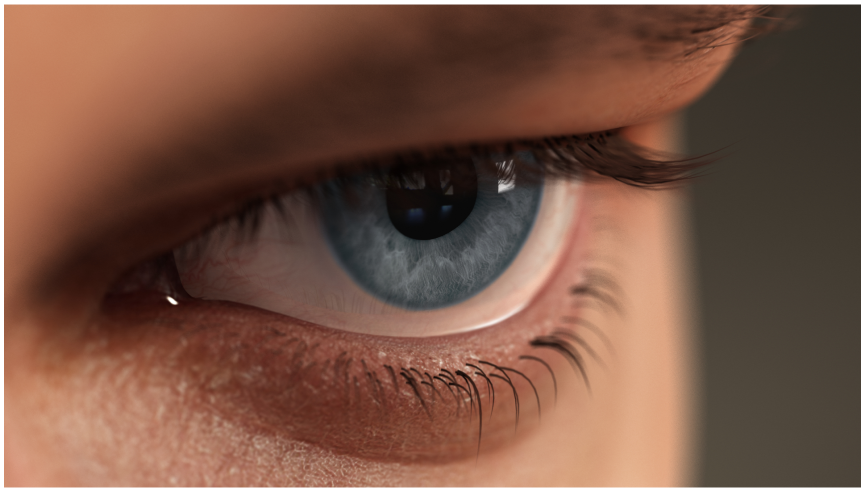


Figure 1.1: Image generated using modeled iridal data computed by the proposed framework.

principles approaches. Due to their relatively high performance overhead, however, these models are usually not considered for use in rendering applications with turnaround times on the order of milliseconds.

Our investigation addresses these issues while focusing on the human iris, arguably one of the most noticeable organic materials employed in the creation of virtual characters. As pointed out by Lee *et al.* [31] and Walt Disney himself [57], natural-looking eyes are quite desirable for entertainment applications depicting close-ups of human faces. From a life sciences perspective, there is also a strong interest in the human iris' chromatic attributes and how they are determined by the propagation and attenuation of light within the iridal tissues. For example, in the medical field, a number of studies have related the incidence of several ocular diseases, such as the degeneration of ocular tissues [13] and melanoma [48], to these attributes.

The model known as ILIT (**I**ridal **Li**ght **T**ransport) was specifically designed for the predictive simulation of light interactions with the human iris [30]. It employs a ray-optics formulation to simulate the scattering and absorption of light within the iridal tissues using Monte Carlo methods. ILIT can provide radiometric responses (*e.g.*, radiance or reflectance) with distinct spectral resolutions, and its predictions have shown close quantitative and qualitative agreement with measured data [3, 30]. Its algorithms, however, are bound to incur relatively high processing times due to their stochastic nature.

As stated earlier, the current state of the art in realistic image synthesis enables either the fast generation of believable images or the predictive generation of high-fidelity images at the expense of higher computational time. To date, it seems that achieving high performance and high fidelity are two conflicting goals. In this thesis, we address this timely issue by answering the following question:

How can we generate realistic iridal images quickly (at interactive rates) without sacrificing fidelity and predictability?

More specifically, we demonstrate that realistic depictions of the human iris can be obtained using predictive simulations executed at interactive rates. That is, we present a framework that consists in the configuration of the ILIT simulation algorithms on CUDA (Compute Unified Device Architecture), the parallel computing platform and API (Application Programming Interface) introduced by NVIDIA to facilitate general-purpose computations on the GPU [10]. We then employ this framework to investigate the sensitivity of iridal appearance attributes to parameters directly associated with the light transport simulations' running time, namely spectral resolution and number of sample rays. This allows us to maximize the performance gains provided by the proposed framework while maintaining appearance fidelity. We believe that the methodology and observations derived from our investigation can be used to obtain similar results for other complex organic materials.

1.1 Related Work

In this section, we briefly review relevant initiatives toward the generation of realistic images of ocular structures, notably the human iris, within and outside the field of computer graphics. Moreover, although a comprehensive review of CUDA applications is beyond the scope of this research, we also highlight a number of selected works to illustrate the diverse use of this platform.

1.1.1 Modeling and Rendering Ocular Structures

Different approaches have been employed by the computer graphics community to generate realistic images of ocular structures, in particular the human iris. For example, Sager *et al.* [51] proposed the rendering of an anatomically detailed eye model to be employed in a surgical simulator. In their work, the iris was represented using a Gouraud shaded polygon with colours specified by a colour ramp. Lefohn *et al.* [32] presented the first biologically-motivated algorithm specifically designed for the rendering of realistic looking irides. Their modeling approach was based on an ocular prosthetics methodology where several multi-transparent layers are combined to create an artificial iris. Wecker *et al.* [58] proposed an image-based technique that consists in decomposing photographs of the human iris into several components. These are then recombined to generate distinct synthetic iridal images. Lam and Baranoski [30] then presented ILIT as the first biophysically-based appearance model for the human iris.

After the development of ILIT, several works, mostly targeting the generation of believable iridal images in real time, followed. For example, Francois *et al.* [20] introduced an image-based method for estimating both iridal morphology and optical parameters from iridal photographs. The resulting iridal layered model was then employed in the real-time rendering of believable eye images using the GPU. Pamplona *et al* [46] proposed an image-based model for iridal pattern deformation to be used in conjunction with a physiologically-based model for pupil light reflex. Chiang and Fyffe [11] used a GPU-adapted normal mapping approach to generate believable eye representations for real-time facial rendering purposes. The appearance of the human iris was modeled employing iridal chromatic attributes obtained from photographs along with artistic tools to emulate effects like caustics and refraction darkening. Jimenez *et al.* [27] presented a real-time shader-based eye rendering solution for use in interactive applications. More recently, Bérard *et al.* [5] presented a system to capture the geometry and texture of ocular structures.

The ocular structures, particularly the human iris, have also been object of detailed modeling investigations outside the field of computer graphics. For example, Cai *et al.* [9] created an anatomically-based parametric eyeball model as part of a surgical simulator. Aiming at the testing of iris recognition algorithms for biometrics applications, Makthal and Ross [35] proposed a technique based on stochastic methods to generate iridal textures, while Zuo and Schmid [61] presented an anatomy-based method to synthesize iris images. More recently, Wood *et al.* [59] used head scan geometry, image-based techniques and texture maps to build a collection of dynamic eye-region models for the purpose of training computer vision systems.

1.1.2 CUDA Supported Applications

Historically, the GPU has been used for the acceleration of the 3D rendering pipeline (*e.g.*, [33, 53, 52]). Phases of this pipeline are manipulated with shader programs. These shader programs perform floating point operations on large sets of vertex and pixel data. All vertices and pixels are independent from one another, which allows the GPU to execute operations on these data constructs in parallel. GPU designers accomplish this by using the SIMD (Single Instruction, Multiple Data) model of computation. Under this model, a single instruction is executed on many different data points simultaneously [17]. GPUs are designed with many more transistors devoted to floating point data processing instead of flow control, scheduling and data caching [43]. This allows GPUs to be orders of magnitude faster than CPUs for problems with a high level of data parallelism. Such a capability gave rise to general purpose computing on the GPU (GPGPU). In addition, developers without computer graphics programming knowledge have been able to leverage GPUs for computation through the use of APIs such as CUDA. Accordingly, the CUDA parallel platform and API are being often exploited to enhance the performance of rendering systems and to accelerate simulation algorithms used in different fields.

In computer graphics, for example, CUDA has been employed in the solution of order-independent transparency problems [33] and in the adaptive tessellation of surface primitives [53]. It has also been used in the implementation of a photon transport model [21] and a random walk generation method [41] for global illumination applications. Similarly, examples of CUDA supported applications in the biomedical field include, but are not limited to, the development of a real-time biological tissue deformation model for medical training systems [60], the implementation of a ray-optics framework for the computation of sieve factors associated with blood samples [40] and the deployment of a computational tool for the simulation of radiation propagation within multi-layered tissues [15].

1.2 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 describes our proposed framework for the configuration of the ILIT algorithms on the CUDA platform. Chapter 3 provides the data and procedures employed in the assessment of the proposed framework's efficacy and in the analysis of the effects of key ILIT running parameters on the fidelity of modeled iridal responses. Chapter 4 presents the results of our research and discusses their practical implications. Finally, Chapter 5 concludes the thesis and outlines directions for future work.

Chapter 2

Framework Description

In this chapter, we describe the steps involved in the reconfiguration of the ILIT algorithms on the CUDA platform as well as the GPU optimizations performed to enhance the efficiency of the proposed framework. Initially, for completeness, we concisely review the main characteristics of the ILIT model [30] and outline relevant background information about CUDA.

2.1 ILIT Model Overview

Within the ILIT ray-optics formulation, light propagation and attenuation within the iridal tissues are simulated through random walks [29, 30]. The states of these stochastic processes are represented by two groups of interfaces. The first group corresponds to interfaces between surrounding media (air, tear film, cornea and aqueous humor) and the iris. The second group corresponds to interfaces between adjacent iridal tissues, namely the anterior border layer (ABL), the stromal layer (SL) and the iris pigment epithelium (IPE).

A ray traversing from one iridal layer to the next must pass the interface between these layers. At these interfaces, the rays can be reflected or transmitted. The results of these interactions are associated with the random walks' transition probabilities. Light scattering events affecting the direction of propagation of a given ray may also trigger the transition from one state to another. Light absorption events, which are determined by the presence of pigments in the iridal tissues, provide the termination probabilities for the random walks. The absorption events are simulated considering the probability of absorption of light (ray) traveling a certain distance at a certain wavelength in the medium. This probability, in turn, is computed taking into account the spectral absorption coefficients and concentrations of the main pigments (melanins, hemoglobins, lutein, zeaxanthin and bilirubin) found within the iridal layers.

Although each ray travelling within the iridal tissues is associated with a wavelength, it is assumed that the energies of different wavelengths are decoupled. As a result, the random walk performed by a given ray can be simulated independently of the random walks performed by the other rays. Hence, each random walk simulation can potentially be executed on a separate CUDA thread with minimal synchronization overhead.

2.2 CUDA Background

The CUDA platform follows a heterogeneous programming model. In the following discussion, when referring to the CPU and RAM, we will use the word *host* for brevity, and to the GPU and VRAM (Video RAM), the word *device*. Although CUDA can be used to accelerate applications developed in different languages, we elected to employ the CUDA C/C++ API in the proposed framework. Code accelerated through CUDA is encapsulated within functions known as kernels. These functions are called from a host process and are executed in parallel by a specified number of threads on the device. The threads, in turn, are organized in a dimensional construct known as an execution grid. A 2D execution grid contains 3D blocks of threads. Blocks are distributed evenly across the SM (Streaming Multiprocessor) units on the device. Threads in each block are scheduled and executed in groups of 32, which is known as a thread warp [10].

Thread synchronization may only occur for threads that are part of the same block. Device memory is separate from host memory, and it is up to the host process to manage allocations and transfers. It is worth noting that CUDA v6, used in this work, introduces unified memory, which allows the video driver to manage all the memory transfers and allocations automatically. When using unified memory, however, one may fall short of achieving the maximum possible performance [24]. For this reason, we elected to perform all allocations manually during the implementation of the proposed framework in order to minimize runtime.

In the following paragraphs, we outline the relevant characteristics of the main components of CUDA's memory hierarchy employed in this research, namely the registers, the local memory, the shared memory and the global memory. The reader interested in more detailed information about all components of this hierarchy is referred to the work by Ryoo *et al.* [50].

Registers are the fastest type of memory that can be used by CUDA threads. Each SM unit has a register file. The available registers are split evenly amongst all threads running on an SM unit. Variables that do not fit into register space spill over to local memory. Similar to registers, this memory is private to a thread. However, local memory is held in VRAM, which is expensive to access (200-300 clock cycles) [50]. As a result, excessive register spill over can be detrimental to performance.

The shared memory is a fast on-chip cache that is declared and initialized within kernel code and has the scope of a thread block. Since all threads within the block have read/write access to this memory, synchronization may be necessary.

The global memory is allocated and initialized outside the kernel by the host process. All data required for a given application resides in this memory at the beginning of kernel execution. It is implemented using VRAM. Data is copied into the global memory from the host, and a pointer to this memory is passed into the kernel function where it can be accessed by running threads.

2.3 Reconfiguration of ILIT Algorithms on CUDA

The GPU version of ILIT is implemented on CUDA as follows. For the most part, the ILIT random walk algorithms were unchanged from their original (C++) CPU-based implementation [36]. However, several adaptations had to be made in order to run the algorithms on the GPU. A specific keyword (represented by `_device_`) was added before all functions that are called during the execution of the algorithms. This keyword turns any arbitrary C++ function into a CUDA kernel, and allows it to be called from any other CUDA kernel.

Figure 2.1 outlines the execution flow of the GPU-based ILIT implementation on CUDA. First, VRAM is allocated and initialized on the GPU for relevant objects including specimen characterization parameters (Chapter 3), spectral absorption coefficients and simulation results. Next, the random number generator (RNG) is initialized and seeded on the GPU for all threads and blocks through the setupRNG kernel. In this work, we used the XORWOW RNG engine provided by the CUDA API [44].

All CUDA kernel launches (executions of kernel functions from the host) have a specific syntax preceding them. This syntax (represented by <<< i,j >>>) specifies the set of launch variables, where i and j correspond to the number of blocks and the number of threads in each block, respectively. The values assigned to i and j are hardware and task dependent. The selection of these values during this work is discussed in the next section.

Once the GPU completes the RNG initialization process, the runModel kernel (responsible for the execution of ILIT's random walk algorithms) is called for each wavelength considered in the simulation. Finally, after the GPU finishes running all kernels, the results are copied back from VRAM and memory is deallocated.

The main operation aspects of the runModel kernel can be described as follows. The CUDA API provides built-in threadIDx variables that enable a thread to determine its location within a block. This allows the programmer to differentiate between different

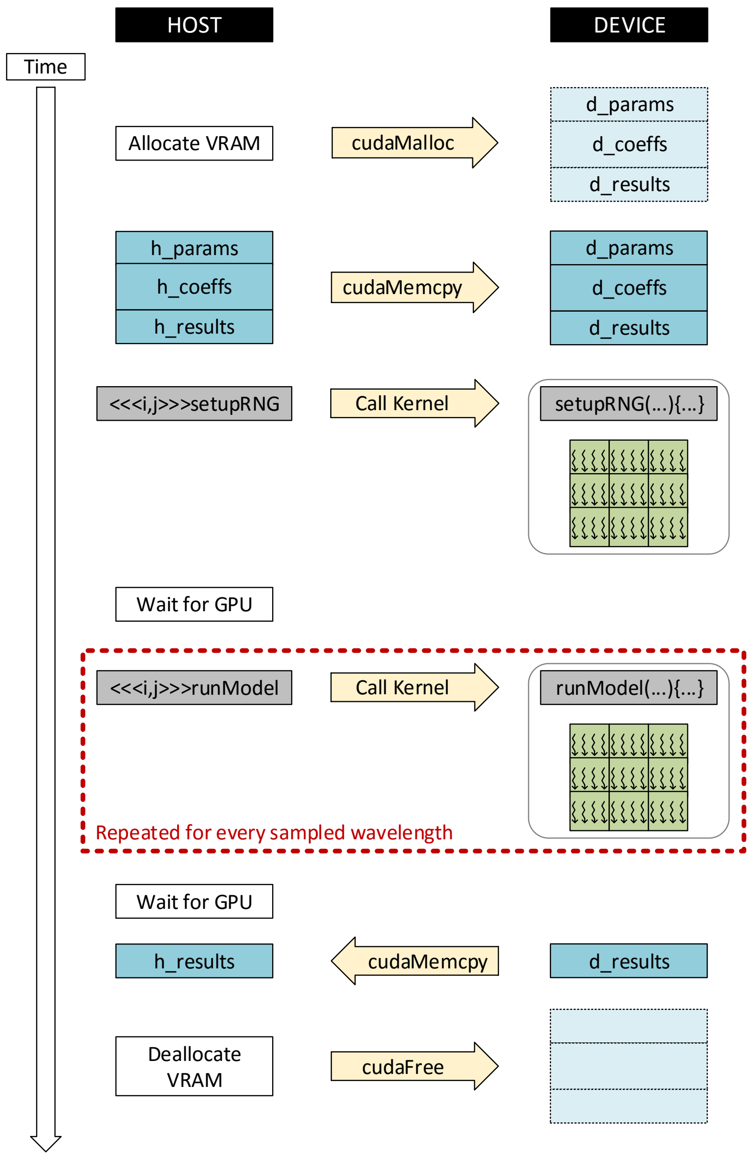


Figure 2.1: Diagram depicting the execution flow of the GPU-based implementation of ILIT on the CUDA platform, including details about the communication between the CPU (host) and GPU (device). Blue, gray and green boxes represent memory allocations, kernel function calls and parallel GPU execution, respectively. Yellow arrows represent CUDA API calls. The kernels setupRNG and runModel are responsible for setting up the random number generator and executing light transport simulations, respectively. The objects *_params*, _coeffs and *_results* correspond to the specimen characterization parameters, spectral absorption coefficients and simulation results, respectively.

threads within the same block. The first thread within a block initializes and copies the objects (specimen characterization parameters and spectral absorption coefficients) from device memory to shared memory, and initializes a model object. This operation is performed in each block. Once this is done, all j threads within a block execute the iridal light transport simulations. The purpose of using shared memory is to minimize data transfer between VRAM and SM units. As described in the previous section, shared memory is implemented on chip (GPU die). Since threads do not need to modify parameters or coefficients, there is no synchronization overhead in this case, *i.e.*, only reading accesses are performed.

2.4 GPU-Specific Optimizations

In this section, we address the GPU-specific optimizations and considerations taken during the implementation of the GPU version of ILIT. This version was developed using NVIDIA's GTX 980, which has been successfully employed to enhance the performance of shading pipelines (*e.g.*, [52]).

First, we selected the most appropriate combination of launch variables. The GTX 980 is able to execute blocks with up to 1024 threads in each block. Furthermore, the number of threads per block should be a multiple of the thread warp size (32) to maximize hardware utilization [42]. Through empirical testing (Appendix C), we determined that a block size of 32 resulted in the fastest runtime for our simulation. Runtimes got progressively worse as block size was increased up to 1024. Ultimately, we employed 3200 blocks (variable i), with each block containing 32 threads (variable j). This combination yields roughly 10^5 threads, which is the number of sample rays used to obtain asymptotically-convergent simulated spectrophotometric readings [4]. Increasing the thread count to higher numbers (*e.g.*, 10^6) caused the XORWOW RNG engine initialization to become slow, limiting potential speedups. Hence, for simulations employing such a higher number of sample rays (usually a multiple, denoted by k, of 10^5), we run these simulations k times on each thread.

Second, all double precision floats in the CPU-based implementation of ILIT were replaced with single precision floats. This was motivated by the fact that consumer GPUs tend to have much faster single precision performance compared to double precision. A GTX 980 is capable of executing $32\times$ more single precision floating point instructions than double precision floating point instructions in a single clock cycle [43]. In addition, all math functions (cos, sin, log ...) were replaced by intrinsic single-precision functions provided by the CUDA API. Although these functions are less precise than their C++ counterparts, they use fewer instructions. As we demonstrate in Chapter 4, the use of single precision floating point functions is sufficient for obtaining simulation results with the same level of convergence and fidelity observed in the results provided by the (C++) CPU-based implementation of ILIT.

Finally, we optimized thread occupancy, which is defined as the ratio between the number of currently executing thread warps per SM unit to the maximum number of thread warps. Higher thread occupancy allows for higher hardware utilization since memory access latency can be compensated by executing a different warp during memory access stalls [42]. This optimization was attained by selecting an appropriate number of registers to be allocated for a thread. A GTX 980 can have up to 2048 threads in-flight on a single SM unit, assuming every thread uses no more than 32 registers. Setting the number of registers to 32, however, caused excessive register spillover to occur, increasing global memory access. Again through empirical testing (Appendix D), we determined that, by using 64 registers, we can improve performance (less time spent accessing global memory) at the expense of having a smaller number of active threads. We found this trade-off to be acceptable and consistent with NVIDIA's performance guidelines [42]: "Higher occupancy does not always equate to higher performance - there is a point above which additional occupancy does not improve performance. However, low occupancy always interferes with the ability to hide memory latency, resulting in performance degradation." Hence, we have allocated 64 registers for each thread in this work.

Chapter 3

Experimental Setup

In our experiments to assess the efficacy of the proposed framework and to find a practical balance between appearance fidelity and performance, we considered three specimens with distinct levels of pigmentation. These specimens, henceforth referred to as lightly pigmented, moderately pigmented and darkly pigmented, are characterized by the biophysical parameters depicted in Tables 3.1 and 3.2. The values assigned for these parameters correspond to actual biophysical data provided in the scientific literature. The reader interested in more detailed information about their sources is referred to the original publications describing the ILIT model [30, 29]. Similarly, for the spectral absorption coefficients of the pigments considered in our implementations of ILIT, we used measured data [29, 37] also reported in the scientific literature. We note that the parameters depicted in Table 3.1 were those effectively modified to characterize the three selected specimens, while the parameters listed in Table 3.2 were kept fixed during our simulations.

ILIT Parameters	Darkly Pigmented	Moderately Pigmented	Lightly Pigmented
Eumelanin mass	2.53E-2 mg	6.33E-3 mg	1.00E-4 mg
Pheomelanin mass	6.90E-3 mg	1.75E-3 mg	3.00E-5 mg
Eumelanin in the ABL	90%	50%	10%
Pheomelanin in the ABL	90%	50%	10%
Blood in the SL	7%	4%	1%

Table 3.1: Three sets of ILIT parameters employed to characterize the distinct iridal specimens considered in this work.

We remark that the ILIT model can provide radiometric readings for distinct illumination and collection geometries. For example, one can obtain bidirectional reflectance quantities by recording the direction of the outgoing rays using a virtual goniophotometer, and directional-hemispherical reflectance quantities by integrating the outgoing rays with respect to the collection hemisphere using a virtual spectrophotometer [4, 29]. The latter group of quantities was used in our analyses of the sensitivity of modeled readings to changes in the ILIT key running parameters, namely spectral resolution (denoted by s_r) and the number of sample rays (denoted by n_r). More specifically, we have computed iridal directional-hemispherical reflectance curves within the spectral region between 400-700 nm for the selected specimens (Figure 4.1). In the computation of these curves, we considered an angle of incidence of $0°$ and distinct values for s_r and n_r. We note that the spectral iridal reflectances computed for a given specimen tend to converge to specific values as n_r increases. We employed the curves associated with these values as the fidelity reference curves in our analyses. To obtain these reference curves, we used s_r equal to 1 nm (301 sample wavelengths) and n_r equal to 10^6 rays. It is also worth mentioning that the reflectance curves depicted in this work match those that can be obtained through the online

ILIT Parameters	Value
Lutein mass	4.03E-6 mg
Zeaxanthin mass	1.54E-6 mg
ABL thickness	0.005675 cm
SL thickness	0.02855 cm
Volume fraction of fibroblasts in the ABL	0.33333
Volume fraction of collagen fibrils in the SL	0.9069
Radius of collagen fibers	30.0 nm
Concentration of hemoglobin in the blood	147 g/L
Percentage of oxygenated hemoglobin	50%
Concentration of carboxyhemoglobin in the blood	1.5 g/L
Concentration of methemoglobin in the blood	1.5 g/L
Concentration of sulfhemoglobin in the blood	0 g/L
Concentration of bilirubin in the blood	0.003 g/L
Refractive index of the fibroblasts	1.42
Refractive index of the ocular base	1.35
Refractive index of the air	1.0003
Refractive index of the tear film	1.337
Refractive index of the cornea	1.3771
Refractive index of the aqueous humour	1.336
Refractive index of the IPE	1.35

Table 3.2: Set of ILIT parameters kept fixed for the simulations that resulted in the modeled reflectance curves depicted in this work.

version of the original implementation of the ILIT model [36] using the same parameter values.

We also generated iris swatches to complement our analyses (Figures 4.2 and 4.3). The swatches' chromatic attributes were obtained from the convolution of the illuminant spectral power distribution spectrum, the modeled reflectance data and the broad spectral response of the human photoreceptors [25]. This last step was performed employing a standard XYZ to sRGB conversion procedure as outlined in Appendix A. Unless otherwise stated, the iris swatches and the additional images presented in this work were rendered

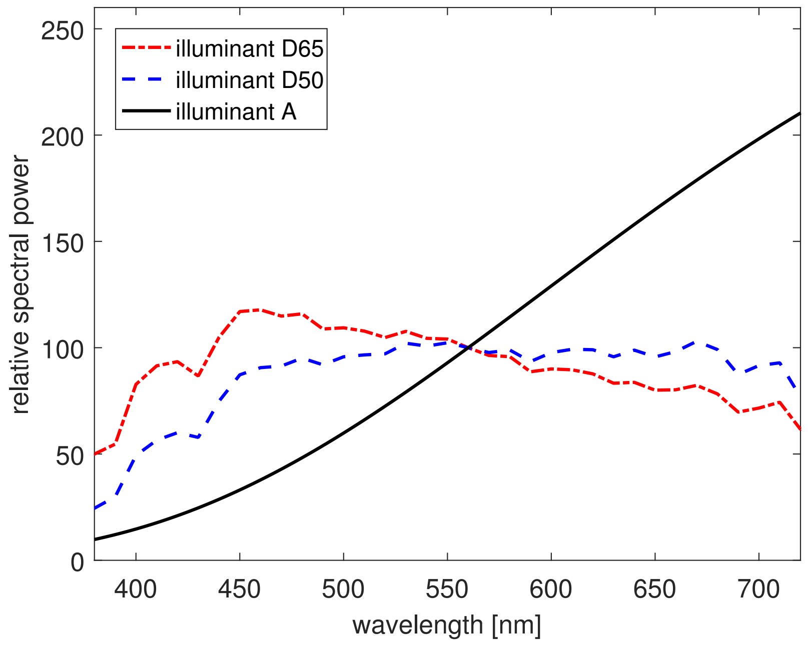


Figure 3.1: Relative spectral power distribution of three CIE standard illuminants, namely D65, D50 and A [45], considered in our investigation.

considering the CIE standard D65 illuminant (Figure 3.1).

Surely, the spectral responses of a given iridal specimen can vary from one measurement point to another. Although one could modulate the values of the biophysical parameters affecting its chromatic attributes according to the characteristics of its underlying iridal structure, this process would require detailed information about the specimen's morphology (*e.g.*, tissue thickness variations) which is not readily available. Hence, for practical purposes, we have elected to use iridal texture variations to modulate the lightness of the colors derived from the modeled reflectance data. For consistency, we employed the same iridal texture for all swatches.

To assess appearance fidelity, we initially visually compared swatches (Figure 4.2) rendered using modeled reflectance data computed considering distinct spectral resolutions and n_r equal to 10^6 with their respective swatch reference, which was rendered using the specimen's fidelity reference curve. We assigned to s_r integer factors of 300 nm (10, 30 and 50 nm) to ensure an uniform sampling of the visible spectral region of interest (from 400 to 700 nm).

Besides visual inspection, we also employed a device-independent CIE-based metric to compare modeled results obtained using different combinations of simulation running parameters. More specifically, we computed the CIELAB differences [54] between modeled chromatic attributes and their corresponding reference values. These differences are defined as

$$\Delta E^*_{ab} = \sqrt{(L^*_1 - L^*_2)^2 + (a^*_1 - a^*_2)^2 + (b^*_1 - b^*_2)^2}, \tag{3.1}$$

where L^*, a^* and b^* correspond to the CIELAB color space dimensions calculated for the modeled chromatic attributes (indicated by subscript 1) and their corresponding reference values (indicated by subscript 2) using standard colorimetric formulas [8] and setting the white point to the CIE D65 standard illuminant (Figure 3.1). The conversion of XYZ values to $L^*a^*b^*$ coordinates is outlined in Appendix B.

We then repeated the comparisons for swatches (Figure 4.3) rendered using modeled reflectance data computed considering different numbers of sample rays and the spectral resolution determined in the previous set of comparisons. The purpose of these two sets of comparisons was to find a practical balance between appearance fidelity and performance (reduced computational time). By performing the comparisons in this order, we have attributed more importance to s_r. This choice was motivated by the fact that this parameter has a larger impact on the qualitative reproduction of appearance attributes.

To assess the performance gains associated with the execution of iridal light transport simulations on the CUDA platform, we have compared the frameworks GPU-based implementation of ILIT with a multithreaded CPU-based implementation of this model. These two implementations are henceforth referred to as ILIT-GPU and ILIT-CPU, respectively. Both implementations were tested on a server running Ubuntu 14.04 LTS with 2 hyperthreaded Intel Xeon E5-2630 2.4 GHz CPUs (comprising a total of 16 cores, 32 threads), 64GB of RAM and a GTX 980 GPU (whose relevant features have been outlined in Chapter 2.4).

Chapter 4

Results and Discussion

In this chapter, we compare the GPU- and CPU-based implementations of the ILIT model and discuss the effects of adjusting key simulation parameters on the fidelity and computational cost of the modeled results. We also address issues regarding the usability of the proposed framework and the reproducibility of our research findings.

4.1 Analysis of Fidelity and Performance Trade-offs

Initially, we compared the modeled reflectance curves provided by the ILIT-GPU and ILIT-CPU implementations in order to assess the correctness of the reconfigured ILIT algorithms employed by the proposed framework. As outlined in Chapter 2.4, ILIT-GPU is expected to provide results with a lower precision than ILIT-CPU since the former employs single-precision floating point arithmetic to maximize performance. The graphs presented in Figure 4.1 demonstrate, however, that the effects of precision differences are negligible in comparison with random fluctuations associated with the stochastic nature of the ILIT model. Such random fluctuations are expected, especially when one sets a relatively low value for n_r. However, as n_r is increased, these fluctuations are reduced accordingly. As a result, the curves computed by both ILIT implementations tend to show a close agreement.

We then proceeded to assess the computational gains that can be provided by the proposed framework. As illustrated by the data presented in Table 4.2, ILIT-GPU outperformed the multithreaded CPU-based implementation of ILIT. We note that ILIT-GPU has an initialization time of 200 ms, which was included in the figures depicted in Table 4.2. This time corresponds to a call to the setupRNG kernel (Chapter 2.3) to seed and initialize the state of the random number generator on the GPU.

For completeness, we also provide the corresponding speedup factors in Table 4.1. It is worth noting that the recorded speedup factors of the ILIT-GPU with respect to a single threaded CPU-based implementation of ILIT were on the order of 100×.

Number of Rays	Darkly Pigmented	Moderately Pigmented	Lightly Pigmented
10^3	4.57×	4.15×	3.94×
10^4	7.10×	7.90×	8.19×
10^5	16.95×	17.50×	15.44×
10^6	19.61×	14.62×	12.24×

Table 4.1: Speedup factors of the ILIT-GPU with respect to the multithreaded CPU-based implementation of the ILIT model. These factors refer to the iridal light transport simulations performed for the selected specimens, which were carried out considering distinct numbers of rays and a spectral resolution of 1 nm.

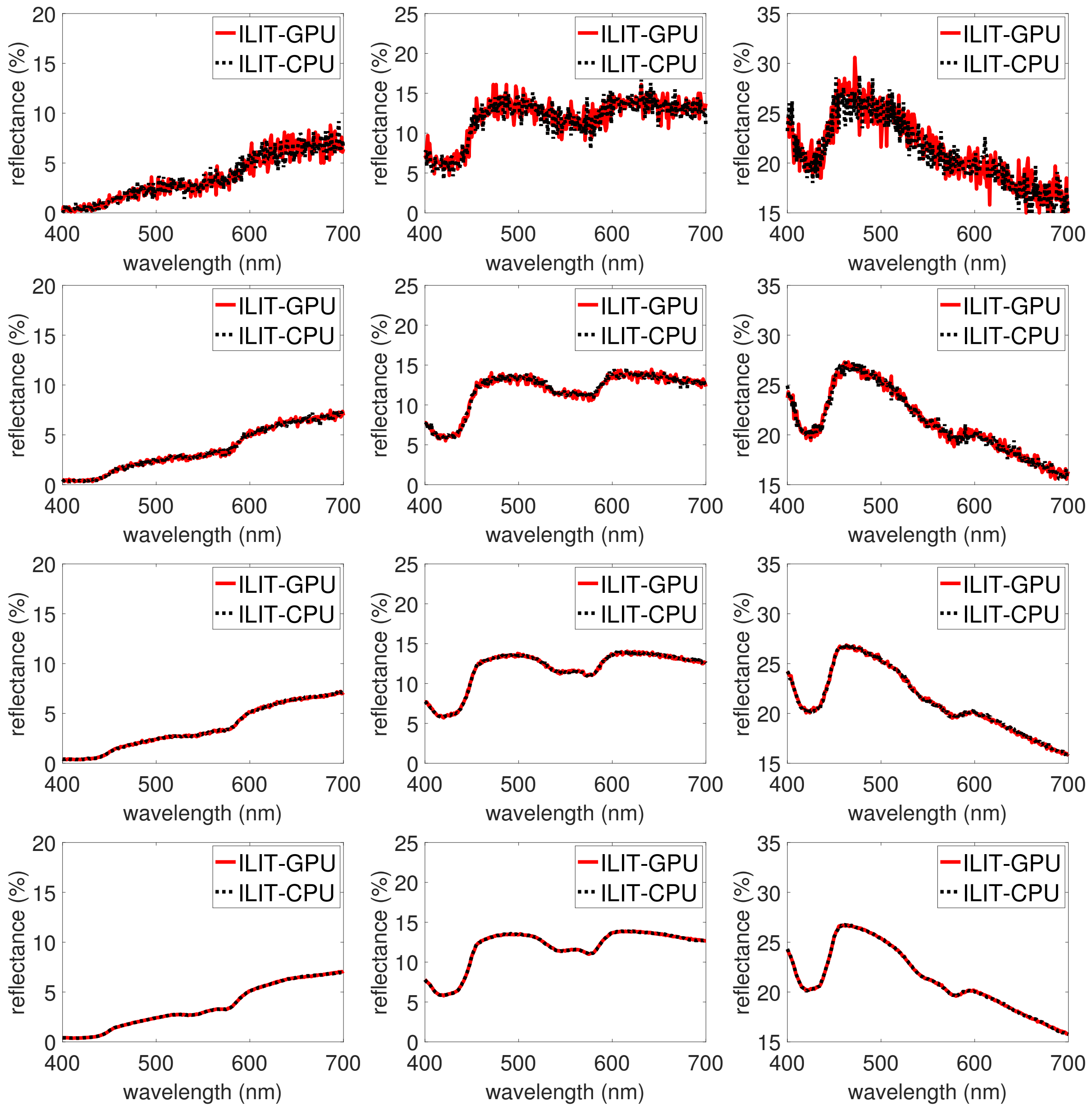


Figure 4.1: Comparisons of modeled reflectance curves computed for the selected iris specimens using the CPU- and GPU-based implementations of ILIT. From left to right: darkly pigmented, moderately pigmented and lightly pigmented specimen, respectively. The curves were computed using distinct numbers of rays. From top to bottom: 10^3, 10^4, 10^5 and 10^6, respectively. All curves were computed considering a spectral resolution of 1 nm.

Number of Rays	Darkly Pigmented	
	ILIT-GPU	ILIT-CPU
10^3	0.39 s	1.80 s
10^4	0.40 s	2.87 s
10^5	0.75 s	12.78 s
10^6	4.04 s	79.33 s
Number of Rays	**Moderately Pigmented**	
	ILIT-GPU	ILIT-CPU
10^3	0.41 s	1.71 s
10^4	0.46 s	3.70 s
10^5	1.12 s	19.62 s
10^6	7.54 s	110.29 s
Number of Rays	**Lightly Pigmented**	
	ILIT-GPU	ILIT-CPU
10^3	0.46 s	1.81 s
10^4	0.54 s	4.49 s
10^5	1.62 s	25.04 s
10^6	12.18 s	149.27 s

Table 4.2: Running times recorded for the light transport simulations performed for the selected specimens using the GPU-based (ILIT-GPU) and the multithreaded CPU-based (ILIT-CPU) implementations of the ILIT model. The simulations were carried out considering distinct numbers of rays and a spectral resolution of 1 nm.

For applications aimed at providing predictable results at interactive rates, one should strive to reduce running time without sacrificing appearance fidelity. With this guideline in mind, we investigated the sensitivity of modeled chromatic attributes to key simulation running parameters, namely s_r and n_r. We started with the former as we describe next.

Traditionally, image synthesis pipelines employ three sample wavelengths, whose selection may vary from one user to another. For comparison purposes, we considered three wavelengths, namely 463 nm, 504 nm and 600 nm, representative of the blue, green and red regions of the light visible spectrum, respectively. As it can be verified by visually inspecting the iris swatches presented in Figure 4.2, the results obtained when one considers only three sample wavelengths may differ markedly from more comprehensive spectral solutions. They also indicate that the results tend to converge to a reference solution as we increase the number of sample wavelengths (from 3 to 7, 11, 31 and 301). However, in order to obtain an image with a high appearance fidelity with respect to this reference solution, one does not necessarily need to resort to an extremely fine, and consequently costly, spectral resolution. For example, as it can also be observed in the images presented in Figure 4.2, using a sampling interval of 30 nm (11 wavelengths), the resulting swatches show no distinguishable visible differences when compared to their respective reference solutions obtained considering an interval of 1 nm (301 wavelengths).

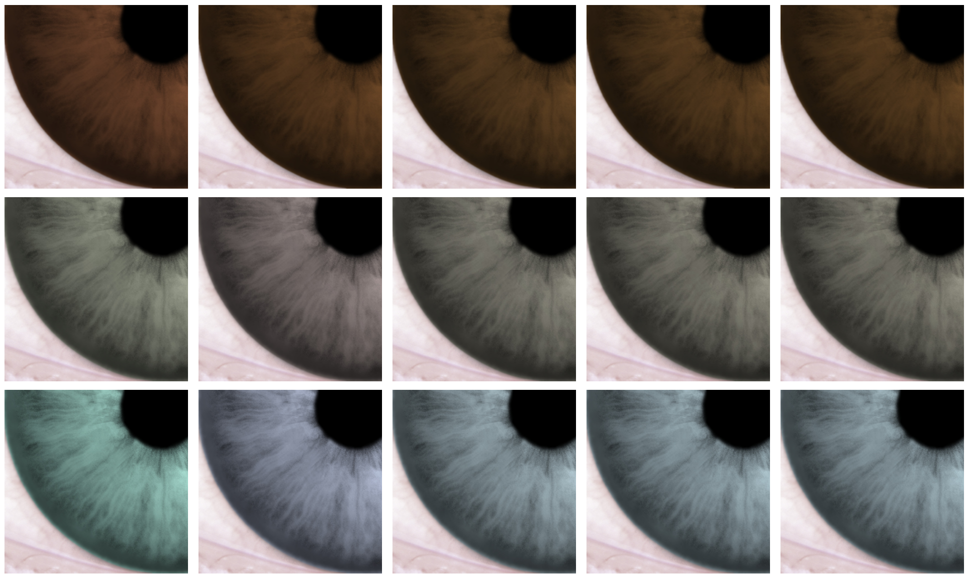


Figure 4.2: Iris swatches rendered using modeled reflectance data computed considering different spectral resolutions. From top to bottom: swatches generated for the darkly pigmented, moderately pigmented and lightly pigmented specimens, respectively. Leftmost column: employing data sampled at three wavelengths (463 *nm*, 504 *nm* and 600 *nm*). Subsequent columns, from left to right: employing data sampled from 400 to 700 *nm* at every 50, 30, 10 and 1 *nm*, respectively. The simulations employed to obtain the modeled data were carried out using 10^6 rays.

Spectral Resolution	Darkly Pigmented	Moderately Pigmented	Lightly Pigmented
50 *nm*	1.59	5.75	6.77
30 *nm*	0.27	0.33	0.38
10 *nm*	0.05	0.11	0.05

Table 4.3: CIELAB ΔE^*_{ab} differences calculated for the selected specimens' swatches (depicted in Figure 4.2) which were generated using modeled data computed considering three distinct spectral resolutions and 10^6 rays. The ΔE^*_{ab} differences were calculated with respect to the specimens' specific reference swatches (also depicted in Figure 4.2, rightmost column) which were generated using modeled data computed considering a spectral resolution of 1 *nm* and 10^6 rays.

In order to quantitatively assess the fidelity of the swatches generated using modeled data obtained considering s_r equal to 10, 30 and 50 nm, we computed their ΔE^*_{ab} differences (Equation 3.1) with respect to the corresponding reference solutions (Table 4.3). As expected, these differences decreased as we reduced the spectral sampling intervals. Moreover, for s_r equal to 30 nm, the ΔE^*_{ab} values are smaller than 2.3 ± 1.3, the experimentally-determined perceptibility threshold for CIELAB chromatic differences [34]. These results reinforced our observation that a sampling interval of 30 nm is sufficient to ensure appearance fidelity.

After determining a suitable value for s_r, we examined the impact of n_r. Again, as it can be verified by visually inspecting the iris swatches presented in Figure 4.3, a combination considering a spectral resolution of 30 nm and 10^4 rays (second column from the left) is sufficient to obtain results which closely agree with their corresponding reference solutions (rightmost column). This aspect is also supported by the ΔE^*_{ab} values (Equation 3.1) computed for these swatches (generated using 10^4 rays and a spectral resolution of 30 nm) with respect to the corresponding reference solutions. More specifically, by comparing these differences (Table 4.4) with the the experimentally-determined perceptibility threshold of 2.3 ± 1.3 [34], it can be verified that this combination of parameter values results in modeled chromatic attributes that, for practical purposes, are indistinguishable from their reference solutions as perceived by a human observer.

Considering the stochastic nature of the ILIT simulations, the CIELAB ΔE^*_{ab} differences may vary from one run to another. Hence, for the sake of transparency, we also computed CIELAB ΔE^*_{ab} differences averaged over 1000 runs (Table 4.5). These averaged values confirmed our previous observations.

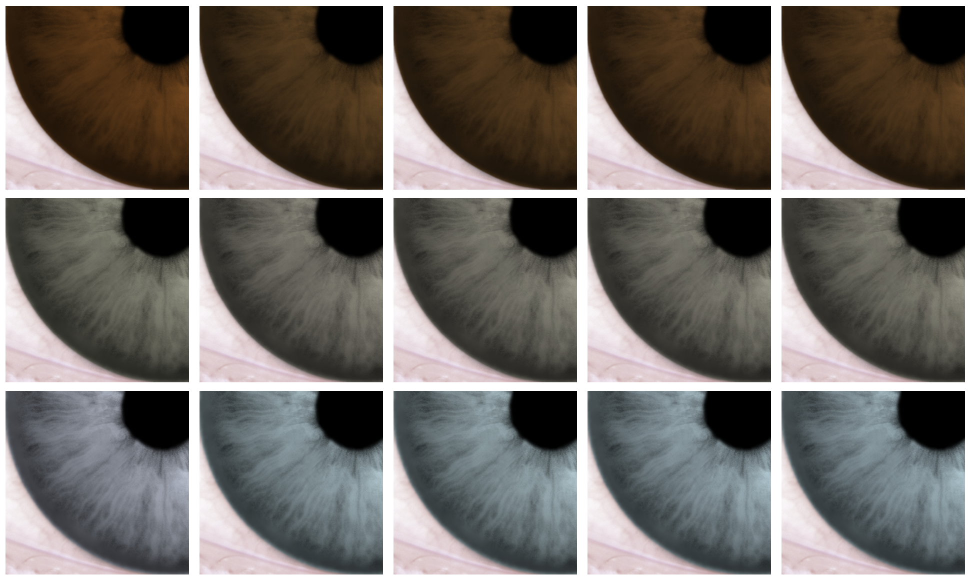


Figure 4.3: Iris swatches rendered using modeled reflectance data computed considering different numbers of sample rays. From top to bottom: swatches generated for the darkly pigmented, moderately pigmented and lightly pigmented specimens, respectively. From the leftmost column to the fourth column: swatches generated using modeled data obtained considering a spectral resolution of 30 nm and 10^3, 10^4, 10^5 and 10^6 sample rays, respectively. Rightmost column: reference solutions obtained considering a spectral resolution of 1 nm and 10^6 rays.

Darkly Pigmented	Moderately Pigmented	Lightly Pigmented
1.16	0.63	0.57

Table 4.4: CIELAB ΔE^*_{ab} differences computed for the selected specimens' swatches (depicted in Figure 4.3, second column from the left), which were generated using 10^4 rays and a spectral resolution of 30 nm. These ΔE^*_{ab} values were computed with respect to the specimens' specific reference swatches (also depicted in Figure 4.3, rightmost column), which were generated using modeled data computed considering a spectral resolution of 1 nm and 10^6 rays.

Darkly Pigmented	Moderately Pigmented	Lightly Pigmented
1.24 ± 0.62	0.98 ± 0.49	0.89 ± 0.42

Table 4.5: Averaged (over 1000 runs) CIELAB ΔE^*_{ab} differences and their respective standard deviations computed for the selected specimens' swatches, generated using 10^4 rays and a spectral resolution of 30 nm. These ΔE^*_{ab} values were computed with respect to the specimens' specific reference swatches (also depicted in Figure 4.3, rightmost column), which were generated using modeled data computed considering a spectral resolution of 1 nm and 10^6 rays.

In terms of performance, as indicated by the data provided in Table 4.6, setting s_r equal to 30 nm and n_r equal to 10^4 rays enables us to reduce the time spent in the simulations to values below a latency time of 16 ms (associated with an update rate of 60Hz) appropriate for interactive applications [18]. We note that these values do not include the GPU initialization time mentioned earlier since this step needs to be performed only once for interactive applications. In other words, it does not need to be repeated for each frame generated using modeled iridal data.

Darkly Pigmented	Moderately Pigmented	Lightly Pigmented
3.72 ms	6.30 ms	9.71 ms

Table 4.6: Running times recorded for iridal light transport simulations performed for the selected specimens using the GPU-based (ILIT-GPU) implementation of the ILIT model with the spectral resolution and number of rays set to 30 nm and 10^4, respectively.

We fully recognize that the generation of realistic-looking human eyes is a challenging process. As such, it requires due attention to be given to all stages of the rendering pipeline, starting with the selection of geometrical models and textures representing the distinct ocular structures. Clearly, these aspects are beyond the scope of this work, which focuses on the iris' spectral responses. However, we remark that these responses play a central role in this process. Hence, we believe that the proposed framework can effectively contribute to the interactive generation of images that are both believable and predictable depictions of the human eye. This aspect is illustrated by the images presented in Figures 1.1, 4.4 and 4.5, which were rendered using modeled iridal data obtained considering the selected combination of simulation running parameters, namely a spectral resolution of 30 nm and 10^4 rays.

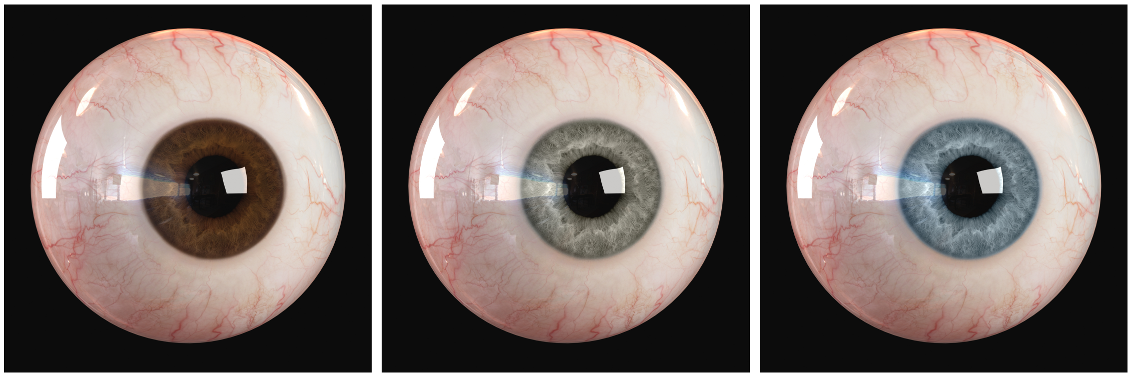


Figure 4.4: Images generated using modeled iridal data computed for the darkly (left), moderately (center) and lightly (right) pigmented iris specimens.

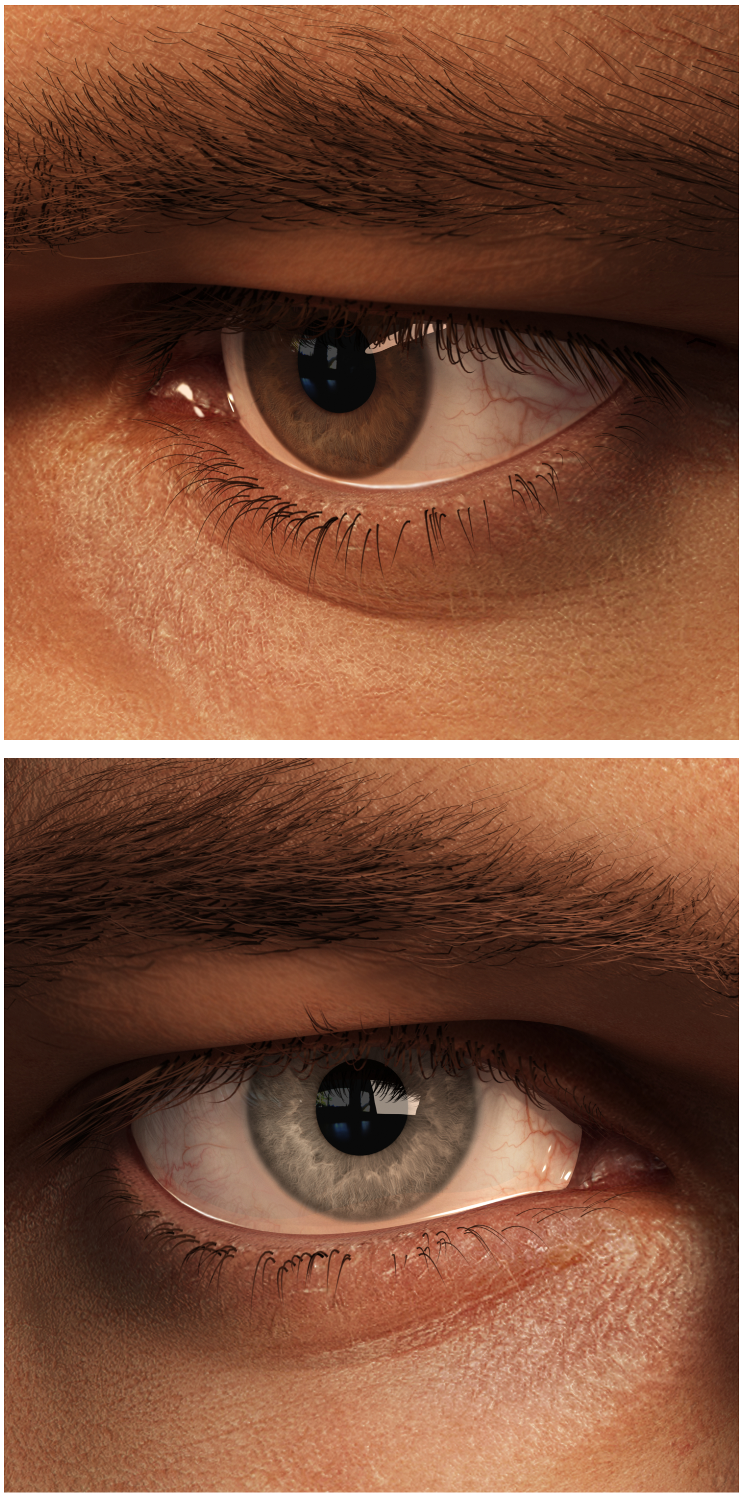


Figure 4.5: Images generated using modeled iridal data computed for the darkly (top) and moderately (bottom) pigmented iris specimens and considering two distinct CIE standard illuminants, namely A (top) and D50 (bottom), whose relative spectral power distributions are provided in Figure 3.1.

4.2 Usability Issues

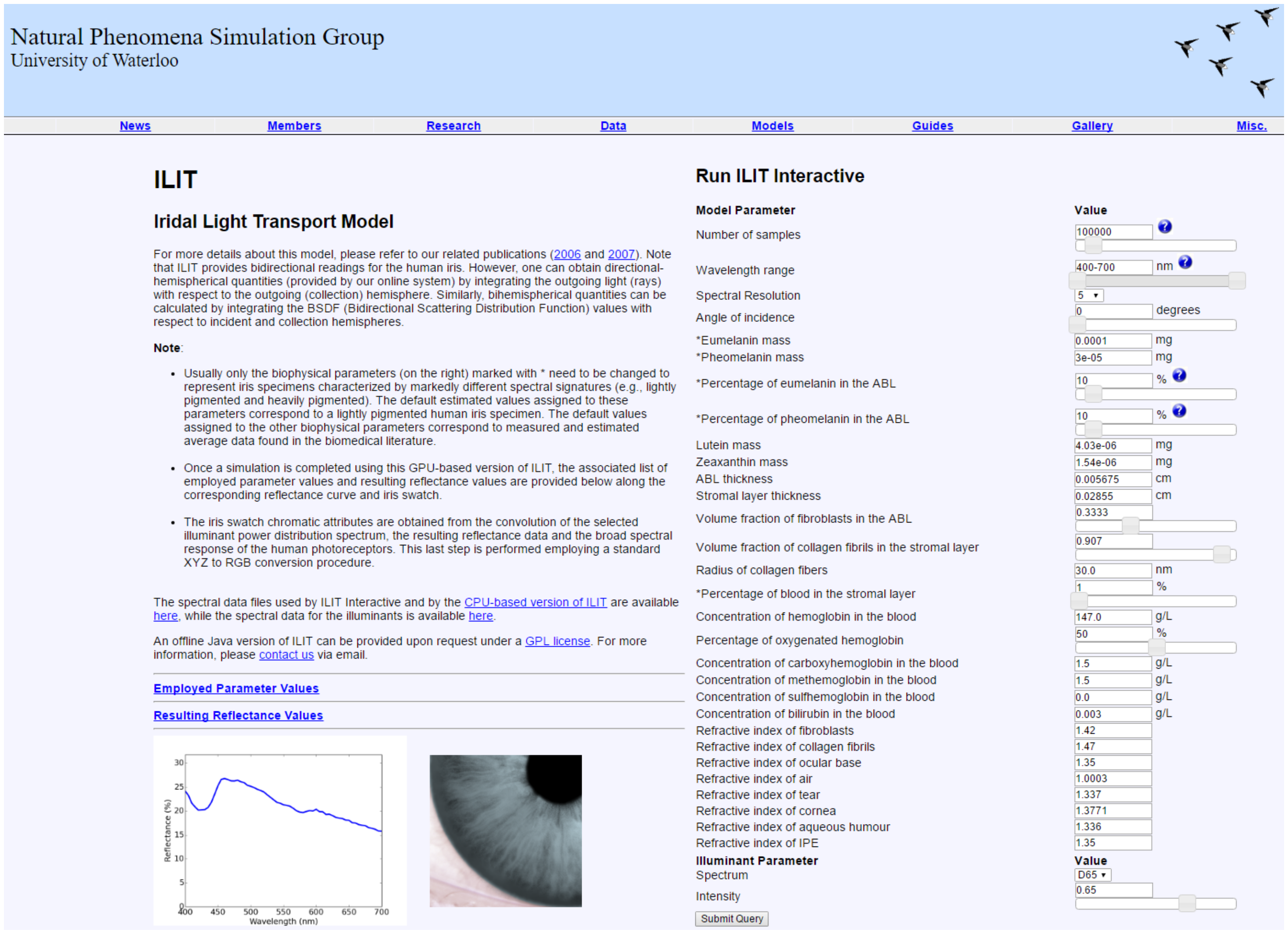
With respect to usability, we note that the proposed framework can be seamlessly integrated into rendering systems that support GPU-based implementations. Alternatively, it can be employed to provide iridal appearance data on demand for image synthesis applications running on different computational environments.

Finally, it is worth stressing that only a small subset of the ILIT parameters (Table 3.1) needs to be modified so that one can obtain markedly distinct chromatic attributes for iridal specimens. This is particularly convenient when one wants to streamline the image generation process. If necessary, however, its detailed parameter space allows experimentation with a wide range of biophysical factors affecting iridal appearance. Hence, besides the support to the efficient rendering of realistic eye images for artistic and entertainment applications, the ILIT-GPU can potentially be employed in educational and scientific applications involving rapid visualizations of iridal appearance variations as illustrated by its online version [38].

4.3 Reproducibility Issues

As part of our investigation, we have made an effort to maximize reproducibility of our findings. Reproducibility is an important aspect of scientific research since it allows other researchers to verify the correctness of the results as well as employed methods. Furthermore, providing executable code and data fosters collaboration and encourages future expansion upon this work [47], which is especially important since light transport models such as ILIT can be useful tools in interdisciplinary research [2].

Accordingly, we have made the ILIT-GPU available online [38] in order to allow other researchers to reproduce the results of our investigation and to employ our model in their studies without having to implement it. The online implementation outputs the spectral reflectance data, spectral reflectance curve as well as an iris swatch for a given set of model parameters. A server with a GTX 980 GPU is used to host the model in order to provide output in a timely manner. Figure 4.6 depicts the interface of the online implementation of ILIT-GPU. In this interface, we provide a runtime breakdown of the model so our claims can be verified.



Natural Phenomena Simulation Group
University of Waterloo

News | Members | Research | Data | Models | Guides | Gallery | Misc.

ILIT

Iridal Light Transport Model

For more details about this model, please refer to our related publications (2006 and 2007). Note that ILIT provides bidirectional readings for the human iris. However, one can obtain directional-hemispherical quantities (provided by our online system) by integrating the outgoing light (rays) with respect to the outgoing (collection) hemisphere. Similarly, bihemispherical quantities can be calculated by integrating the BSDF (Bidirectional Scattering Distribution Function) values with respect to incident and collection hemispheres.

Note:

- Usually only the biophysical parameters (on the right) marked with * need to be changed to represent iris specimens characterized by markedly different spectral signatures (e.g., lightly pigmented and heavily pigmented). The default estimated values assigned to these parameters correspond to a lightly pigmented human iris specimen. The default values assigned to the other biophysical parameters correspond to measured and estimated average data found in the biomedical literature.
- Once a simulation is completed using this GPU-based version of ILIT, the associated list of employed parameter values and resulting reflectance values are provided below along the corresponding reflectance curve and iris swatch.
- The iris swatch chromatic attributes are obtained from the convolution of the selected illuminant power distribution spectrum, the resulting reflectance data and the broad spectral response of the human photoreceptors. This last step is performed employing a standard XYZ to RGB conversion procedure.

The spectral data files used by ILIT Interactive and by the CPU-based version of ILIT are available here, while the spectral data for the illuminants is available here.

An offline Java version of ILIT can be provided upon request under a GPL license. For more information, please contact us via email.

Employed Parameter Values

Resulting Reflectance Values

Run ILIT Interactive

Model Parameter	**Value**	
Number of samples	100000	
Wavelength range	400-700	nm
Spectral Resolution	5	
Angle of incidence	0	degrees
*Eumelanin mass	0.0001	mg
*Pheomelanin mass	3e-05	mg
*Percentage of eumelanin in the ABL	10	%
*Percentage of pheomelanin in the ABL	10	%
Lutein mass	4.03e-06	mg
Zeaxanthin mass	1.54e-06	mg
ABL thickness	0.005675	cm
Stromal layer thickness	0.02855	cm
Volume fraction of fibroblasts in the ABL	0.3333	
Volume fraction of collagen fibrils in the stromal layer	0.907	
Radius of collagen fibers	30.0	nm
*Percentage of blood in the stromal layer	1	%
Concentration of hemoglobin in the blood	147.0	g/L
Percentage of oxygenated hemoglobin	50	%
Concentration of carboxyhemoglobin in the blood	1.5	g/L
Concentration of methemoglobin in the blood	1.5	g/L
Concentration of sulfhemoglobin in the blood	0.0	g/L
Concentration of bilirubin in the blood	0.003	g/L
Refractive index of fibroblasts	1.42	
Refractive index of collagen fibrils	1.47	
Refractive index of ocular base	1.35	
Refractive index of air	1.0003	
Refractive index of tear	1.337	
Refractive index of cornea	1.3771	
Refractive index of aqueous humour	1.336	
Refractive index of IPE	1.35	
Illuminant Parameter	**Value**	
Spectrum	D65	
Intensity	0.65	

Figure 4.6: Screenshot of the online implementation of ILIT-GPU. Results are provided to the user in an interactive manner.

Chapter 5

Conclusion and Future Work

Distinct trends can be observed in rendering research. At one end of the spectrum, efforts have been directed toward the fast generation of believable images for a large variety of artistic and entertainment applications, from movie effects to video games. At the other end of the spectrum, different initiatives have been aimed at the generation of images with the highest possible level of biophysical correctness. Among these initiatives, one can include the development of predictive appearance models for a large variety of natural and man-made materials.

The simultaneous realization of high levels of performance and correctness is usually problematic since researchers often have to sacrifice one to get the other. In this thesis, we attempted to conciliate these seemingly conflicting goals by proposing a framework that incorporates advances at both ends of the rendering research spectrum. More specifically, by reconfiguring the light transport algorithms of the ILIT model on the CUDA platform, we were able to examine the trade-offs involving different combinations of key simulation running parameters. Our findings indicate that high-fidelity appearance attributes can be obtained through first-principles light transport algorithms executed at interactive rates. Accordingly, these attributes can be employed in the efficient generation of iridal images that are not only believable, but also predictable.

As future work, we plan to employ the proposed framework in the investigation of physiological phenomena affecting iridal appearance attributes. We also intend to extend our research to other organic materials. It is worth noting that some of the existing stochastic light transport models developed for distinct human tissues, such as human skin and blood, may differ significantly in terms of their specific simulation approaches. Nonetheless, we believe that such differences would not represent insuperable obstacles. Hence, we expect that similar combinations of simulation running parameters leading to the generation of predictable images at interactive rates can also be found for these materials.

References

[1] Image processing toolbox. http://www.mathworks.com/products/image/. Accessed: 2016-08-24.

[2] G.V.G. Baranoski, T. Dimson, T.F. Chen, B. Kimmel, D. Yim, and E. Miranda. Rapid dissemination of light transport models on the web. *IEEE computer graphics and applications*, 32(3):10–15, 2012.

[3] G.V.G. Baranoski and M.W.Y. Lam. Qualitative assessment of undetectable melanin distribution in lightly pigmented irides. *Journal of Biomedical Optics*, 12(3):(030501):1–3, August 2007.

[4] G.V.G. Baranoski, J.G. Rokne, and G. Xu. Virtual spectrophotometric measurements for biologically and physically-based rendering. *The Visual Computer*, 17(8):506–518, 2001.

[5] P. Bérard, D. Bradley, M. Nitti, T. Beeler, and M.H. Gross. High-quality capture of eyes. *ACM Transactions on Graphics*, 33(6):223–1, 2014.

[6] S. Bianco and R. Schettini. Two new Von Kries based chromatic adaptation transforms found by numerical optimization. *Color Research & Application*, 35(3):184–192, 2010.

[7] P. Blum. Reflectance spectrophotometry and colorimetry. *Physical Properties Handbook, Ocean Drilling Program*, 1997. Chapter 7.

[8] D.H. Brainard. Color appearance and color difference specification. *The science of color*, 2:191–216, 2003.

[9] Y. Cai, C. Chui, Y. Wang, Z. Wang, and J.H. Anderson. Parametric eyeball model for interactive simulation of ophthalmologic surgery. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2001*, pages 465–472. Springer, 2001.

[10] S. Chei, M. Boyer, J. Meng, D. Tarjan, J.W. Sheaffer, and K. Skadron. A performance study of general-purpose applications on graphics processors using CUDA. *J. Parallel Distrib. Comput.*, 68:1370–1380, 2008.

[11] M. Chiang and G. Fyffe. Realistic real-time rendering of eyes and teeth. Technical Report ICT-TR-01-2010, University of Southern California, 2010.

[12] International Electrotechnical Commission. Multimedia systems and equipment–colour measurement and management–part 2-1: Colour management–default RGB colour space–sRGB. Technical report, IEC 61966-2-1, 1999.

[13] F.C. Delori, C.K. Dorey, and K.A. Fitch. Characterization of ocular melanin by iris reflectometry. *Investigative Ophthalmology and Visual Science*, 32(1-4):1144, 1991.

[14] E. d'Eon and D. Luebke. Efficient rendering of human skin. In H. Nguyen, editor, *GPU Gems 3*, pages 293–347. Addison-Wesley, 2007.

[15] A. Doronin and I. Meglinski. Online object oriented Monte Carlo computational tool for needs of the biomedical optics. *Biomedical Optics Express*, 2(9):2461–2469, 2011.

[16] J. Dorsey, H. Rushmeier, and F. Sillion. *Digital Modeling of Material Appearance.* Morgan Kaufmann/Elsevier, 2007.

[17] Y. Drew. A closer look at GPUs. *Communications of the ACM*, 51(10):50–57, 2008.

[18] S.R. Ellis, K. Mania, B.D. Adelstein, and M.I. Hill. Generalizeability of latency detection in a variety of virtual environments. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, pages 2632–2636, 2004.

[19] A. Fournier. The tiger experience. In *Workshop on Rendering, Perception and Measurement.* Cornell University, USA, 1999.

[20] G. Francois, P. Gautron, G. Breton, and K. Bouatouch. Image-based modeling of the human eye. *IEEE Transactions on Visualization and Computer Graphics*, 15(5):815–827, 2009.

[21] R. Geist and J. Westall. Lattice-Boltzmann lighting models. *GPU GEMS*, 4, 2011.

[22] D.P. Greenberg, J. Arvo, E. Lafortune, K.E. Torrance, J.A. Ferwerda, B. Walter, B. Trumbore, P. Shirley, S. Pattanaik, and S. Foo. A framework for realistic image synthesis. In *SIGGRAPH, Annual Conference Series*, pages 477–494, 1997.

[23] D.C. Gross. Report from the fidelity implementation study group. In *Simulation Interoperability Workshop, Simulation Interoperability and Standards Organization*, Orlando, FL, USA, 1999. Paper 99S-SIW-167.

[24] M. Harris. Unified memory in CUDA 6, 2013.

[25] J.J. Hopfield. Olfaction and color vision: more in simpler. In N.P. Ong and R.N. Bhatt, editors, *More is Different: Fifty Years of Condensed Matter Physics*, pages 269–284, Princeton, NJ. USA, 2001. Princeton University Press.

[26] R.W.G. Hunt. *Measuring Colour.* Ellis Horwood Limited, Chichester, England, 2nd edition, 1991.

[27] J. Jimenez, E. Danvoye, and J. Pahlen. Separable subsurface scattering and photorealistic eyes rendering, 2012. In: Advances in Real-Time Rendering in Games, SIGGRAPH Course.

[28] J. Jimenez, K. Zsolnai, A. Jarabo, C. Freude, T. Auzinger, X. Wu, J. Pahlen, M. Wimmer, and D. Gutierrez. Separable subsurface scattering. *Computer Graphics Forum*, 34(6):188–197, 2015.

[29] M.W.Y. Lam. ILIT: A predictive light transport model for the human iris. Master's thesis, University of Waterloo, Canada, August 2006.

[30] M.W.Y. Lam and G.V.G. Baranoski. A predictive light transport model for the human iris. *Comp. Graph. Forum*, 25(3):359–368, 2006.

[31] S. Lee, J. Badler, and N. Badler. Eyes alive. *ACM Transactions on Graphics*, 21(3):637–644, 2002.

[32] A. Lefohn, B. Budge, P. Shirley, R. Caruso, and E. Reinhard. An ocularist's approach to human iris synthesis. *IEEE Computer Graphics and Applications*, 23(6):70–75, 2003.

[33] F. Liu, M. Huang, X. Liu, and E. Wu. Cuda renderer: A programmable graphics pipeline. In *ACM SIGGRAPH ASIA 2009 Sketches*, page 34. ACM, 2009.

[34] M. Mahy, L. Van Eycken, and A. Oosterlinck. Evaluation of uniform color spaces developed after the adoption of CIELAB and CIELUV. *Color Research and Application*, 19(2):105–121, 1994.

[35] S. Makthal and A. Ross. Synthesis of iris images using Markov random fields. In *Signal Processing Conference, 2005 13th European*, pages 1–4. IEEE, 2005.

[36] Natural Phenomena Simulation Group (NPSG). *Run ILIT Online*. School of Computer Science, University of Waterloo, Ontario, Canada, 2013. http://www.npsg.uwaterloo.ca/models/ilit.php.

[37] Natural Phenomena Simulation Group (NPSG). *Human Iris Data*. School of Computer Science, University of Waterloo, Ontario, Canada, 2016. http://www.npsg.uwaterloo.ca/data/iris.php.

[38] Natural Phenomena Simulation Group (NPSG). *Run ILIT Interactive*. School of Computer Science, University of Waterloo, Ontario, Canada, 2016. http://www.npsg.uwaterloo.ca/models/ilitInteractive.php.

[39] M. Nielsen and M. Stokes. The creation of the sRGB ICC profile. In *Color and Imaging Conference*, volume 1998, pages 253–257. Society for Imaging Science and Technology, 1998.

[40] L. Northam and G.V.G. Baranoski. A novel first principles approach for the estimation of the sieve factor of blood samples. *Optics Express*, 18(7):7456–7468, 2010.

[41] J. Novák, V. Havran, and C. Dachsbacher. Path regeneration for random walks. *GPU Computing Gems Emerald Edition*, page 401, 2011.

[42] NVIDIA. CUDA C Best practices guide. Technical report, NVIDIA Corporation, Santa Clara, CA, USA, 2015.

[43] NVIDIA. Cuda C Programming guide. Technical report, NVIDIA Corporation, Santa Clara, CA, USA, 2015.

[44] NVIDIA. CUDA random number generation library. Technical report, NVIDIA Corporation, Santa Clara, CA, USA, 2015.

[45] N. Ohta and A.R. Robertson. *Colorimetry Fundamentals and Applications.* John Wiley & Sons, New York, NY, USA, 1982.

[46] V.F. Pamplona, M.M. Oliveira, and G.V.G Baranoski. Photorealistic models for pupil light reflex and iridal pattern deformation. *ACM Transactions on Graphics*, 28(4):106:1–106:12, 2009.

[47] R.D. Peng. Reproducible research in computational science. *Science*, 334(6060):1226–1227, 2011.

[48] S. Regan, K.M. Egan, and E.S. Gragoudas. Iris color as a risk factor in uveal melanoma. *Investigative Ophthalmology and Visual Science*, 38(4):3771–3771, 1997.

[49] P. Robertson and J. Schonhut. Color in computer graphics. *IEEE Computer Graphics and Applications*, 19(4):18–19, 1999.

[50] S. Ryoo, C.I. Rodrigues, S.S. Baghsorkhi, S.S. Stone, D.B. Kirk, and W.W. Hwu. Optimization principles and application performance evaluation of a multithreaded GPU using CUDA. In *Proceedings of the 13th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, pages 73–82. ACM, 2008.

[51] M. Sagar, D. Bullivant, G. Mallinson, D. Gordon, and P.J. Hunter. A virtual environment and model of the eye for surgical simulation. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, pages 205–212. ACM, 1994.

[52] A. Schollmeyer, A. Babanin, and B. Froehlich. Order-independent transparency for programmable deferred shading pipelines. *Comp. Graph. Forum*, 34(7):67–76, 2015.

[53] M. Schwarz and M. Stamminger. Fast GPU-based adaptive tessellation with CUDA. *Comp. Graph. Forum*, 28(2):365–374, 2009.

[54] M. Stokes, M.D. Fairchild, and R.S. Berns. Precision requirements for digital color reproduction. *ACM Transactions on Graphics*, 11(4):406–422, 1992.

[55] S.E. Susstrunk, J.M. Holm, and G.D. Finlayson. Chromatic adaptation performance of different RGB sensors. In *Photonics West 2001-Electronic Imaging*, pages 172–183. International Society for Optics and Photonics, 2000.

[56] L. Wang, W. Wang an J. Dorsey, X. Yang, B. Guo, and H. Shum. Real-time rendering of plants. *ACM Transactions on Graphics*, 24(3):712–719, 2003.

[57] A. Watt and M. Watt. *Advanced Animation and Rendering Techniques.* Addison-Wesley, New York, N.Y., USA, 1992.

[58] L. Wecker, F. Samavati, and M. Gavrilova. Iris synthesis: A reverse subdivision technique. In *3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*, pages 121–125, 2005.

[59] E. Wood, T. Baltrusaitis, X. Zhang, Y. Sugano, P. Robinson, and A. Bulling. Rendering of eyes for eye-shape registration and gaze estimation. *arXiv preprint:1505.05916*, 2015.

[60] Z. Yuan, Y. Zhang, J. Zhao, Y. Ding, C. Long, L. Xiong, D. Zhang, and G. Liang. Real-time simulation for 3d tissue deformation with CUDA based GPU computing. *Journal of Convergence Information Technology*, 5(4):109–119, 2010.

[61] J. Zuo and N.A. Schmid. A model based, anatomy based method for synthesizing iris images. In *Advances in Biometrics*, pages 428–435. Springer, 2006.

APPENDICES

Appendix A

Conversion from Spectral Reflectance Data to sRGB Values

As indicated in Chapter 3, the modeled spectral reflectance data was converted to sRGB in order to generate the iris swatches employed in our investigation. This appendix outlines the main components of this conversion process, including the formulation for the conversion from spectrum to XYZ, the conversion from XYZ to sRGB as well as the corresponding implementation in Matlab.

A.1 Spectrum to XYZ Conversion

The XYZ tristimulus coordinates can be obtained from a given spectral reflectance curve using the following equations [7]:

$$
\begin{aligned}
X &= K \sum_{\lambda=400}^{700} S(\lambda)x(\lambda)R(\lambda), \\
Y &= K \sum_{\lambda=400}^{700} S(\lambda)y(\lambda)R(\lambda), \\
Z &= K \sum_{\lambda=400}^{700} S(\lambda)z(\lambda)R(\lambda)
\end{aligned}
\tag{A.1}
$$

where $x(\lambda)$, $y(\lambda)$, $z(\lambda)$ are CIE 1931 color-matching functions (two-degree observer), $S(\lambda)$ is the spectral power distribution of the illuminant, $R(\lambda)$ is the spectral reflectance and K is defined as

$$
K = \frac{100}{\sum_{\lambda=400}^{700} S(\lambda)y(\lambda)}. \tag{A.2}
$$

A.2 XYZ to sRGB Conversion

The sRGB triple can be obtained from XYZ tristimulus values through a number of steps described in the literature [12, 49] and concisely outlined in this section for completeness. Initially, intermediate linear sRGB values R_l, G_l and B_l are computed using the following equation:

$$
\begin{bmatrix} R_l \\ G_l \\ B_l \end{bmatrix} = \begin{bmatrix} 3.2404542 & -1.5371385 & -0.4985314 \\ -0.9692660 & 1.8760108 & 0.0415560 \\ 0.0556434 & -0.2040259 & 1.0572252 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \tag{A.3}
$$

where X, Y and Z are normalized tristimulus coordinates. The resulting intermediate values are then used to compute the final sRGB triple:

$$R = \begin{cases} 12.92R_l & R_l <= 0.0031308 \\ 1.055R_l^{\frac{1}{2.4}} - 0.055 & R_l > 0.0031308 \end{cases},$$
$$G = \begin{cases} 12.92G_l & G_l <= 0.0031308 \\ 1.055G_l^{\frac{1}{2.4}} - 0.055 & G_l > 0.0031308 \end{cases}, \tag{A.4}$$
$$B = \begin{cases} 12.92B_l & B_l <= 0.0031308 \\ 1.055B_l^{\frac{1}{2.4}} - 0.055 & B_l > 0.0031308 \end{cases}.$$

This method assumes the use of the CIE standard D65 illuminant [26].

In cases where the XYZ tristimulus values were captured under a different illuminant, chromatic adaptation to D65 is necessary [39]. This is done using the following equations [6]:

$$\begin{bmatrix} X'' \\ Y'' \\ Z'' \end{bmatrix} = \begin{bmatrix} M_{CAT} \end{bmatrix}^{-1} \begin{bmatrix} \frac{R''_w}{R'_w} & 0 & 0 \\ 0 & \frac{G''_w}{G'_w} & 0 \\ 0 & 0 & \frac{B''_w}{B'_w} \end{bmatrix} \begin{bmatrix} M_{CAT} \end{bmatrix} \begin{bmatrix} X' \\ Y' \\ Z' \end{bmatrix},$$
$$\begin{bmatrix} R'_w \\ G'_w \\ B'_w \end{bmatrix} = \begin{bmatrix} X'_w \\ Y'_w \\ Z'_w \end{bmatrix} \begin{bmatrix} M_{CAT} \end{bmatrix}, \begin{bmatrix} R''_w \\ G''_w \\ B''_w \end{bmatrix} = \begin{bmatrix} X''_w \\ Y''_w \\ Z''_w \end{bmatrix} \begin{bmatrix} M_{CAT} \end{bmatrix}$$

where X', Y', Z' are tristimulus values under the captured illuminant, X'', Y'', Z'' are tristimulus values under the target illuminant, X'_w, Y'_w, Z'_w are tristimulus coordinates of the white point under the captured illuminant and X''_w, Y''_w, Z''_w are tristrimulus coordinates of the white point under the target illuminant. The matrix M_{CAT} corresponds to a chromatic adaptation transform [6]. A widely used choice for M_{CAT} is the Bradford transform [55]:

$$M_{CAT} = \begin{bmatrix} 0.8951 & 0.2664 & -0.1614 \\ -0.7502 & 1.7135 & 0.0367 \\ 0.0389 & -0.0685 & 1.0296 \end{bmatrix}. \tag{A.5}$$

We note, however, that other chromatic adaptation transform matrices exist in literature [6].

The color conversion from spectral reflectance data to sRGB was implemented using functions provided by the Matlab Image Processing Toolbox [1] shown in Figure A.1.

```matlab
function rgb = getRGB(waves, refls)

%Create color transformation structure

% Transformation from xyz (1931 CIE XYZ tristimulus values
% (2 degree observer)) to srgb (Standard computer monitor RGB
% values (IEC 61966-2-1)), using adapted white point according
% to illuminant choice

%ill is 'd65', 'd50' or 'a'
[wI, iI] = illuminant(ill);
XYZ2sRGB = makecform('xyz2srgb','AdaptedWhitePoint',whitepoint(ill));

iI = interp1(wI,iI,waves);

% Multiply reflectance by illuminant spectral power distribution
refls = refls .* iI;

% Get xyz color matching functions for wavelengths
[¬, xyz] = spectrumRGB(waves);

% Compute integrals (Trapezoidal numerical integration) to get ...
   XYZ triple
X = trapz(waves,xyz(1,:,1).*refls);
Y = trapz(waves,xyz(1,:,2).*refls);
Z = trapz(waves,xyz(1,:,3).*refls);
XYZ(1,:,1) = X;
XYZ(1,:,2) = Y;
XYZ(1,:,3) = Z;

% Modulate XYZ by intensity value
XYZ = XYZ/(trapz(waves,iI)*intensity);

% Apply color transformation to get resulting sRGB triple
rgb = applycform(XYZ, XYZ2sRGB);
rgb = reshape(rgb,[1 3]);

end
```

Figure A.1: The conversion pipeline from spectral reflectance data to sRGB as implemented in Matlab. Spectral reflectance values are converted to XYZ tristimulus coordinates using trapezoidal numerical integration, and a color transformation from XYZ to sRGB is applied using functions provided by the Image Processing Toolbox [1].

Appendix B

Steps in the Computation of CIE1976 Color Differences

This appendix outlines the conversion from XYZ values to CIE 1976 $L^*a^*b^*$ coordinates that is required to compute ΔE^*_{ab} color differences [54] that were introduced in Chapter 3.

The $L^*a^*b^*$ coordinates can be obtained from the CIE 1931 XYZ values using the following equations [8]:

$$\begin{aligned}
L^* &= \begin{cases} 116\left(\frac{Y}{Y_n}\right)^{\frac{1}{3}} - 16, & \frac{Y}{Y_n} > 0.008856 \\ 903.3\left(\frac{Y}{Y_n}\right) & \frac{Y}{Y_n} <= 0.008856 \end{cases}, \\
a^* &= 500\left[f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right], \\
b^* &= 200\left[f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right]
\end{aligned} \tag{B.1}$$

where X_n, Y_n and Z_n are tristimulus coordinates of a selected white point (standard illuminant) and $f(s)$ is defined as

$$f(s) = \begin{cases} s^{\frac{1}{3}} & s > 0.008856 \\ 7.787(s) + \frac{16}{116} & s <= 0.008856 \end{cases} . \tag{B.2}$$

We regard that the standard illuminant considered in the computation was D65 [26].

The XYZ to $L^*a^*b^*$ conversion as well as ΔE^*_{ab} color difference calculation was implemented using functions provided by the Image Processing Toolbox [1] in Matlab as shown in Figure B.1.

```
%Convert colors from XYZ to LAB
c1 = xyz2lab([X1 Y1 Z1],'WhitePoint','d65')
c2 = xyz2lab([X2 Y2 Z2],'WhitePoint','d65')

%Compute the CIE76 Delta E Formula
val = sqrt((c1(1)-c2(1))^2 + (c1(2)-c2(2))^2 + (c1(3)-c2(3))^2)
```

Figure B.1: Conversion from tristimulus XYZ coordinates to $L^*a^*b^*$ and computation of the CIE1976 ΔE^*_{ab} formula as implemented in Matlab. Functions provided by the Image Processing Toolbox [1] are used to perform the color conversion.

Appendix C

Effects of Different Block Sizes on Runtime

As described in Section 2.4, in order to fully optimize the CUDA implementation of ILIT for the GTX980 GPU, we needed to determine the most optimal launch parameters. To find these launch parameters, we performed experiments to determine the effects of block size on the overall runtime. As can be verified in Figure C.1, the overall trend for the computationally-heavy runs (10^5 and 10^6 rays) indicates that a block size of 32 provides the best runtime. In addition, one can also note that, for the less computationally-heavy runs (10^3 and 10^4 rays), a different block size may be more effective. However, since we are more concerned with the former scenario, we opted to use a block size of 32.

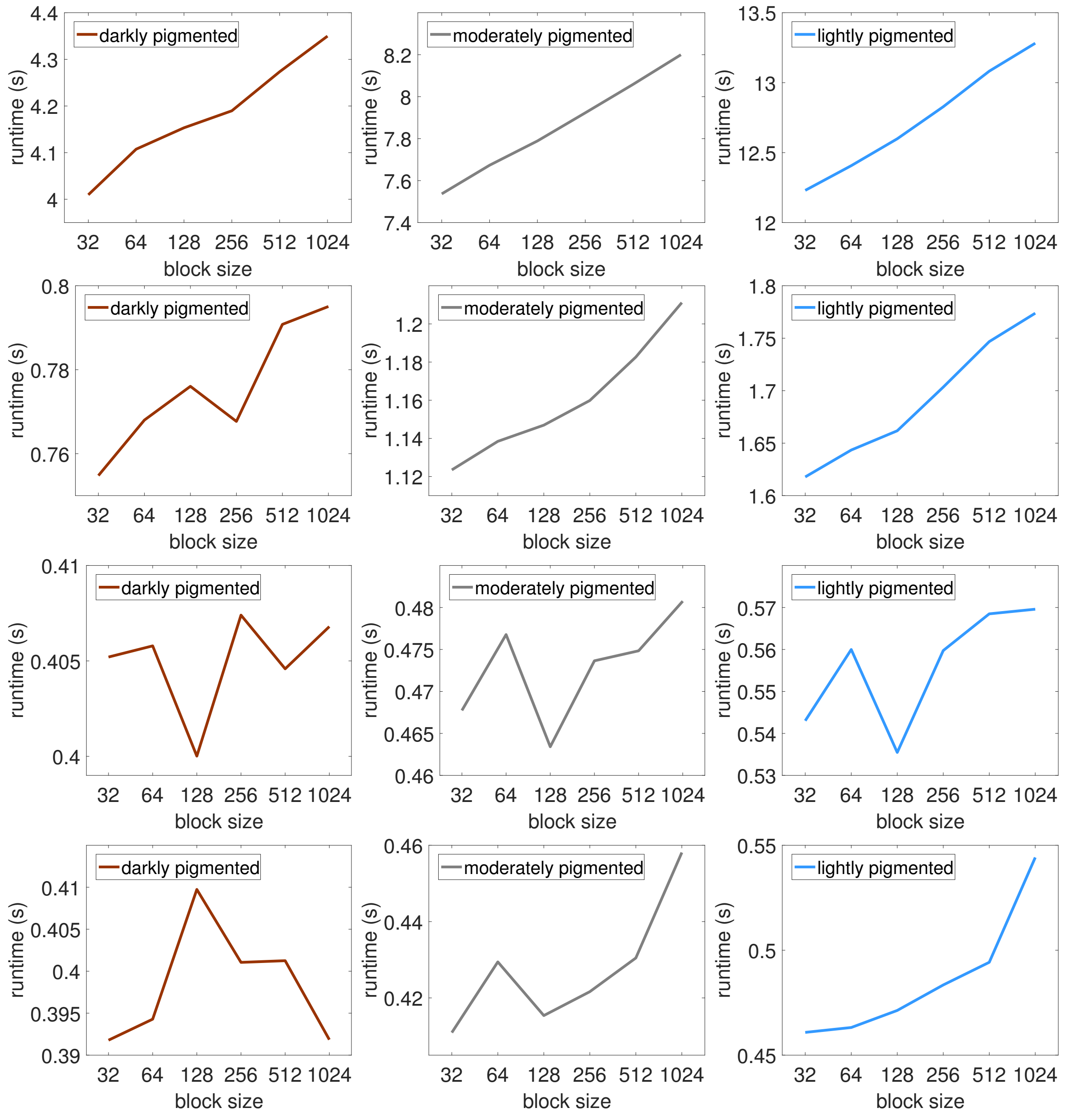


Figure C.1: Effects of block size selection on runtime computed for the selected iris specimens using the GPU-based implementation of ILIT. From left to right: darkly pigmented, moderately pigmented and lightly pigmented specimen, respectively. The curves were computed using distinct numbers of rays. From top to bottom: 10^6, 10^5, 10^4 and 10^3 rays, respectively. All runtimes are averages of 10 distinct runs executed considering a spectral resolution of 1 nm.

Appendix D

Effects of Different Per-Thread Register Allocation Configurations on Runtime

As described in Section 2.4, in our efforts to minimize runtime of the CUDA implementation of ILIT, we performed experiments to determine the effects of different per-thread register allocation configurations. The results are summarized in Table D.1. For the computationally-heavy runs (10^5 and 10^6 rays), it can be seen that for 3 out of 6 tests, a register allocation size of 64 provided the best runtimes. On the other hand, a similar trend was not observed for the less computationally-heavy runs (10^3 and 10^4 rays). Again, since the computrationally-heavy runs represent the most time consuming cases, we elected to allocate 64 registers per thread.

Register Allocation	Darkly Pigmented	Moderately Pigmented	Lightly Pigmented
32	4.03 *s*	7.55 *s*	12.24 *s*
64	4.04 *s*	7.54 *s*	**12.18** *s*
128	4.01 *s*	7.54 *s*	12.23 *s*
compiler	**4.00** *s*	**7.53** *s*	12.23 *s*

Register Allocation	Darkly Pigmented	Moderately Pigmented	Lightly Pigmented
32	0.763 *s*	1.147 *s*	**1.616** *s*
64	**0.754** *s*	**1.121** *s*	1.622 *s*
128	0.766 *s*	1.138 *s*	1.620 *s*
compiler	0.755 *s*	1.123 *s*	1.618 *s*

Register Allocation	Darkly Pigmented	Moderately Pigmented	Lightly Pigmented
32	0.410 *s*	0.475 *s*	0.558 *s*
64	0.405 *s*	0.470 *s*	0.548 *s*
128	**0.404** *s*	**0.460** *s*	0.567 *s*
compiler	0.405 *s*	0.468 *s*	**0.543** *s*

Register Allocation	Darkly Pigmented	Moderately Pigmented	Lightly Pigmented
32	0.392 *s*	**0.392** *s*	0.464 *s*
64	0.394 *s*	0.413 *s*	0.461 *s*
128	**0.384** *s*	0.414 *s*	0.481 *s*
compiler	0.392 *s*	0.411 *s*	**0.460** *s*

Table D.1: Effects of register allocation limit on runtime computed for the selected iris specimens using the GPU-based implementation of ILIT. From top to bottom: 10^6, 10^5, 10^4 and 10^3 rays, respectively. All runtimes are averages of 10 distinct runs executed considering a spectral resolution of 1 *nm*. and block size set to 32. The term *compiler* indicates runs where the register allocation limit was not set, and the appropriate parameters were automatically determined by the compiler. Bold values correspond to minimum runtimes obtained for a given column.

Index