NTNU – Trondheim
Norwegian University of
Science and Technology

Estimation of Skin Optical Parameters for Real-Time Hyperspectral Imaging Applications using GPGPU Parallel Computing

Asgeir Bjørgan

Master of Science in Physics and Mathematics
Submission date: June 2013
Supervisor: Jon Andreas Støvneng, IFY
Co-supervisor: Lise Lyngsnes Randeberg, IET

Norwegian University of Science and Technology
Department of Physics

Project description

The objective of the project is to develop an inverse model for the extraction of optical parameters in skin from reflectance spectra. This inverse model should be applicable on hyperspectral images and deliver results within a deadline limit specified by the speed of the hyperspectral camera, for future real-time processing. The project should:

- Develop an inverse approach using a two-layered skin model and test it using simulated and real reflectance data
- Through the use of CUDA and GPUs, achieve real-time performance for the inverse model and apply it on hyperspectral data
- Integrate the inverse model in a hyperspectral processing framework for modularity
- Develop preliminary calibration and visualization tools for the hyperspectral data and results

Supervisor at the Department of Physics:
Associate Professor Jon Andreas Støvneng

Supervisor at the Department of Electronics and Telecommunications:
Professor Lise Lyngsnes Randeberg

Assignment given: 2013-01-14

Abstract

Hyperspectral imaging with a high spatial and spectral resolution can be used to analyze materials using spectroscopic methods. This can be applied on skin as a general purpose real-time diagnostic tool.

Light transport models, like the diffusion model, can describe the light propagation in tissue before the light is captured by the hyperspectral camera. The aim of this project is, through the inversion of these models as applied on hyperspectral images of skin, to estimate optical properties resident in the tissue. This will be done within a deadline limit defined by the speed of the hyperspectral camera, 30 ms per line of data.

A two-layered skin model is used to estimate the melanin content in epidermis and the dermal absorption coefficient. The skin properties are fitted to the absorption coefficients using a non-negative least squares algorithm at separate wavelength intervals to yield properties down to some penetration depth. Melanin is first fitted in dermis before moved to epidermis. The approach is implemented in CUDA for parallelization on Graphics Processing Units across hyperspectral pixels and wavelengths. The inversion models are integrated in a hyperspectral processing framework developed by Forsvarets Forskningsinstitutt for modularity. Visualization of the results and calibration tools have been developed using Qt and OpenCV.

The resulting inversion chain was found to meet the deadline, finishing the results after 3.5 ms for 1600 pixels, 160 bands and three wavelength intervals. The inversion approach was found to characterize the relative variations of the optical parameters in hyperspectral images of wounds and normal tissue and the absolute values were found to be within physical levels. The inversion routine was tested using Monte Carlo simulations and found to characterize the relative variations in the parameters.

The developed hyperspectral inversion module can be used to estimate skin parameters in real-time. The parameters extracted can be used to characterize different afflictions in skin, and will be one of many necessary processing blocks in a future real-time diagnostic system using hyperspectral imaging.

Sammendrag

Hyperspektral avbildning med høy romlig og spektral oppløsning kan brukes til å analysere stoffer ved hjelp av spektroskopiske metoder. Dette kan brukes på hud som et generelt sanntidsdiagnostiseringsverktøy.

Modeller for lystransport, slik som diffusjonsmodellen, kan brukes til å beskrive propagasjonen av lys i vevet før lyset fanges av det hyperspektrale kameraet. Målet med dette prosjektet er, gjennom inversjonen av disse modellene anvendt på hyperspektrale bilder av hud, å estimere optiske egenskaper i vevet. Dette vil bli gjort innenfor en tidsfrist definert av hastigheten på det hyperspektrale kameraet, 30 ms per linje med data.

En tolagsmodell blir brukt til å estimere melanininnholdet i epidermis og absorpsjonskoeffisienten i dermis. Hudegenskapene blir tilpasset absorpsjonskoeffisientene ved hjelp av en ikke-negativ minste kvadraters metode på separate bølgelengdeintervall for å gi ut egenskapene ned til en eller annen penetrasjonsdybde. Melanin tilpasses først i dermis før det flyttes oppover til epidermis. Framgangsmåten er implementert i CUDA for parallellisering på grafikkort på tvers av hyperspektrale piksler og bølgelengder. Inversjonsmodellene er integrert i et prosesseringsrammeverk for hyperspektrale bilder utviklet av Forsvarets Forskningsinstitutt. Visualisering av resultatene og kalibreringsverktøy har blitt utviklet ved hjelp av Qt og OpenCV.

Den resulterende inversjonskjeden ligger innenfor tidsfristen, og gir ut ferdige resultat etter 3.5 ms for 1600 piksler, 160 bølgelengder og tre bølgelengdeintervall. Inversjonsmetoden viste seg å være i stand til å karakterisere relative forandringer i de optiske parametrene i hyperspektrale bilder av sår og normal hud, og de individuelle verdiene lå innenfor de fysiske grensene som forventes i slik hud. Inversjonsrutinen ble testet på Monte Carlo-simuleringer, og viste seg å karakterisere de relative forandringene i parametrene.

Den utviklede inversjonsmodulen for hyperspektrale bilder kan brukes til å estimere hudparametere i sanntid. De ekstraherte parametrene kan brukes til å karakterisere forskjellige tilstander i hud, og dette kommer til å være en av mange nødvendige prosesseringsblokker i et framtidig diagnosesystem basert på hyperspektral avbildning.

Preface

This thesis is written as the final part of a Master of Technology in Engineering Physics at the Norwegian University of Science and Technology. The work was conducted at the Department of Electronics and Telecommunication. The motivation behind the project is to develop real-time skin diagnostic tools based on hyperspectral imaging for use in medical imaging. The work presented here is one processing block which will be a part of a larger system. The work is an intersection between physics, modelling, medicine and computer science. It has not been easy. Making dents in the table with my head has often seemed like the only escape, though I won through in the end. Tables still standing.

The work in this thesis is based on previous project work performed by me [10]. Some parts from the project work has been reused in the theory and methods sections, although with some major or minor modifications. These parts are parts of the introduction, the overview over GPU architecture, some parts of the derivation of the diffusion model, scattering and absorption descriptions, Monte Carlo description, description of light transport in tissue, some parts of the general description of inversion strategy. Some figures have been modified and reused, noted in each respective figure text.

I'd like to first and foremost give a heartfelt thanks to my supervisor, Lise Lyngsnes Randeberg, for all help and support and oppourtunities. I also want to thank Norsk Elektro Optikk and Forsvarets Forskningsinstitutt for sharing of code, and Trym from FFI for answering the few questions I had about their code framework. Thanks to Lukasz for dealing me data.

Also a thanks to Terje for proof-reading. Thanks to #middagsfjas @IRCNET for intellectual dinner discussions. Thanks to Kjetil for discussions. Thanks to my parents for shoes and upbringing. You probably did it right.

Asgeir Bjørgan
June 2013, NTNU Trondheim

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Hyperspectral imaging is a technique where images are taken and each pixel represents the whole visible spectrum instead of intensity values for red, green and blue. Hyperspectral imaging is widely used for remote sensing, and various techniques have been developed for information extraction. Hyperspectral imaging has also recently been adopted for diagnostic purposes in imaging of skin [76].

Human skin is a large structure covering the entire human body. Skin is subdivided into three main layers, epidermis, dermis and hypodermis [25]. The epidermis protects the skin against water loss and disease-causing organisms, and is a dynamic structure which will protect itself when affected by external stimuli [25]. For instance, it will produce melanin to protect itself against UV-radiation, providing a tan [25]. Dermis consists mainly of connective tissue and blood vessels supplying nourishment to the skin [25]. Hypodermis contains fat cells [25]. Light can penetrate through mainly the first two of these layers and be reflected back. The amount of light reflected back at different wavelengths will reveal information about the structures and properties in the different layers.

A hyperspectral image of a skin sample will therefore contain important information about different materials at different depths. This information can be extracted and used for various purposes in medicine. Examples are determining the age of bruises [75], characterizing wounds [20], characterizing port-wine stains [100] and more. As a part of a medical instrument, this can be used by medical doctors to non-invasively and objectively assess patients instead of relying on visual inspection only. The information extraction procedures must be fast for this to be viable.

Hyperspectral images are large and contain much information. While a hyperspectral camera today will be able to deliver its images fast enough, the necessary processing of the data will be a bottleneck. The hyperspectral community has used FPGAs [34] and GPUs [34, 29, 28, 84, 87, 88, 90] to alleviate the computational stress associated with the huge data load.

Traditional hyperspectral processing methods used in the remote viewing community will often rely on spectral unmixing and endmember extraction methods applied directly on radiance data. Objects present in the scene can be recognized by having different signature spectra. In case of skin, the materials present will be buried within the skin layers and obstructed by wavelength-dependent scattering, which complicates the problem. The dermal properties will also be shielded by a high melanin absorption in epidermis.

Light transport models may describe the light transport through layered models approximating human skin. Scattering can be eliminated through inversion of these models. Monte Carlo methods for calculating light transport are regarded to be more accurate solutions to the Boltzmann transport equation than approximations like the diffusion approximation. In particular is MCML (Monte Carlo for Multi-Layered media) [109] widely regarded to be the so-called gold standard in the biomedical optics-field. The MCML package is freely available on the Internet [59], which might be part of the reason for its wide regard. However, Monte Carlo simulations are slow, since only one photon packet at a time can be simulated. Typically, the number of photon packets need to be in the order of magnitude of 30000 for the results to be accurate. A calculation time of 20 minutes will be typical for one forward calculation of

the full visible spectrum given known parameters, from 400 nm to 800 nm with a spectral resolution of 400 wavelengths [10]. A parallelized approach using GPUs [5, 4] will reduce this time to 40-60 seconds [10]. This Monte Carlo approach is not viable for real-time use as a hyperspectral image will typically consist of 160 bands $\times$ 1600 pixels $\times$ an unknown number of hyperspectral data lines. Another method is to use the diffusion theory, which is an approximation. It has however been used successfully by many different researchers [100, 93, 74, 77, 62, 61, 75], also for inverse modelling. The diffusion theory has an analytical solution to the problem, evaluatable using simple arithmetics.

The work of this thesis was initiated by half a year's worth of project work presented in [10]. The performance of GPU-MCML was here compared against the diffusion model. The derivation of the absorption needed in dermis from dummy input hyperspectral reflectance data was found to be feasible within the defined deadline using CUDA and the diffusion model. This depended on the knowledge of the melanin absorption in epidermis. Some traditional methods for quantifying the relative differences in melanin content were investigated and found insufficient due to overestimation in the presence of deoxy hemoglobin crosstalk. Much of the testing in the work was also done using two-layered skin models, although subdividing dermis into two layers has been seen to be more realistic [75] situation to test the inverse models against. More emphasis was put on Monte Carlo in the project work than has been done in this thesis. Before starting this master thesis' work, some major parts were therefore missing:

- A good method for determining the melanin amount, feasible for GPU implementation
- Optimization of the derivation of the dermal absorption
- Integration into a modular hyperspectral framework so that real hyperspectral data could be input into the model, and the results easily visualized or processed further
- Spectral unmixing for determination of optical properties from derived absorption coefficients
- Use of three-layered forward models for testing
- Validation of the two-layered approach
- Instant visualization of the results
- Calibration of the hyperspectral radiance data into reflectance data

The total inverse modelling is desired to be fast with little to no waiting time. There is an external limit given by how fast the hyperspectral camera can capture its surroundings, and the ambition is to deliver results equally fast, or faster. This thesis will present a solution to the problem delivering results for visualization well within the above defined deadline. The used approach will be evaluated both using real data and simulations. There have been earlier approaches to the inversion of hyperspectral images [104, 44, 30], but none with real-time performance while using more advanced light transport models.

The theory chapter will introduce the light transport problem in tissue and show its solution in terms of Monte Carlo modelling and the diffusion model. Absorption and scattering properties in skin as assumed in the literature will be presented. An overview of the GPU architecture is given, along with some real-time theory and image processing. The spectral unmixing problem and its solutions is presented.

The methods chapter will go through the inversion strategy and detail the CUDA and framework implementations.

The results and discussion chapter will first give some general results for the choice of melanin inversion model and discuss alternative methods. The effects of over- and underestimation of melanin will be given. The three-layered model fit will be investigated in some problematic wavelength ranges to give some explanations for later, possible two-layered modelling failure. The timing results for the implementation will be presented and discussed, first some individual CUDA kernels and then the entire inversion chain. Numerical accuracy and convergence is next investigated, after which the inverse model is verified against real hyperspectral data and Monte Carlo simulations. The visualization and calibration parts of the chain are discussed in the end.

Chapter 2

Theory and background

Light transport theory will first be presented, after which the GPU architecture is presented. Some smaller parts of the theory like least squares fitting, image processing and real-time theory are presented in the end, before spectral unmixing is discussed.

2.1 Light transport in tissue

2.1.1 Absorption and scattering mechanisms of photons in human tissue

Photons will travel down to some penetration depth when light is shined on tissue, readily illustrated by shining a flashlight through a hand. Photons will scatter upon encounter with materials with different refraction indices than the rest of tissue, for example collagen fibres or red blood cells.

Single-scattering theory is assumed to be valid. The scattering from a single particle can be described by its scattering cross section, σ_s, meant to denote the amount of power scattered in all directions [42]. The scattering events can be considered independent and single-scattering theory can be assumed if the mean distance between scattering particles is greater than both scatterer size and the wavelength [110]. In case of a homogeneous medium, the scattering can be described by its scattering coefficient, $\mu_s = \sigma_s \cdot N_s$, where N_s is the number density of scatterers. Cells are the largest structures in tissue, 10 microns in size [110], the packing of which will greatly depend on the cellular soup in question. For the sake of simplicity, biological materials are still in general assumed to be sparsely distributed materials and single-scattering theory is assumed. Rayleigh and Mie theories can give measures of the scattering cross section [110]. Rayleigh theory can be applied if the particles are much smaller than the wavelength, and Mie theory will be valid regardless of the particle size, but is more difficult to calculate [110].

The energy of a photon may be absorbed by a particle and re-emitted as a new photon, or in part cause the particle to enter a vibrational state. This process can similarly be described by an absorption cross section σ_a and an absorption coefficient μ_a, but with no reservations about sparse distributions [110]. If the medium in question is a non-scattering medium, the transmittance through the medium can be expressed as

$$T(x) = e^{-\mu_a x}, \tag{2.1}$$

but this will generally not be valid if the medium is a scattering medium. Absorbing materials will be referred to as chromophores, or in the case of the more hyperspectral application, endmembers, throughout this thesis.

Light will lose coherence and polarization can be neglected when the light undergoes many scattering events. Taking the above properties into account and accounting for all scattering and absorption losses, the situation may be summed up in the Boltzmann equation for photon transport, alternatively known

as the radiative transfer equation. Its derivation can be found in [110], only the time-independent result will be presented here:

$$\hat{s} \cdot \nabla L(\vec{r}, \hat{s}) = \mu_s \cdot \int_{4\pi} L(\vec{r}, \hat{s}')p(\hat{s}' \cdot \hat{s})d\Omega' - \mu_t L(\vec{r}, \hat{s}) + S(\vec{r}, \hat{s}) \tag{2.2}$$

The radiance L is the energy flow per unit normal area per unit solid angle directed in direction $\hat{s}$ at the position $\vec{r}$. The probability of scattering in the direction $\hat{s}'$ when the photon originally had the direction $\hat{s}$ is described by p, the phase function. The extinction coefficient μ_t is equal to $\mu_a + \mu_s$, and the last term, S, is the source term.

The left-hand side is loss due to divergence of the beam. The integration involving L and p is the main scattering term, wherein photons are scattered in some direction as according to the phase function. The extinction coefficient μ_t is loss due to extinction, and the source term represents new photons entering the position $\vec{r}$ in the direction $\hat{s}$.

The Henyey-Greenstein phase function is often used for biological materials to describe the phase function p. This was originally used by Henyey and Greenstein [40] to describe diffuse radiation in galaxies. It was found by Jacques et al. [43] to be a good fit also for the angular dependence of the scattering in biological tissues, at least at the wavelength 632 nm. This phase function is given as

$$p(\hat{s}' \cdot \hat{s}) = p(\cos\theta) = \frac{1 - g^2}{2(1 + g^2 - 2g\cos\theta)^{3/2}}. \tag{2.3}$$

θ is the angle between the original photon direction and the scattered photon direction, while g is the anisotropy factor, defined as

$$g = \int \cos\theta p(\cos\theta) d(\cos\theta), \tag{2.4}$$

i.e. the average of the cosine of the scattering angle.

The two most popular solutions to (2.2) are Monte Carlo and diffusion theory. Monte Carlo will rely on the phase function, while whether or not it will be used in the diffusion model depends on the chosen source function.

2.1.2 Monte Carlo

The derivation of the Monte Carlo program is best described by referring to other sources like [109, 110], but it will be summarized.

In essence, a Monte Carlo-implementation of (2.2) will involve tracking each photon packet and accounting for absorption and scattering losses as according to the mechanisms described by equation (2.2), using probabilistic methods. A figure displaying the program flow for the Monte Carlo package MCML is shown in figure 2.1. The step size is represented by s. This is sampled from a random distribution which describes the likelihood of encountering an absorption or scattering event, hence the logarithm of an uniformly distributed number ξ between 0 and 1. Fresnel's equations will be used to describe the reflection and transmission if the particles hit the boundary between two layers. Otherwise are they moved according to the step size. The weight of the photon packet is reduced as according to absorption and back-scattering determined by μ_t, and the new scattering direction is determined by sampling a random variable from (2.3). Following photons with small weights yields little interest and such photons are terminated early. The most of the photons are eliminated from the simulation once the photon weight becomes small, and a Russian roulette keeps a few of them alive to ensure energy conservation.

If the photons come back up through the first layer, they are registered in a reflection array. If they are absorbed, they are registered in an absorption array. If they end up at the other side of the simulated slab, they are registered in a transmission array. At the end of the day, these arrays are summed up to yield diffuse reflectance, absorbance and transmittance of the simulated light.

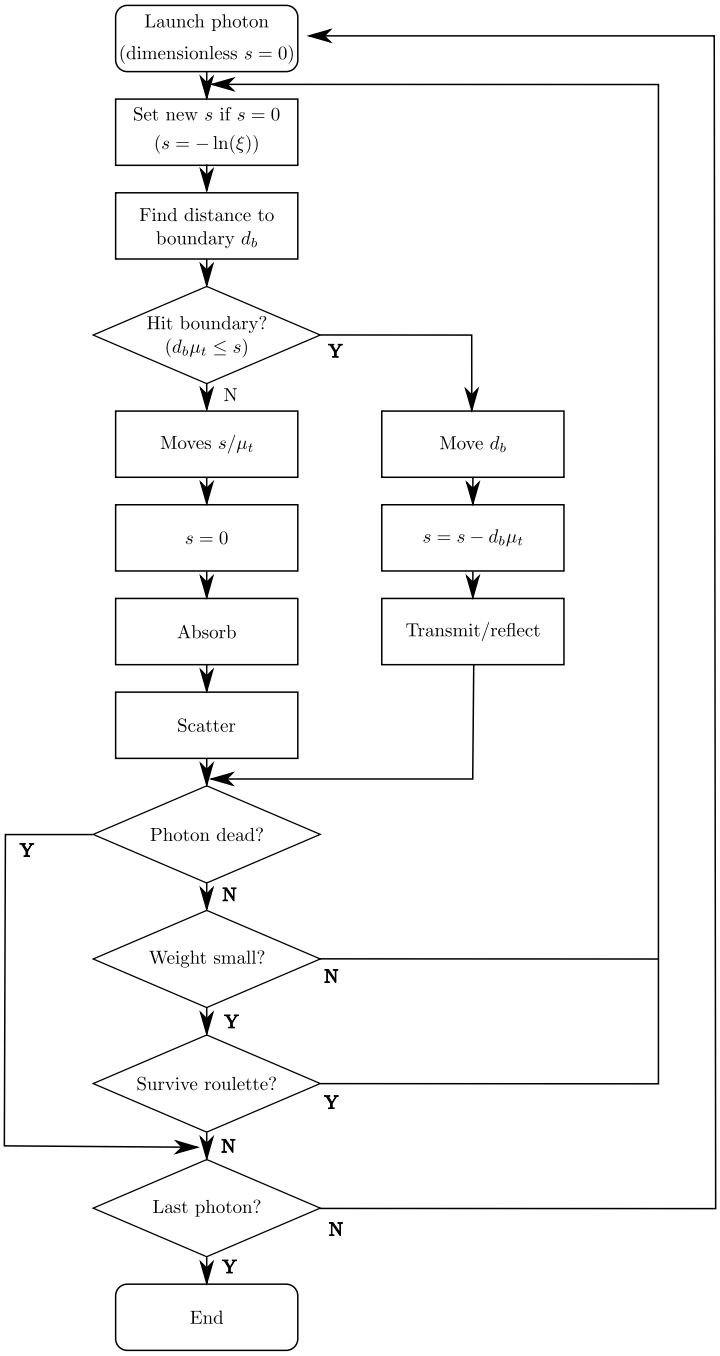


Figure 2.1: The program flow in MCML. From [10], where it was reproduced from Wang et al. [109].

2.1.3 Diffusion model

The derivation of the diffusion model will be shown here, for both the Delta-Eddington source function and the isotropic source function.

Isotropic source function

Equation (2.2) can be integrated over all solid angles in order to yield a continuity equation which may be used later on. The integrated L in the phase function integral will be independent of Ω since it will become the isotropic ϕ after integration, and the phase function will be integrated away to 1 as ϕ can be taken outside of the integral [38]:

$$\nabla\vec{j}(\vec{r}) = -\mu_a\phi(\vec{r}) + q(\vec{r}) \tag{2.5}$$

The integrated L and $L\hat{s}$ have respectively been replaced by the fluence rate ϕ and the diffuse photon flux vector $\vec{j}$, while the integrated source function S has been replaced by q [100].

Equation (2.2) has no analytical solutions, but approximations can be made. The radiance L can be assumed to be almost isotropic if μ_s is much larger than μ_a, and the radiance can then be written as the first two terms of a Legendre polynomial expansion [100]:

$$L = \frac{\phi}{4\pi} + \frac{3}{4\pi}\vec{j}\cdot\hat{s}. \tag{2.6}$$

The first term represents the isotropic part while the second term is the deviation from isotropy in the direction given by the unit directional vector $\hat{s}$ [100]. One main assumption is that the deviation from isotropy is not too large, or else higher order terms would have to be included. The photon flux j should therefore not be too large when compared to ϕ.

If this is inserted into (2.2), the resulting equation multiplied by $\hat{s}$ and integrated over all solid angles will be [38]

$$\vec{j}(\vec{r}) = -D\nabla\phi(\vec{r},t), \tag{2.7}$$

where D is defined as $\frac{1}{3[(1-g)\mu_s+\mu_a]}$. The source term disappears from (2.2) when multiplied by $\hat{s}$ and integrated over all solid angles if it is assumed to be isotropic, which is why it no longer is a part of the equation.

Equations (2.7) and (2.5) can be combined to yield [100]

$$\nabla^2\phi - \frac{\phi}{\delta^2} = -\frac{q}{D}, \tag{2.8}$$

where $\delta = \sqrt{\frac{D}{\mu_a}}$, also known as the optical penetration depth. This equation can be applied for each layer in the skin model, each with its own depth-varying source function. The solution will depend on some boundary conditions at the skin-air surface:

When L from earlier on is integrated over half the solid angle, the result is the irradiance E, the power passing through an unit area. Integrating (2.6) results in [100]

$$E = \frac{\phi}{4} \pm \frac{j}{2}. \tag{2.9}$$

The plus or minus sign depends on whether the photon flux $\vec{j}$ is pointing along or opposite of the surface normal for the surface over which the irradiance is passing through. This can be used for constructing boundary conditions between differently scattering layers, since the irradiance must be the same on both sides of the layer boundary [100]:

$$\frac{\phi_1}{4} + \frac{j_1}{2} = \frac{\phi_2}{4} + \frac{j_2}{2} \tag{2.10}$$

$$\frac{\phi_1}{4} - \frac{j_1}{2} = \frac{\phi_2}{4} - \frac{j_2}{2} \tag{2.11}$$

The flux and fluence rate in layer one are respectively denoted by j_1 and ϕ_1, and j_2 and ϕ_2 are the flux and fluence rate in layer 2. The irradiance should be continuous regardless of which way the surface normal is pointing, which is why there are two boundary conditions, one for the case where the surface normal has the same direction as $\vec{j}$, and one for the case where the surface normal points in the opposite direction. These boundary conditions can also be simplified into the simple criterion that each quantity must be continuous across the surface.

The start boundary condition coupling everything together is the boundary condition at the air-skin interface. By integrating the Fresnel reflection coefficient over all angles of incidence, an R_ϕ and R_j can be found [38]. The reflected part of the irradiance at the air-skin interface must be related to the irradiation propagating back into the skin by [100]

$$R_\phi \frac{\phi}{4} - R_j \frac{j}{2} = \frac{\phi}{4} + \frac{j}{2} \tag{2.12}$$

This can be summed up in a boundary condition $j = -A\phi$, where A depends on R_ϕ and R_j. By using the formulas for R present in [38], A can be calculated to be 0.14 for a refraction index of $n = 1.4$ for the tissue. The criterion that the flux should be much smaller than the fluence rate is therefore fulfilled only to a limited degree at the skin surface [100], but it can be argued that it still will work nicely out [100].

All boundary conditions are summed up as

$$\begin{aligned} j_1 &= j_2 && (2.13)\\ \phi_1 &= \phi_2 && (2.14)\\ j(x=0) &= -A \cdot \phi(x=0). && (2.15) \end{aligned}$$

One important concern is the source function, describing the photons arriving at a given point. If isotropic source functions throughout the layers is assumed, it may be expressed, for a three-layered model, as [100]

$$\begin{aligned} q_1 &= P_0 \mu'_{s,1} e^{-\mu_{tr,1} x} && (2.16)\\ q_2 &= P_0 \mu'_{s,2} e^{-\mu_{tr,1} d_1} e^{-\mu_{tr,2}(x-d_1)} && (2.17)\\ q_3 &= P_0 \mu'_{s,3} e^{-\mu_{tr,1} d_1} e^{-\mu_{tr,2}(d2-d_1)} e^{-\mu_{tr,3}(x-d_2)} && (2.18) \end{aligned}$$

Each source function is applied to a different layer, within the boundaries $x \leq d1$, $d1 \leq x \leq d2$ and $d2 \leq x \leq \infty$. The depth of each layer is denoted by d_i, with the last being semi-infinite. The isotropic source function radiates equal power in all directions. The combination $\mu_s(1-g)$ is often abbreviated as μ'_s, the reduced scattering coefficient. After enough scattering events and little to no absorption, this will be the effective scattering coefficient and can in such cases be used instead of the actual scattering coefficient and complex scattering mechanisms. The combination $\mu_s(1-g) + \mu_a$ is abbreviated as μ_{tr}, the transport coefficient. Indices 1, 2 and 3 refer to the upper skin layer and the two lower skin layers, respectively. The parameter P_0 is the incident intensity after specular reflection at the air-skin surface.

The diffuse reflection coefficient will be expressed by

$$\gamma = \frac{-j(x=0)}{P_0}. \tag{2.19}$$

This is the quantity which will be used throughout this thesis.

The solutions to (2.8) will be of the form

$$\phi = f(x) + Ae^{x/\delta} + Be^{-x/\delta}, \tag{2.20}$$

where $f(x)$ depends on the source function and will follow the same shape, albeit with a different constant.

By inserting the solution and source functions into (2.8) for each layer, the solutions will be found [100]

$$\phi_1 = \frac{P_0\delta_1^2\mu'_{s,1}}{\zeta_1(1-\mu_{tr,1}^2\delta_1^2)}e^{-\mu_{tr,1}x} + A_1e^{-\frac{x}{\delta_1}} + A_2e^{\frac{x}{\delta_1}} \tag{2.21}$$

$$\phi_2 = \frac{P_0\delta_2^2\mu'_{s,2}}{\zeta_2(1-\mu_{tr,2}^2\delta_2^2)}e^{-\mu_{tr,1}d_1}e^{-\mu_{tr,2}(x-d_1)} + A_3e^{-\frac{x}{\delta_2}} + A_4e^{\frac{x}{\delta_2}} \tag{2.22}$$

$$\phi_3 = \frac{P_0\delta_3^2\mu'_{s,3}}{\zeta_3(1-\mu_{tr,3}^2\delta_3^2)}e^{-\mu_{tr,1}d_1}e^{-\mu_{tr,2}(d_2-d_1)}e^{-\mu_{tr,3}(x-d_2)} + A_5e^{-\frac{x}{\delta_3}} \tag{2.23}$$

The constants are found by calculating j_i from each ϕ_i and applying the boundary conditions in (2.15). For A_4 and A_5 equal to zero, the model is reduced to the two-layered diffusion model. The diffuse reflectance is then found by applying (2.19). The full solution for the two-layered case can be found in [100]. The full three-layered solution has been implemented as C++-code.

Delta-Eddington

There are also alternatives to using the isotropic source functions. One alternative is the Delta-Eddington source function, presented by Spott and Svaasand [92]. The diffusion approximation is to neglect the higher orders of non-isotropy, which when μ_s is no longer much larger than μ_a will no longer be valid. The Delta-Eddington source function seek to remedy this by keeping some of its non-isotropy. It is derived using the Henyey-Greenstein phase function, shown in [92]. It is also shown to have better correspondence with MCML [92, 77, 10].

The derivation of the solutions is mostly the same as for the isotropic case, although with some differences. The step right before equation 2.8, in the integration of S, was in reality to assume the isotropy of the source function, which is no longer valid. Instead, assuming a one-dimensional situation, this will be expressed as [91]

$$\mu_a\phi + \frac{d}{dx}j = q_0 \tag{2.24}$$

$$\frac{1}{3}\frac{d}{dx}\phi + \mu_{tr}j = q_1. \tag{2.25}$$

The terms q_0 and q_1 are respectively the first and second moments of the source function. The term q_0 is the isotropic term, while q_1 is the next moment in the deviation from isotropy. For the isotropic source function, q_1 is set to 0 and the situation reduces itself to equation (2.8). The first and second moments of the Delta-Eddington source function are given by Spott and Svaasand [92] to be

$$q_0 = P_0\mu_s(1-g^2)e^{-(\mu_s(1-g)+\mu_a)x} \tag{2.26}$$

$$q_1 = P_0\mu_s(1-g^2)\frac{g}{1+g}e^{-(\mu_s(1-g)+\mu_a)x}. \tag{2.27}$$

The combination $\mu_s(1-g^2)+\mu_a$ will be referred to as μ'_{tr}. These can be applied to each layer by exchanging μ_a, μ_s and g for the appropriate coefficients and factors for the particular layer. Eliminating j in (2.25), inserting the solution outlined in (2.20) and the source functions above will yield the solutions

$$\phi_1 = \frac{P_0\delta_1^2\mu_{s,1}(1-g^2)}{(1-\mu'^2_{tr,1}\delta_1^2)}(\frac{1}{\zeta_1}+3\frac{g\mu'_{tr,1}}{1+g})e^{-\mu'_{tr,1}x} + A_1e^{-\frac{x}{\delta_1}} + A_2e^{\frac{x}{\delta_1}} \tag{2.28}$$

$$\phi_2 = \frac{P_0\delta_2^2\mu_{s,2}(1-g^2)}{(1-\mu'^2_{tr,2}\delta_2^2)}(\frac{1}{\zeta_2}+3\frac{g\mu'_{tr,2}}{1+g})e^{-\mu'_{tr,1}d_1}e^{-\mu'_{tr,2}(x-d_1)} + A_3e^{-\frac{x}{\delta_2}} + A_4e^{\frac{x}{\delta_2}} \tag{2.29}$$

$$\phi_3 = \frac{P_0\delta_3^2\mu_{s,3}(1-g^2)}{(1-\mu'^2_{tr,3}\delta_3^2)}(\frac{1}{\zeta_3}+3\frac{g\mu'_{tr,3}}{1+g})e^{-\mu'_{tr,1}d_1}e^{-\mu'_{tr,2}(d_2-d_1)}e^{-\mu'_{tr,3}(x-d_2)} + A_5e^{-\frac{x}{\delta_3}} \tag{2.30}$$

The definition of j will be slightly different with these higher order source terms present, as obtained from equation (2.25):

$$j = -\frac{1}{\zeta}\frac{d}{dx}\phi + \frac{1}{\mu_{tr}}q_1. \tag{2.31}$$

Using the solutions for ϕ_i shown in (2.30) to obtain j, the boundary conditions in (2.15) can be satisfied to yield the unknown constants A_1, A_2, A_3, A_4 and A_5. This is reduced to the two-layered diffusion model when A_4 and A_5 are equal to zero. The solution is implemented as C++-code both for the three- and two-layered cases. Delta-eddington was not implemented for the GPU due to its complicated structure, but it was still used for comparison.

Solutions

The one-layered solution to both the Delta-Eddington and isotropic source function will be shown for demonstration purposes.

The isotropic solution:

$$\gamma = \frac{\mu_s' \cdot A \cdot \delta^2}{(\frac{\delta}{3D} + 1)(D + \delta \cdot A)} \tag{2.32}$$

The Delta-Eddington solution:

$$\gamma = 3 \cdot (\mu_{tr}'\delta - 1)\mu_s A \frac{((-\frac{\mu_{tr}' g^3}{3} + ((D\mu_{tr}' + \frac{1}{3})(g+1)(g-1)\mu_{tr} + \frac{\mu_{tr}'}{3})g + (\frac{g^2}{3} - \frac{1}{3})\mu_{tr})\delta + \frac{1}{3}(g - g^3))\delta}{(1+g)(-1 + \mu_{tr}'^2\delta^2)\mu_{tr}(D + \delta \cdot A)} \tag{2.33}$$

The solutions will become more complicated for each layer added.

2.1.4 Skin model

The two-layered skin model employed is described by Spott et al. [91], Randeberg et al. [77, 53] and Svaasand et al. [100]. The three-layered skin model has been described by Randeberg et al. [75, 77].

Two or three-layered skin models are usually deemed sufficient, even if the skin models used for simulating diffuse reflectance might sometimes have as many as seven layers [4, 60]. Due to errors in the simulation methods and many shortcuts in the description of the scattering and absorption mechanisms, the extra amount of layers will only cause unnecessary complications and not extra realism. The number of parameters will increase sevenfold for each layer added, and detecting misfitting will become more difficult since the added number of parameters will be able to shield any apparent misfitting with overdetermination.

Three layers is needed for good fits all throughout the spectrum due to the inhomogeneity of skin. Only two layers were used in the inverse models presented in this thesis, though three-layered models are used in some forward calculations.

The two layers in question are epidermis and dermis. In case of a three-layered model, dermis is subdivided into two new layers to account for the difference in blood volume and blood oxygenation across dermis. Hypodermis is never taken into account as photons scattered back from such depths is negligible [53].

In reality, both epidermis and dermis will be sub-divided into different layers with different properties. These properties are as an approximation assumed uniformly distributed in each layer. Each layer is flat and uniform, but the epidermis will in reality reach partially into the dermis through the papillae [85]. A small fraction of blood, 0.2%, is therefore included in the epidermis even if epidermis does not contain blood.

Scattering

Scattering will predominantly rise from the presence of collagen fibers in the skin, which will induce both Rayleigh and Mie scattering. The wavelength dependency was found by Saidi et al. [82] to be

$$\mu_{s,t}' = C_{Mie}(1 - 1.745 \cdot 10^{-3}\lambda + 9.843 \cdot 10^{-7}\lambda^2) + C_{Rayleigh}\lambda^{-4}. \tag{2.34}$$

This is the reduced scattering coefficient. C_{Mie} and $C_{Rayleigh}$ are some constants. In order to get the non-reduced scattering coefficient, one will have to divide the Mie-part of the scattering by $1-g$ and the Rayleigh-part by 1.0, since Rayleigh scattering is isotropic and will not be reduced when considering the effective scattering coefficient. The anisotropy factor was for skin found by van Gemert et al. [106] to be

$$g = 0.62 + 29 \cdot 10^{-5}\lambda. \tag{2.35}$$

Age-dependencies of C_{Mie} and $C_{Rayleigh}$ are determined by Saidi et al. [82] for newborns. The reported curves were extrapolated into adulthood by Randeberg and the values respectively found to be 10500 m^{-1} and $1.05{\cdot}10^{14}$ $\mathrm{nm}^4\mathrm{m}^{-1}$. There exists other variants of the scattering function, but these were not used.

There will also be scattering from red blood cells, as presented by Ishimaru [42]. The scattering coefficient may be written as [42, 98, 97]

$$\mu_{s,b} = \sigma_{s,b}\frac{H(1-H)(1.4-H)}{v_e}\left(\frac{685}{\lambda}\right)^{0.37}. \tag{2.36}$$

The scattering cross section $\sigma_{s,b}$ was reported by [42] to be $55.09 \cdot 10^{-12}$. The volume of the red blood cells, v_e, is claimed to be $1.26{\cdot}10^{-16}$ m^2 by Spott et al. [93]. H is the haematocrit, the fraction of red blood cells to the total volume of blood. The haematocrit will be 40-52% for males and 38-48% for females [85]. Blood is reported to have an anisotropy factor of $g = 0.9937$ [48], which is duly multiplied against (2.36) to yield the reduced blood scattering coefficient.

The total scattering coefficient for each layer will be

$$\mu_{s,e} = B_e\mu_{s,b} + (1-B_e)\mu_{s,t} \tag{2.37}$$

$$\mu_{s,d} = B_d\mu_{s,b} + (1-B_d)\mu_{s,t}. \tag{2.38}$$

B_d and B_e are the blood volume fractions (bvf) for dermis and epidermis respectively. B_d will later be referred to as the bvf. The scattering functions are extended to a third layer by subdividing the d layer into $d1$ and $d2$, with similar properties although with different B_d parameters.

Only scattering from collagen and blood were implemented. Some will also report a significant melanin scattering [7], while others will report a negligible melanin scattering [79, 117]. Seemingly a controversial issue, implementing a melanin scattering was not attempted.

Absorption

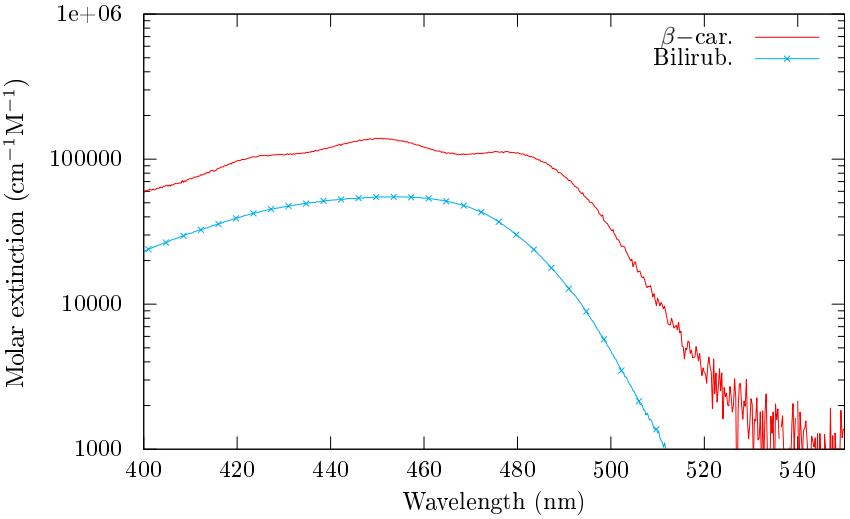
Light will be absorbed in skin mainly due to deoxygenated (deoxy) hemoglobin, oxygenated (oxy) hemoglobin, methemoglobin, melanin, bilirubin, betacarotene and water. The absorption coefficients of these are displayed in figure 2.2. CO-Hb [115], present in smokers, will also absorb light. CO-Hb will be similar to the other blood absorption spectra, but slightly wavelength-shifted.

The ratio of oxy hemoglobin to deoxy hemoglobin is called the oxygen saturation and will be denoted as O or oxy. The absorption of blood will be written as

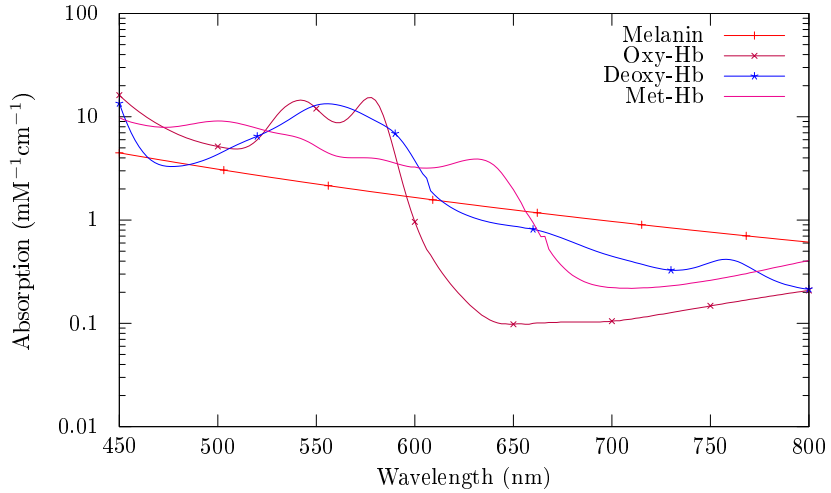
$$\mu_{a,b} = \mu_{a,Hb}\cdot(1-O) + \mu_{a,HbO_2}\cdot O, \tag{2.39}$$

where $\mu_{a,Hb}$ is the absorption coefficient for deoxygenated blood and μ_{a,HbO_2} the absorption coefficient for oxygenated blood. Zijlstra's blood absorption values were plotted in figure 2.2 for more easy comparison with the other chromophores, but Spott's values were used in calculations due to encompassing more wavelengths. Spott's blood absorption values are shown in figure 2.3.

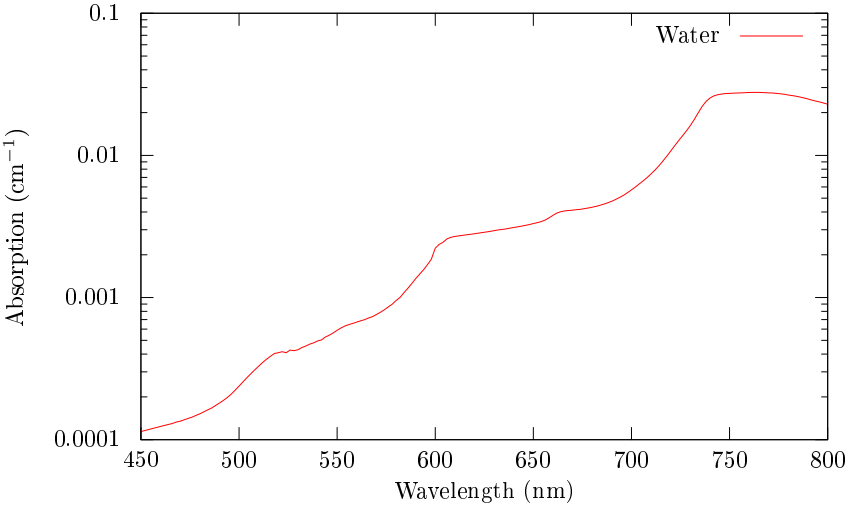
The scaling of the blood absorption maxima is plotted as a function of blood oxygenation in figure 2.4. The scaling can potentially be higher than the maximum at 100% deoxy hemoglobin due to underestimation of melanin or the presence of methemoglobin, although a scaling less than the minimum at 100%



(a) The absorption coefficients of bilirubin in chloroform [22] and β-carotene in hexane [22].



(b) The absorption coefficients of blood [115], methemoglobin [115] and the melanin from (2.40). Note that an arbitrary melanin coefficient has been chosen for comparison.



(c) Absorption of water [13].

Figure 2.2: Wavelength-dependency of the absorption coefficients of different chromophores present in skin. Note differences in scale.

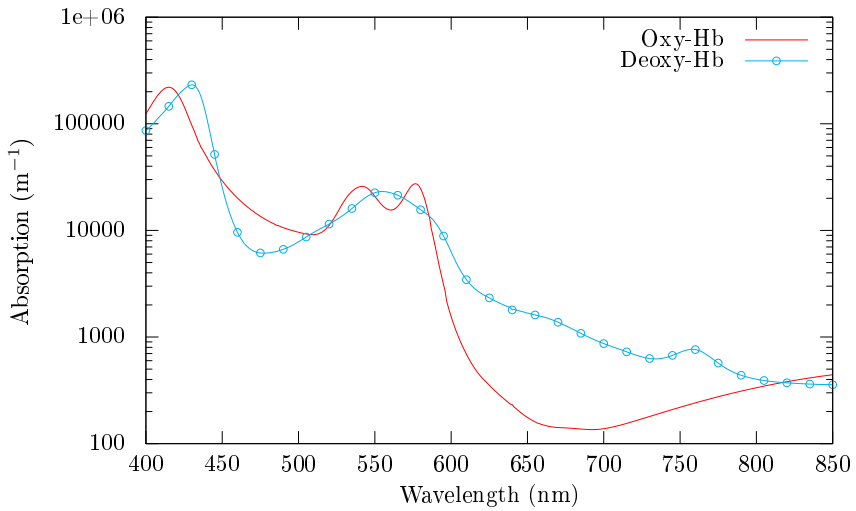


Figure 2.3: Absorption of blood, as measured by Spott [91]. Note the difference in units and scaling as compared to the other chromophore plots.

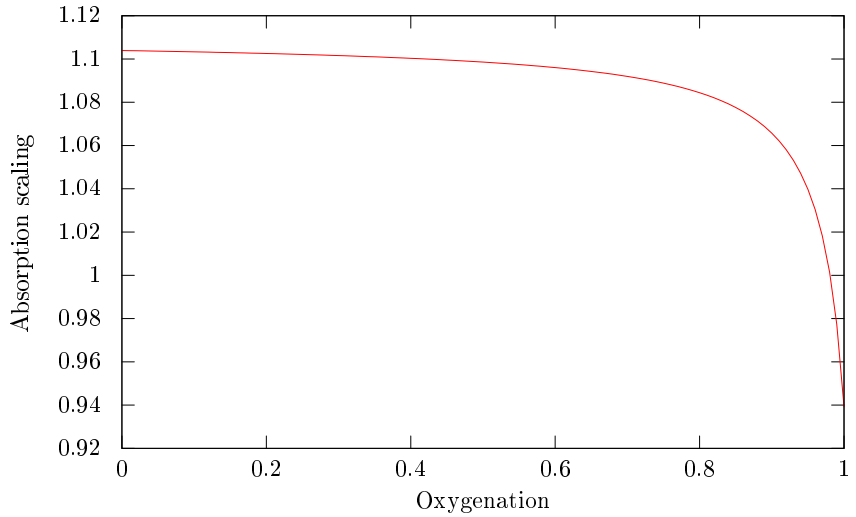


Figure 2.4: The absorption value at 541 nm divided by the absorption value at 576 nm as a function of blood oxygenation.

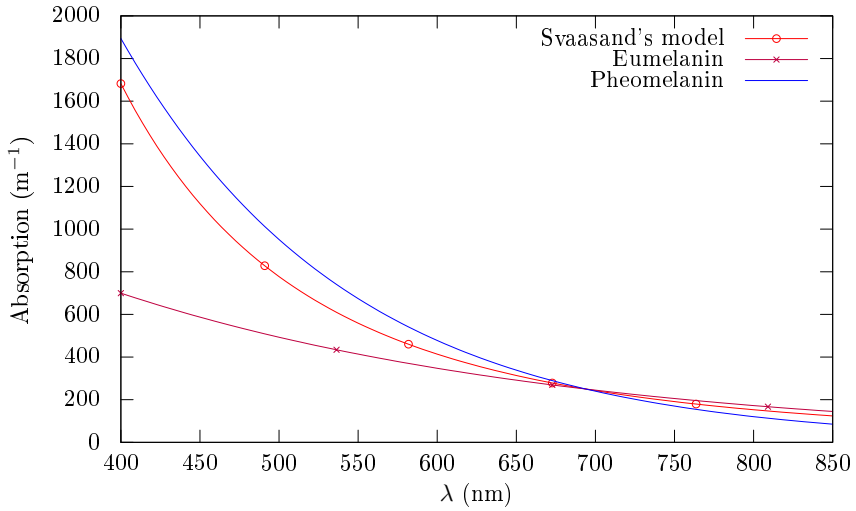


Figure 2.5: Comparison of different melanin types and Svaasand's melanin model for $\mu_{a,m,694} = 225$ m^{-1}.

oxygenated hemoglobin should not be expected other than when the scattering assumption is wrong or the melanin absorption has been overestimated.

According to the model in use, melanin is the main absorber in epidermis. The melanin content will be quantified by the melanin absorption coefficient at 694 nm. Some will report melanin contents by using fractions, but since the melanin will not be evenly distributed, going from fractions to the actual melanin absorption will be difficult. Instead, the melanin absorption at 694 nm is by itself used to indicate the amount of melanin present in the epidermis. Sun-protected European skin will, for example, have a melanin absorption coefficient at 694 nm in the range of 280 to 325 m^{-1} [100]. The wavelength dependence of melanin absorption in skin was investigated by Wolbarsht et al. [111], and again observed by Spott et al. [93] to be

$$\mu_{a,m} = \mu_{a,m,694} \cdot \left(\frac{694}{\lambda}\right)^{3.46} . \tag{2.40}$$

This will be referred to as Svaasand's melanin model. Using 694 nm as a reference point for melanin is a common measure in some parts of the literature [61, 62, 100, 112, 21], making way for easy comparisons. Melanin will also be present in hair [56] and absorb light before the light penetrates the tissue.

There will be two types of melanin present in skin, pheomelanin and eumelanin [56]. Pheomelanin is described to be yellow-red and eumelanin brown-black [56]. The above expression can be seen as a mix of the two. Zonios et al. [117] gave exponential fits for the pheomelanin and eumelanin absorption curves:

$$\mu_{a,m} = \mu_{a,m,694} \cdot e^{-k \cdot \left(\frac{\lambda - 694}{694}\right)} \tag{2.41}$$

This has been modified to use 694 nm as its reference wavelength. Zonios et al. used 400 nm. The coefficient k is given to be 4.780 for pheomelanin and 2.429 for eumelanin, appropriately recalculated for the reference wavelength shift. These are displayed in figure 2.5 along with Svaasand's melanin curve.

The melanin absorption mechanisms are disputed. The melanin is assumed to be highly scattering in [111], the paper used to derive (2.40). The explanation model is that light is scattered back and forth until the melanin has the appearance of being highly absorbing. Bashkatov et al. [7] have claimed to measure the melanin scattering ex vivo, and have precise data for the absorption and scattering parts. Zonios et al. [117] and Riesz [79] will on the other hand claim the scattering to be negligible. Riesz found the melanin to be absorbing only. Zonios et al. claim they have measured the melanin absorption in vivo as opposed to the usual ex vivo measurements, as they are fitting a model to the reflectance spectra and deriving the required melanin curves using an exponential model for the melanin curve. They found the in vivo melanin absorption to differ from the absorption curves measured ex vivo, and they found no contributions from scattering. However, they also used a one-layered model and it will become apparent

in this thesis that this introduces errors both for the melanin level itself and the wavelength-dependency of the melanin curve.

A constant background absorption of $\mu_{a,o} = 25\ \mathrm{m}^{-1}$ will be assumed, as Spott et al. [93] did. Some will incorporate a wavelength-dependency for this baseline absorption, but this was not implemented for this thesis.

The absorption properties of each layer can be summed up as

$$\mu_{a,e} = \mu_{a,b} \cdot B_e + \mu_{a,m} + \mu_{a,o} \cdot (1 - B_e) \tag{2.42}$$

$$\mu_{a,d} = \mu_{a,b} \cdot B_d + \mu_{a,o} \cdot (1 - B_d) \tag{2.43}$$

Other chromophores will be located in dermis, linearly mixed. The above properties are similarly extended to a third layer. Some chromophores, like beta-carotene, will be more pronounced in epidermis [9], but also be present in the other layers [3] and especially in the lower ones since it is fat-soluble [3]. It will still only be included in dermis for simplicity.

2.2 GPU-architecture

The problem of rendering and displaying graphics is in its nature a problem which, if to be done efficiently, requires parallelization and hyperthreading [101]. Due to this fact, Graphics Processing Units (GPUs) are designed to handle multiple calculations at the same time, not only by employing multiple processor cores but also by enabling the different cores to process multiple threads at the same time. GPUs are therefore commonly put to use to other, parallelizable problems than graphics rendering [4, 35, 84, 90]. The GPUs are used as General-purpose graphics processing units (GPGPU). NVIDIA has a C framework called CUDA [2] for easier programming for their GPU devices, but will also have a slightly less advertised support for OpenCL [63], a similar but more flexible framework for parallelization on different devices [64]. AMD's graphics cards support OpenCL [6]. The GPU architecture as exposed by the CUDA framework will be presented, and some major optimization possibilities will be pointed out.

The GPU follows a SIMD (Single instruction, multiple data) principle. The GPU is targeted towards employing the same processor instruction on different datasets [2]. Each processing core on the GPU can handle 32 threads at once. This handling is referred to as a *warp* . A warp can happen only provided each thread performs the same instruction in the same sequence on different data. There are therefore some limits to how well a GPU may parallelize a given problem. All threads need also be independent, so that the GPU may switch between threads at any given time to make up for thread lag in memory access. [2]

CUDA will organize its threads in blocks, as shown in figure 2.6. Each block is assigned to an available GPU multiprocessor, as is shown in figure 2.7.

Each multiprocessor will run the threads of its assigned block until completion, in warps of 32 threads each [2]. A necessary requirement for maximum multiprocessor utilization will hence be to have a number of threads per block divisible by 32. The blocks will be organized in a larger grid. Both the blocks and the grid can be one- or two-dimensional, but this has no performance benefits and is for convenience when accessing any array data [2].

Each thread will run its own instance of a *kernel* [2]. This is a C-like function with some CUDA extensions, compiled for the GPU processor. The kernel is instantiated for a grid of blocks, and each thread in each block of threads will process the instructions in the kernel, on different data accessed by using thread- and block indices. Should any of the threads try to process different instructions, the warp will break down and the multiprocessor will run the threads sequentially instead of in parallel in groups of 32 [2].

GPUs have different kinds of memory [2]. The GPU has some DRAM, called *global memory*, and some multiprocessor-associated caches, a registry and shared memory. Compared to the last three, the DRAM is slow and transfers to and from this will be the bottleneck [1]. Each multiprocessor has a scarce registry into which local variables within the scope of each thread will be allocated. Any intermediate results in

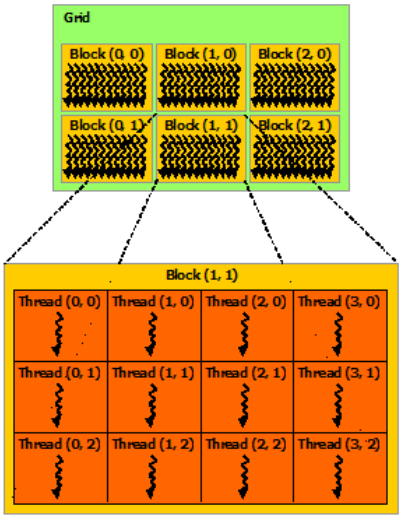


Figure 2.6: The relation between CUDA blocks, the grid and each thread. The figure is taken from [2].

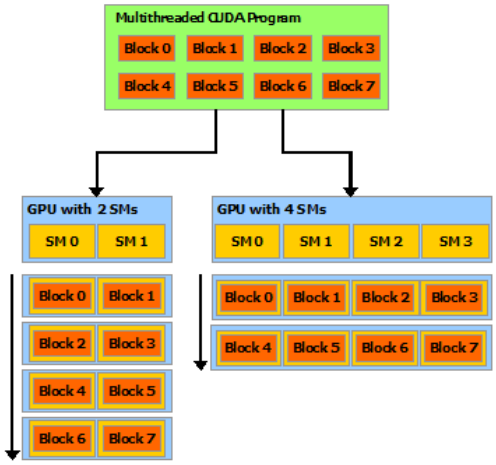


Figure 2.7: Distribution of blocks over the multiprocessors. Here, they are called *streaming multiprocessors*. The figure is taken from [2].

a larger calculation will also temporarily be saved here. This will put an upper limit on the allowable number of threads per block, depending on the number of local variables needed per thread. In addition to the registry is there shared memory. CUDA may allocate arrays or variables in shared memory which will be shared across threads inside a given block. There are some limits to how this shared memory may be accessed. It is divided into *banks* . If two threads access the same bank (but not the exact same memory position), there will be a bank conflict requiring the GPU to perform each memory access in sequence [2].

There are some requirements for optimal memory access [1]. Each multiprocessor has a cache that may load a given amount of bytes. When one thread tries to read four bytes from memory into a floating point variable, the cache will also load the subsequent bytes from memory in one chunk as far as there is space left in the cache (principle of spatial locality [69]). The threads in a warp will all benefit from the same chunk of cache given that subsequent threads must access subsequent memory locations. Thus will the number of cache requests to the slow global memory be lessened. The warps will on the other hand break down if data from subsequent memory locations is not needed or if the needed memory is not properly aligned with the cache reading lines. Reading memory into the cache will in these cases be have to made sequentially, once for each thread [1]. Memory access is said to be *coalesced* when memory access can be done in parallel [2].

The computer to which the graphics card is hooked up to will be referred to as the *host* .

Some difference in the results produced from similar GPU and CPU code is to be expected [2]. The GPU can decide to truncate small numbers to zero in specific settings. The order of the operations will decide the correctness of the results, and it will be important to perform the calculations in the correct sequence in order to produce similar results for CPU and GPU.

Each NVIDIA GPU has a designated compute capability, which describes which features the GPU can use. A GPU program must be compiled for a specific compute capability [2].

2.3 Real-time systems and general concurrency

Some real-time terms will be defined.

There are multiple definitions of a real-time system. One definition is a system where both output and the time at which the output is delivered are equally important [14]. It is distinguished between hard and soft real-time systems. A hard real-time system will require that the results must be output within a specific deadline, otherwise all value is lost, while a soft real-time system can allow for some slack in when the results are output [14]. In a hard real-time system, it must be known that the results always will be delivered within the specified deadline. There must be a timing guarantee in place. The deadline is at which time the results should be ready [14]. Deadline requirements should be known to be fulfilled by mathematical calculation of scheduling, not by running tests.

Tasks to be run, each which its own deadline, can be triggered by different mechanisms [14]. They can be event-triggered, triggered by some external event, or periodic, triggered every other n ms. For event-triggered events in a real-time system, it will be necessary to have bounds on how often the tasks will be released for proper scheduling. Variability in input and output times will be termed by jitter [14].

There are mechanisms in place in operating systems to ensure some extent of real-time behavior [94]. A priority system will be in place to ensure that task needing it are properly prioritized above all others. Preemption will typically be implemented to ensure that lower-priority tasks, either other programs or the operating system itself, is stalled for the benefit of the task in question. Not all of these mechanisms will be implemented in operating systems typically being in use [94].

All common operating systems will have support for threads [94]. Independent programs will be run as threads, and individual programs can implement specific program routines as threads to be run internally in the program, independently. Threads will be scheduled to run concurrently on a single CPU core, with CPU time being given to each thread in turn according to some scheduling system, or the threads may be run in parallel on separate CPU cores. Though only a single CPU core is used, concurrency can still ensure proper utilization as another thread can be run while one thread is waiting for the hard drive or similar [94].

2.4 Image processing

Some image processing techniques have been used.

- Morphology. Structures in binary images can be eroded or dilated using set operations [33]. This can be used to remove artifacts or close holes in binary images.
- Histogram thresholding. An image histogram is a plot over the pixel intensity count of an image. The image can be segmented based on grouping of intensity values either by visual inspection or using mathematical methods [33].
- Smoothing. An image can be smoothed to remove spatial noise through blurring mechanisms. The image may be transformed into Fourier space and filtered using some spectral filter like a Gaussian filter [33].

All above image processing was performed using OpenCV, a C/C++ library for real-time image processing. The above approaches are the simplest approaches possible, and no more advanced methods were looked into as it was beyond the scope of this thesis.

A hyperspectral specific processing tool will be used. Each pixel in the hyperspectral image may be assumed to be a vector in a space spanned by hyperspectral pixels [47]. An inner product can be defined as

$$\vec{x} \cdot \vec{y} = \sum_i x_i y_i, \tag{2.44}$$

where $\vec{x}$ and $\vec{y}$ are two hyperspectral pixel vectors and x_i and y_i their components. Using this definition of the inner product, the length of a pixel vector may be defined as

$$||\vec{x}|| = \sqrt{\sum_i x_i^2}, \tag{2.45}$$

and the cosine of the angle between two pixel vectors as

$$\cos\alpha = \frac{\vec{x} \cdot \vec{y}}{||x|| \cdot ||y||} = \frac{\sum_i x_i y_i}{\sqrt{\sum_i x_i^2 \sum_i y_i^2}}. \tag{2.46}$$

This can be used as a measure of the similarity of between two hyperspectral pixels. This will incidentally be equivalent to some measure of covariance.

2.5 Least squares methods

A mixture may in a linear mixture model be written as

$$\vec{x} = \sum a_i \vec{s_i} + \vec{w} = S\vec{a} + \vec{w}, \tag{2.47}$$

where $\vec{x}$ is a mixture (for example an hyperspectral pixel vector, each entry corresponding to the intensity value of a band), a_i is the abundance of material i, $\vec{s_i}$ the signature spectrum of material i and $\vec{w}$ some noise. This equation may, as shown above, be written in matrix notation with S corresponding to the matrix consisting of the signature spectra. Given $\vec{x}$ and the mixture model S and that $\vec{x}$ must follow the linear mixture model, $\vec{a}$ can be inverted using linear least squares. The least squares error

$$J = \frac{1}{2}(S\vec{a} - \vec{x})^T(S\vec{a} - \vec{x}) \tag{2.48}$$

can be minimized to yield the solution [46]

$$\vec{a} = (S^T S)^{-1} S^T \vec{x}. \tag{2.49}$$

The term $(S^T S)^{-1} S^T$ will be referred to as the pseudo-inverse. For unmixing using a single signature spectrum, i.e. that S contains only a single vector $\vec{s}$, the solution can be expressed as

$$a = \frac{1}{\vec{s} \cdot \vec{s}} \vec{s} \cdot \vec{x}. \tag{2.50}$$

The solution in (2.49) is unconstrained. Negative abundances in a mixture have no physical meaning, and the solution must be constrained to non-negativity.

The usual way to handle an optimization problem like this with additional constraints is to use Lagrange multipliers, but the non-negativity constraint will not allow for the Lagrangian to be used directly. The non-negativity constraint is instead included as an extra term [39]

$$J = \frac{1}{2}(S\vec{a} - \vec{x})^T(S\vec{a} - \vec{x}) + \lambda(\vec{a} - \vec{c}), \tag{2.51}$$

with $\vec{a} = \vec{c}$. This will lead to two equations for which there is no closed form solution [39]. The Lagrangian multiplication vector $\vec{\lambda}$ must satisfy the Kuhn-Tucker conditions in order to yield a valid solution at all [39]

$$\lambda_j = 0, j \in P \tag{2.52}$$
$$\lambda_j < 0, j \in R, \tag{2.53}$$

where P and R are the so-called passive and active sets. Using the equations presented in [39], these conditions can be transformed to the equivalent conditions

$$a_i > 0, j \in P \tag{2.54}$$
$$a_i = 0, j \in R. \tag{2.55}$$

The passive set does therefore represent the solution for which the abundances are greater than zero, while the rest of the solutions must be constrained to be exactly zero. As will be seen is one of the solutions to this problem to fix some of the parameters to 0 and run an unconstrained ordinary least squares on the rest, repeatedly until all conditions are fulfilled.

2.6 Spectral unmixing

The spectral unmixing problem and some potential solutions will here be presented.

2.6.1 The common remote viewing problem

A hyperspectral pixel is in remote viewing commonly assumed to be linearly mixed of signature spectra from different materials, as described by equation (2.47).

An *endmember* is a material present in the scene with a specific signature reflectance spectrum. This can for example be a type of stone present in a hyperspectral image of a landscape [47]. The *abundance* is the amount of the material present in each pixel in the image [47]. These terms will commonly be used throughout this thesis, but "chromophore" will also be used interchangeably for endmember.

Conventional hyperspectral imaging aims to extract abundance maps for each endmember present in the scene directly from the hyperspectral reflectance data. The linear model is often assumed [47]. The unmixing may also be non-linear, where scattering effects between endmembers are taken into account.

The remote viewing community will generally compare reflectances from different materials against the reflectance of the scene instead of attempting a conversion to absorption data. The reflectance level and influence from surrounding materials and even the light source will be varying from image to image, reducing the valuability of endmembers collected on a different day [70].

Most advanced hyperspectral unmixing papers focus on the problem of extracting the material reflectance spectra directly from the hyperspectral image. This is also known as the non-negative matrix factorization problem [16] in other communities. The unmixing problem for the full scene can be written as

$$X = S \cdot A + W, \tag{2.56}$$

where A is the abundance map across the entire image and X is the image. When S, the matrix consisting of the endmember spectra, is unknown, the task at hand is to determine both S and A at the same time. Some [84] will only concern themselves with finding a good estimate of S and estimate A using ordinary least squares, while others [39] will in addition attempt to constrain the abundances to be non-negative.

2.6.2 Abundance estimation

One proposed abundance estimation method is ordinary linear least squares. It will concern itself with finding the best possible mixture without constraints, and the solution will be non-physical.

The algorithm presented by Lawson and Hanson [54] represents the most basic and intuitive algorithm for the non-negative least squares problem, and is a so-called active set method [16].

The Lawson-Hanson algorithm solves this problem in a series of unconstrained, ordinary linear least squares solutions, where endmembers are included into either P or R. The set P contains the endmembers which can be freely varied in a unconstrained, ordinary linear least squares solution, while the endmembers in R are fixed to zero [11]. The Lagrangian vector $\vec{\lambda}$ is calculated for each iteration, where all components will be negative. The endmember corresponding to the maximum component is included in the passive set as it will be closest to zero and likely a good candidate. The endmembers currently in the passive set are set to be a part of an unconstrained linear least squares. Given that some of the components in the solution vector suddenly turn negative, the solution vector is moved linearly through the solution space until all the components turn non-negative [11]. This is continued until the constraints are satisfied, and the next endmember may be moved into the passive set [11]. The iteration is finished when the Kuhn-Tucker conditions are satisfied for the Lagrangian vector. Convergence is proven in [54].

The algorithm will always converge, and convergence is ensured within a number of iterations less or equal to the number of endmembers [54]. The main problem is the computation overload in each iteration. At least one full matrix inversion is needed. The matrix inversion cannot be pre-computed, as the matrix to invert will change every time due to the ever-changing flow of endmember in and out of the active and passive sets. There have been some changes to the main algorithm to alleviate the computation overload somewhat. Bro and Jong [11] restructured the endmember matrix to be able to precompute some parts of the inverse. Benthem et al. [105] does similar operations on the pseudoinverse to optimize and precompute parts of the pseudoinverse before the algorithm is applied, optimized for solving multiple systems at the same time (i.e. for hyperspectral applications). Still, calculating the rest of the pseudoinverse will be required for each individual least squares system, that is, each individual pixel, and not only each individual iteration. Luo and Duraiswami [57] have implemented this for CUDA and released code.

Roberts et al. [80] tried to work themselves upwards by finding the endmembers from an endmember library best fitting the spectra and always adding the endmember that does not lead to any negative coefficients. It will yield a non-negative solution, but not in the least squares sense. Ramsey and Christensen [73] chose to use an ordinary least squares algorithm, but exclude any endmembers leading to negative values. They admit that the residual error will be larger, but they also claim that the solution at least will be physically plausible and that the endmembers leading to negative values generally are not present in the spectra. Franke et al. [27] has a similar strategy by choosing the best endmember to match each pixel. Kim et al. [50] will discuss that equivalent algorithms to the above are unoptimal solutions with no convergence guarantees. None of the above approaches enforces non-negativity in the least squares sense and do not represent good solutions to the problem.

Hyperspectral applications will also often try to enforce a sum-to-one-constraint. This has closed form solutions using ordinary least squares [46], but is more difficult using non-negative least squares. Heinz and Chang [39] presents an algorithm for both extracting endmembers and estimating their abundances using a linear least squares algorithm constrained both to be non-negative and sum to one. This is based on an linear least squares algorithm presented in Chang and Heinz [15], which only constrains the abundances to be non-negative. The algorithm presented here is more or less the Lawson-Hanson algorithm.

Gonzaléz et. al [35, 32] implements the Image Space Reconstruction Algorithm (ISRA) in CUDA for parallelization of the abundance estimation part of spectral unmixing. They had some conflicting requirements for the choice of algorithm, as they chose the algorithm based on its computational complexity to prove the efficiency of the GPU implementation. Vélez-Reyes et al. [107] proves that ISRA is a gradient descent algorithm. Algorithms based on gradient descent are known to be slow to converge [71]. Gonzalez et al. [35] required a large number of iterations. ISRA has been proven to converge to a NNLS solution fulfilling the Kuhn-Tucker conditions [19]. Finding the pseudo-inverse is avoided, but the computational burden associated with the matrix inversion is exchanged for the computational burden of an extreme number of iterations. A large number of iterations is needed, in the order of 2000. Using big O-notation [17], each iteration necessitates $O(e^2o)$ operations, where e is the number of endmembers and o the number of observations.

The algorithm for projective quasi-newton NNLS (PQN-NNLS) is presented by Kim et al. [49]. Their method outperform other alternatives for large problems. This is equivalent to gradient descent, but with some preconditioning and measures to ensure faster convergence. Some endmembers are fixed to zero. Convergence of this algorithm is proven in the same paper.

Franc et al. [26] present the sequential coordinate-wise algorithm for non-negative least squares problems (SCA). All variables except for one are fixed. The optimal solution for this specific coordinate with the respect to the problem is found very efficiently as an analytic solution may be found for this problem. The Lagrangian must be updated for every coordinate update, which requires in the order of $O(e)$ updates. The computational burden will therefore be $O(e^2)$ for each iteration. The convergence of the algorithm is not proven. It has been proven that once the algorithm reaches a fixed point, the solution satisfies the Kuhn-Tucker conditions. The monotonic decrease of the objective function is only remarked as being "obvious" [26].

Chapter 3

Materials and methods

3.1 GPU-MCML

GPU-MCML [4] was used for Monte Carlo simulations, parallelized using CUDA. The implementation was slightly modified in [10]. In the project work [10] an inverse model using the same inverse strategy and GPU-MCML was developed, but this was not used in this thesis. Here, GPU-MCML is used only for forward simulations.

3.2 Camera and computer hardware

The hyperspectral camera the inverse model will be developed for is a HySpex VINIR-1600 camera [41], developed and manufactured by Norsk Elektro Optikk. This is a line-scanning camera using a push-broom technique. Including autofocusing [86], the camera will use 30 ms per line. Each line consist of 1600 pixels (samples) and 160 wavelengths (bands). Camera data is transferred over the TCP/IP-protocol.

The GPU programs were tested on two different computer setups. These are shown in table 3.1. The

Table 3.1: Computer setup

	RAM	CPU	GPU
HP EliteBook 8730w	4 GB	Intel Core 2 Extreme (4 cores)	NVIDIA Quadro FX 3700M
Unbranded computer	6 GB	Intel Core i7 (8 cores)	GeForce GTX 670

operating system used was 64-bit GNU/Linux. The code was not tested in Windows, though compilation for Windows should be possible as only cross-platform libraries were used.

Any single diffuse spectrum measurements of skin presented in this thesis are measured using an USB4000 fiber optic spectrometer (Ocean Optics) [68] and an ISP-REF integrating sphere (Ocean Optics) [67].

3.3 Inversion strategy

The inversion scheme employed for both models is based on previous inversion strategies applied on the diffusion model [93, 53, 77, 10]. The main idea will be to invert the required melanin content and dermal absorption coefficient separately and apply some spectral unmixing on the dermal absorption coefficient to yield the required properties, thus unmixing absorption data buried within scattering.

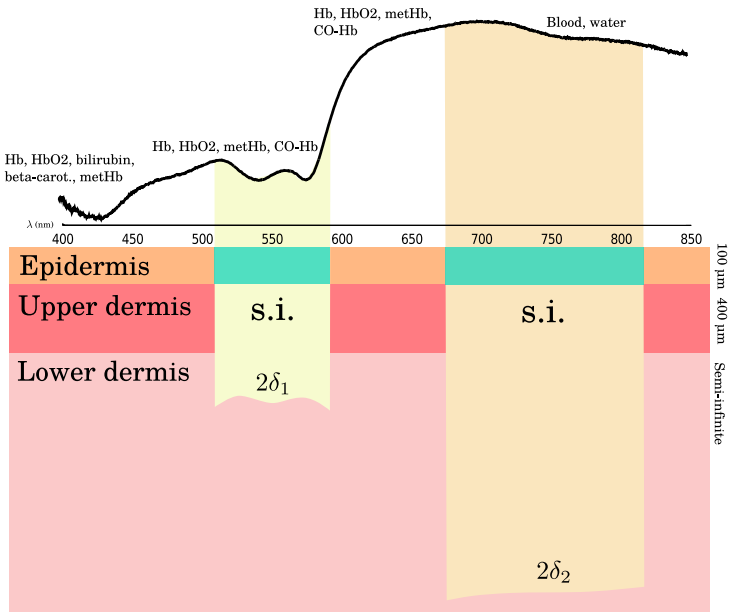


Figure 3.1: Illustration of the two-layered skin model applied to a three-layered situation. The two-layered skin model will approximate multiple, inhomogeneous layers to a single, homogeneous and semi-infinite layer with the derived properties distributed evenly in the layer. Different parts of the diffuse reflectance spectrum will represent different penetration depths. The chromophores taken into account in the different parts is shown above the example spectrum. Typical depths assumed in the three-layer model is shown to the right.

It is desired to keep the number of possible parameters low in order to make the optimization as simple as possible. Within the chosen skin model, the unknown parameters are:

- Melanin absorption in epidermis
- Oxygen saturation in the blood
- Blood volume fraction in dermis
- Depth of the layers
- Additional chromophores

The scattering coefficients are assumed to be known and dependent on the blood volume fraction. They are not mentioned as unknown parameters, though they should strictly also be allowed to vary.

A two-layered model is used in homogeneous parts of the spectrum to yield properties down to some penetration depth, as is done in [53]. See figure 3.1 for an illustration.

Within the two-layer model, the depth of dermis is irrelevant as the dermis is semi-infinite, and it will be applied to different parts of the spectrum to scan the skin down to some penetration depth. The depth of epidermis will be assumed to be known. The justification for this is that this property generally will stay the same within the same area of skin, and any variations can be corrected by correcting the absorption contained therein. Within the model presented by Randeberg et al. [77], it was set between 100 and 200 microns. In this model, 100 microns will be used. The thickness of the upper dermis would have been set to 20% of the total thickness for the three-layered model [75], but total thickness will vary [75].

Only the melanin absorption will be unknown in epidermis, since here the blood volume fraction is a skin model quirk known beforehand. The artificial blood volume fraction in epidermis is so low that it is unlikely that the oxygen saturation here does matter much. Thus, provided the scattering functions can be trusted, the unknown parameters will be the melanin absorption in epidermis, the blood volume

fraction in dermis, the associated oxygen saturation and any additional chromophores.

When the melanin absorption and the blood volume fraction are known, the rest of the unknown parameters can be summed up as the dermal absorption coefficient. The dermal absorption coefficient will for each wavelength remain the single unknown parameter that can be inverted from the spectrum by iteration of the forward model. The constituents of the skin can be recognized by fitting the different absorption spectras against the acquired dermal absorption spectrum. Discrepancies and maladjusted fitting can be controlled and evaluated at leisure.

A different inversion strategy is to assume all parameters unknown and fit them all at once, but this was not used since all chromophores will have to be accounted for beforehand. This approach is also less feasible in a GPU approach, since its increased complexity will cause the approach to be less deterministic. This will later be discussed.

The two-layered model is used for simplicity and for lessened computational complexity, but it cannot be used for the whole spectrum all at once since the skin is not homogeneous. However, the different layers will generally influence different parts of the spectrum due to large differences in absorption and scattering across the spectrum, leading to the spectrum at different wavelengths being influenced mainly by the properties down to some penetration depth. The thought is that the two-layered model can be fitted to different parts of the spectrum having an uniform penetration depth to find the properties down to some penetration depth.

The dermal absorption coefficient $\mu_{a,d}$ and epidermal absorption coefficient $\mu_{a,e}$ can each in turn be inverted using the analytic function for the diffusion model, its analytic derivatives and Newton-Rhapson's method in less than 15 iterations [10], independently of wavelength and pixel. The reflectance is well-behaved with respect to each absorption coefficient, suitable for use of Newton's method [10].

3.3.1 Melanin absorption

The melanin absorption in epidermis must be inverted before the dermal absorption may be found.

Fitting the melanin is problematic. If the skin model in use had been one-layered, could the absorption be extracted and the melanin been separated from the other chromophores using simple unmixing, but the reality is more complex. The melanin is disassociated from the rest of the chromophores by lying in its own layer only containing melanin, and finding the exact, needed melanin absorption will therefore be slightly more difficult as it must be placed in a different layer. Three different methods have been proposed. Only the first was investigated in the project work [10].

- Looking at the general characteristics of the reflectance spectrum to determine how much melanin is present
- Force all the absorption into a one-layered model and somehow extract the correct melanin absorption
- Multi-dimensional fitting

Methods consisting of using the general features of the reflectance spectrum has long been the main method for quantifying the melanin content, and has been used by others [75, 62, 53] to iteratively find the melanin content. Dawson [18] assume the logarithm of the diffuse reflectance to be written as a sum of the absorption and scattering coefficients of each layer of the skin, and develops two indices using this fact to quantify melanin and blood using different parts of the spectrum (melanin and erythema indices, respectively). Melanin is quantified using some reflectance values from 730-800 nm. Dawson's melanin index is given as

$$MI = (\log\left(\frac{R(695)\cdot R(700)\cdot R(705)}{R(645)\cdot R(650)\cdot R(655)}\right) + \alpha)\cdot 100. \tag{3.1}$$

The quantity $R(\lambda)$ is the reflectance at wavelength λ. The constant α was chosen by Dawson to be 0.01. Dawson's erythema index is given as

$$EI = (-\log(R(560)) - 1.5(\log(R(543)\cdot R(576))) - 2.0(\log(R(510)\cdot R(610))))\cdot 100, \tag{3.2}$$

and he corrects it against the melanin index by

$$EI_c = EI \cdot (1 + \gamma \cdot MI), \tag{3.3}$$

where γ is defined as 0.04 by Dawson [18].

Kollias and Baqer [51, 52] presented a method where melanin content is quantified by fitting a straight line from 630 to 720 nm and taking the slope to be the melanin content, similar to Dawson, only using more wavelengths. Stamatas and Kollias [95] will also find a blood quantificator from the blood maxima around 530-590 nm after subtracting the melanin line from the spectrum, and Stamatas et al. [96] will use the blood estimate to correct the melanin estimate by subtraction. It was shown in the project work [10] that such methods will be too influenced by the deoxy hemoglobin to be able to give good estimates, it was too difficult to decouple the effects of blood and the effects of melanin sufficiently. This is also investigated in [95], but the methods proposed to rectify the behavior was not found to be sufficient in the project report.

The second method, forcing the absorption into a one-layered model, will be rife with errors even if the effects of melanin in theory are more readily separateable from the effects of other chromophores. The absorption need only be inverted and unmixed into blood and melanin components. Three different variants are proposed:

- Fit all absorption in a one-layered model
- Fit all absorption in the epidermis of a two-layered model, unmix to yield melanin
- Fit all absorption in the dermis of a two-layered model, unmix, remove derived melanin from dermis, fit melanin to epidermis

These are to be used on wavelength intervals being relatively free of the crosstalk mentioned above. It will be seen that the last one-layered method will be most optimal. The method will be run twice. Initially is the melanin in epidermis set to 100 m^{-1}. This is in the second iteration set to the melanin estimate obtained in the first iteration to gradually make the situation more physical.

There is a choice for how the dermal absorption is set when the melanin is moved from dermis to epidermis. The dermal absorption can either be set to the absorption represented by the extracted parameters except for melanin, or it can be set to the original, derived dermal absorption with the extracted melanin content subtracted. The last alternative will be referred to as "forcing" the melanin from dermis to epidermis, and will only be tested in inverse modelling of simulations. After the epidermal absorption is fit can the melanin be fitted to epidermis either using the melanin curve in (2.40) directly or a straight line fit. The choices between the four alternatives will be investigated.

3.4 Choice of unmixing method

The algorithm for non-negative least squares developed by Franc et al. [26] was chosen for the unmixing stage.

Spectral unmixing will be central to the success of the inversion chain two times during its processing. After the reflectance has been fitted to some absorption model, a spectral unmixing will have to be applied in order to decouple the melanin from the blood. After the melanin is found and the dermal absorption has been inverted, a spectral unmixing will find the components contained therein. Spectral unmixing is central when finding the melanin absorption, and thus important also for the inversion of the dermal absorption coefficient.

Using a linear approach with endmember extraction directly on the reflectances will not yield the desired results. The different chromophores are buried within layers and hidden beneath high scattering. Endmember extraction methods will likely segregate different kinds of skin rather than finding the exact properties.

The previously described skin inversion approach can be seen as a non-linear unmixing problem which in the end leads to a linear mixing problem. The absorption is segregated from the scattering, and

the absorption mechanisms can be seen as independent. The reflectances are obtained in a laboratory situation, and endmember variance is not to be expected. The endmember extraction stage can therefore be skipped, as the chromophore absorption spectra will be known a priori. Only abundance estimation will be needed.

There are two requirements for a good abundance estimation algorithm.

- Should give a non-negative estimate for the coefficients in the least squares sense
- Should be efficient for GPU implementation

Active-passive set methods are passed over. CUDA implementations do exist, but the desired real-time performance will not be possible due to the matrix inversion stage.

Sum-to-one constraints are not interesting. It will be difficult to enforce a sum-to-one constraint on the parameters extracted from the skin in the two-layered model, as it scans down into some penetration depth, though they should have some kind of sum-to-one constraint in the more real, three-layered model.

Gonzaléz et al. did provide code for a CUDA kernel implementing ISRA for BIL-interleaved hyperspectral images, but the code was in their case tested on a higher end Tesla GPU. The hyperspectral image has been allocated in the registry of the GPU, which is not possible for a lower-end GPU as the registry is too scarce. The results obtained will therefore be worse than the running time they obtained.

The PQN-NNLS algorithm has some computational burden in each iteration. Some larger matrix multiplications are needed per system and a prohibitive line search must be done differently for each pixel and iteration. This will cause GPU warp breakdown.

The only matrix needed in SCA's computations is S^TS. This can be precomputed once at the very beginning of the full hyperspectral inversion. This will be small enough for storage in shared memory. ISRA, on the other hand, will require matrices so large that they must be swapped in and out of shared memory, increasing memory lag. This can be avoided for SCA, increasing the performance.

SCA, opposing most other algorithms, had no proven convergence. There is significant risk associated with using such an unproved algorithm should it give unpredictable results. France et al. found SCA to converge within 60 iterations. The author of this thesis found it wise to increase this to 300. It can be used for demonstration purposes until better algorithms with the same advantages are found. Especially will it be seen that using classical algorithms like these on the problem can be ill-posed as the absorptions found are not necessarily correct and some leniency should have been allowed. Using this algorithm, easy to implement and seemingly being able to converge to the correct solution, will be a good demonstrator and a benchmark to compare other algorithms against should this become necessary.

SCA has therefore been chosen as the main unmixing algorithm to implement. Comparing different unmixing algorithms is beyond the scope of this thesis.

3.5 Implementation

The implementation of the hyperspectral inversion framework will here be described. First are some general notes about the GPU memory and data structures presented. Then are the CUDA kernels described, after which the processing blocks encapsulating the CUDA kernels are described. An overview is shown in figure 3.2.

3.5.1 GPU allocation

Recapping, the CUDA functions will be run as blocks of threads. Each block will contain a pre-defined amount of threads, and each thread will access a given position in a given array at a given line of instruction. When neighboring threads access data, the subsequent 32 memory positions in the array will be read into the cache. Given that the subsequent memory positions are the memory addresses the subsequent threads should access at the same time, optimality of the memory accesses will be ensured.

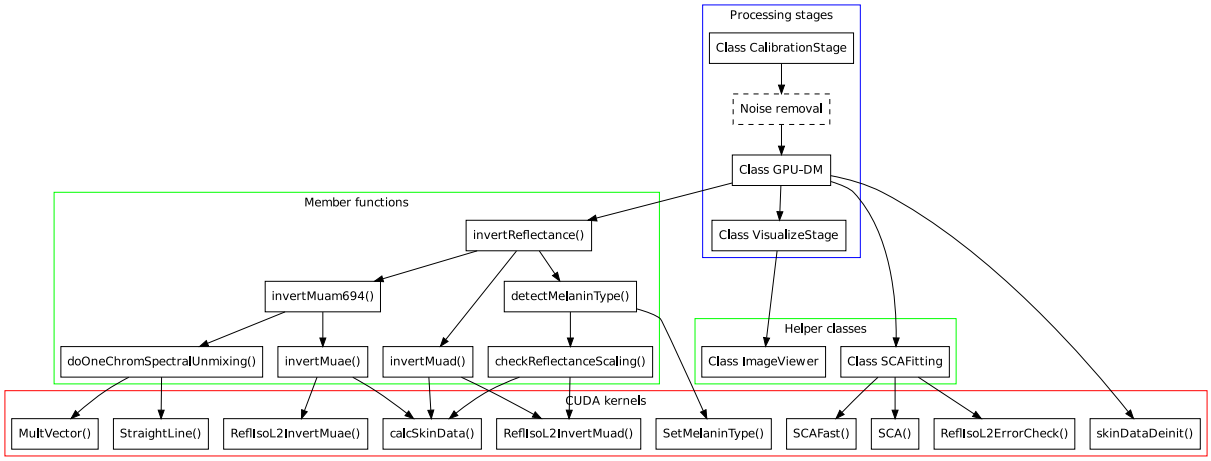


Figure 3.2: Overview over the relation between processing blocks, helper classes, member functions and CUDA kernels.

As long as the most significant thread index match the least significant memory index, this is automatically ensured. However, the memory must also be aligned as the cache will only start its reading at memory addresses being a multiple of 128. The data can for the most part be copied directly from the host to the GPU and the memory access and thread distribution altered accordingly, but to ensure that the first thread also accesses a memory address being a multiple of 128, the arrays are divided in smaller chunks. Each chunk corresponds to a block of threads, and each chunk is allocated with a slightly larger space of memory than is necessary to ensure that the total space used is a multiple of 128 bytes. Memory allocations are therefore rife with the variable **threadsPerBlock** being used to calculate the necessary bytes and the number of necessary memory chunks, referred to as *pitch* .

GPU allocation is done once at the beginning of the program, and arrays are reused.

3.5.2 Data structures

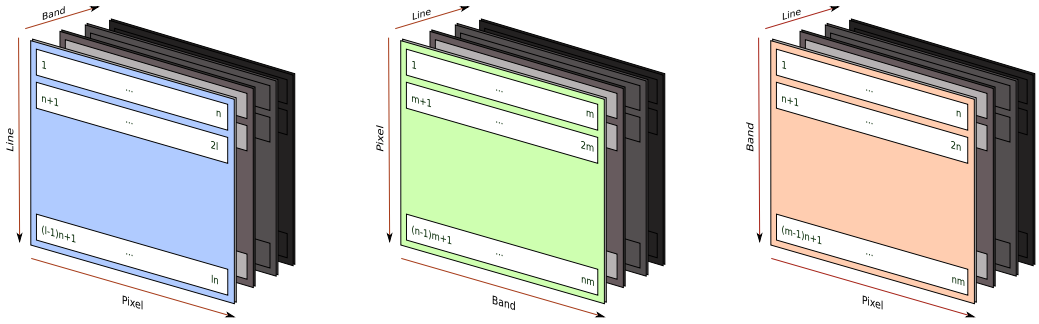
Here, the data structures present will be presented.

Hyperspectral data

With the hyperspectral data arranged in pixels, lines and bands, the natural representation is three-dimensional arrays. However, the three-dimensionality of data arrays allocated on any memory structure is a mere illusion as it is in reality a one-dimensional array arranged in blocks. CUDA needs coalesced memory reads to be efficient, and this requires a control over the memory positions of the arrays beyond three-dimensional handling. The arrays are therefore treated as one-dimensional arrays both throughout the program and throughout this report, with no abstraction into multi-dimensional arrays.

There are three common ways of representing hyperspectral data as one-dimensional data arrays: BIL interleave, BSQ interleave and BIP interleave. These are shown in figure 3.3. A one-dimensional array will start at the zeroth position, and once reaching the end of one row, it will continue at the next row, and when reaching the end of all rows, it will continue in the next "page".

The BSQ (band-sequential) interleave is perhaps the most intuitive interleave when posed by the concept of hyperspectral pictures, as it will arrange the data as one, giant contiguous memory slot per band, with pixel access within each band being the most optimized. Accessing the spectrum associated with each pixel will be less optimal as each band value will be scattered across the memory. BSQ is also not optimal for streaming, as incoming data on a per-line basis will have to be saved at positions far apart in memory space and the full length must be known in advance.



(a) BSQ interleave (b) BIP interleave (c) BIL interleave

Figure 3.3: The different interleaving possibilities for m bands, n pixels and l lines. Starting index 1 is used for convenience in the figure, though starting index 0 will be used in the code and elsewhere.

BIP (band-interleaved-by-pixel) interleave lessens the memory space scattering by grouping the band values associated with each pixel. This will apparently be the most optimal when working on single pixel spectra, but it will later be seen that this is not necessarily the best interleave when working in CUDA.

BIL (band-interleaved-by-line) interleave will group all pixels on a specific line associated with a single band in contiguous memory space. This will be a compromise between BIP's single spectra access and BSQ's intuitive pixel access. Both BIP and BIL are usable for hyperspectral streaming from a push-broom camera as each line can be disassociated from the other. The current hyperspectral camera in use as of this thesis' writing will deliver its data as BIL interleave, although future cameras may use BIP. C code for accessing pixel p, band b and line l in a BSQ, BIL and BIP interleaved array is shown below.

Code 3.1:

```
1 bsq_array[b*numSamples*numLines + l*numSamples + p] = 0;
2 bip_array[l*numSamples*numBands + p*numBands + b] = 0;
3 bil_array[l*numSamples*numBands + b*numSamples + p] = 0;
```

Which interleave is the most efficient depends on the application. For sequential code accessing single pixel spectra, BIP may be the most efficient. BIL will be more efficient if the code is parallellized across pixels, multiple threads should read values at the same time and each thread needs access to all subsequent band values. This will be more clear further below.

BIL, BSQ and BIP interleave can all be converted into each other by a simple matrix transpose along some axis.

Arrays containing chromophore values will be structured in a similar way to the BIL-interleaved hyperspectral data, although with the number of chromophores along the band axis. Arrays with one value assigned to each pixel will also be stored in the same way.

Convenience structs

The CUDA functions will require several arrays for storing optical properties, specific chromophore values and more. These are grouped within structs defined in **base.h**.

- **GPUSkinBase** contains base arrays for the wavelength-dependencies of different absorption and scattering spectra. Its arrays is used by **calcSkinData** to calculate the optical properties resident in **GPUOpticalProps**.
- **GPUOpticalProps** contains the spectral and spatial dependencies of the calculated absorption and scattering spectra associated with each pixel in the hyperspectral image. Its allocation and interleave matches the hyperspectral image exactly.

- **GPUSkinData** contains the basic skin data associated with the skin present in each pixel: The melanin content, oxygenation, blood volume fraction and melanin type. It is input into **calcSkinData** to calculate the **GPUOpticalProps** described above, and decided through some other means before **calcSkinData** is called.

Each of these data structures have an **allocate()** function used to allocate the arrays on the GPU using the correct sizes and pitch. The input parameters common to all is the number of bytes associated with data that should be accessed by one block of CUDA threads, and the total number of these blocks (the multiplication of these two numbers will yield the number of bytes associated with one line of data and one band). This approach is low-level and not fail-safe, but more sophisticated methods of inputting the desired sizes and automatizing the process was not implemented as these arrays are at most only used once and mucking about with them should not be necessary.

Chromophore arrays

Arrays containing the wavelength-dependencies of the absorption spectra of the different, possible chromophores are input as a matrix into the functions dealing with spectral unmixing. The output is a nameless chromophore array containing the abundances of each chromophore at every position in the hyperspectral line of data, in the sequence defined by the input chromophore absorption matrix.

In order to overcome the namelessness, the sequence of the chromophores in the chromophore matrix is assigned in a given, well-defined order through the class **Chromophores**, defined in **chromophores.h**. For example will oxy hemoglobin always be located at the first position and deoxy hemoglobin at the second.

After a **Chromophores** instance is created, the different chromophores may be set using **set*()** and **unset*()** functions, for instance **setMethb()** for including methemoglobin. The chromophore absorption values at a specific wavelength can be output as a array containing the data in a well-defined order using **getAbs()** and **getAbsArray()**. Given that the **Chromophores** instance is unchanged, it can be used to trace back what chromophore a given, derived chromophore value belongs to. Should any absorption spectra be changed or any new chromophores be added, this is the class to change.

3.5.3 CUDA kernels

The main part of the program is the CUDA kernels. Most of the other parts of the code are wrappers, convenience functions and access functions to ease the cumbersome parameter choices for the CUDA kernels.

Overview and choice of interleave

There are mainly two different tasks at hand in the inversion chain which each require different parallelization strategies.

- The inversion of the absorption coefficient μ_a for each wavelength and pixel
- Spectral unmixing of the absorption spectra across all pixels

Spectral unmixing, if done on a pixel basis and making no special use of any spatial information, will only require that the spectral information associated with each pixel is readily available. The simplest possible spectral unmixing will be to use ordinary least squares, which requires that the pseudoinverse of the matrix containing the chromophore absorption arrays is multiplied by each spectral vector. There are mainly two ways to parallelize this.

1. Assign one thread for every pixel and band
2. Assign one thread for every pixel

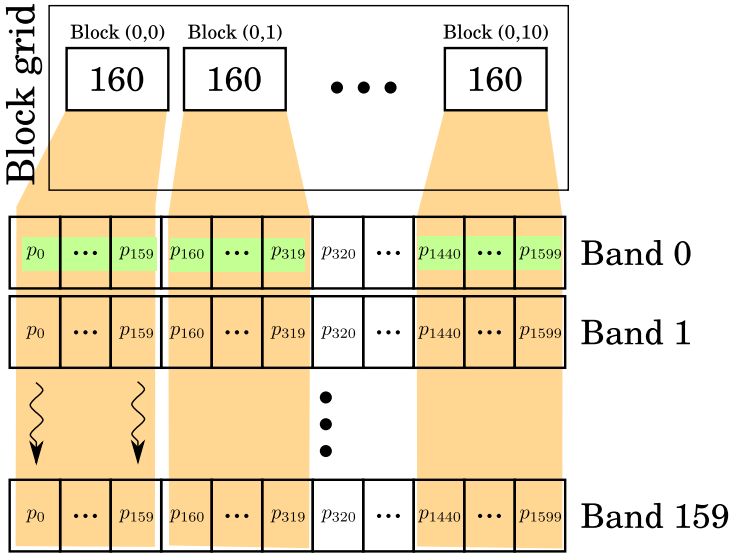


Figure 3.4: Spectral unmixing as parallelized in CUDA for BIL interleave. Threads are assigned in blocks of 160, each dealing with the unmixing of an independent pixel. At each step where values are read in from the hyperspectral data array at a specific band, the memory reads are coalesced across threads. Any matrices dealing with chromophore absorption values are read into shared memory and broadcast across the threads within a specific block.

The latter possibility is less cumbersome to develop and maintain. Each thread will take care of its own spectrum and unmix it. For neighboring threads to read neighboring memory addresses, BIL interleave must be used for reasons illustrated in figure 3.4. The unmixing algorithm must, in some way, loop through the band values associated with each pixel. In each instruction of the necessary for loop, data will be loaded from and saved into data arrays across threads. For this to be done in a coalesced way must each thread access neighboring pixels for a specific thread, which is only possible for the BIL interleave.

On the other hand, BIP interleave must be used if parallelization also is applied across bands. The reasoning is that the resulting chromophore variables must be written to shared variables as the threads now will cooperate to calculate the chromophore values, and sharing of variables can only be done in a fast way within a block of threads. One block of threads must therefore deal with one or more spectra, and each thread will read one band value associated with a specific pixel. Neighboring threads will require neighboring band values in memory space, and BIP is therefore the most optimal interleave for this situation. For linear least squares, the latter, completely parallelized approach is feasible as the order of the matrix multiplication is irrelevant and race conditions are avoided.

It has been clear that ordinary linear least squares is not suitable, and a more convoluted non-negative least squares is warranted. SCA, the algorithm which primarily will be used, requires at first a least squares-like matrix multiplication which technically could have been implemented using the above strategy. Once this is finished, it will run through some matrix multiplications where the matrices only are of size **numEndmembers** $\times$ **numEndmembers** and with no potential parallelization across wavelengths, only pixels. Therefore, BIL interleave is required for this particular algorithm, and BIL interleave has been used throughout the program. BIP will be warranted if spectral parallelism is required.

No such measures regarding interleave are required for the inversion of the absorption coefficient for each layer, as the inversion at each pixel and band is completely independent. Only at the unmixing stage are the bands no longer independent as the wavelength-dependencies become important. The inversion functions themselves can therefore be used regardless of the interleave, as long as the **calcSkinData()** function and associated **GPUSkinBase** arrays have the correct behavior with regards to the interleave and the kernel is correctly called from the host.

There is also another caveat in BIP interleave, namely the free wavelength choice. The two-layered diffusion model is used to fit the spectra, and the unmixing can only fit parts of the spectra at the

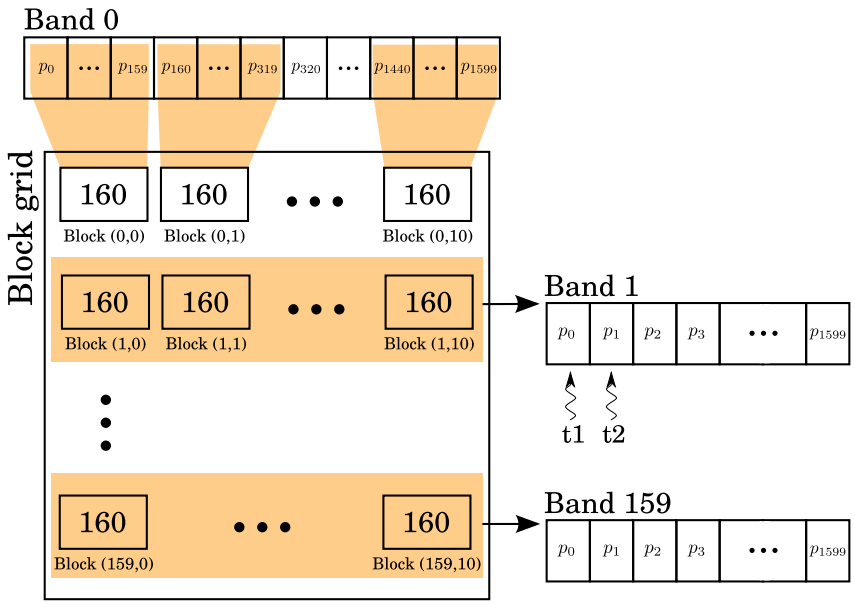


Figure 3.5: Inversion of absorption coefficients for one hyperspectral line of data: CUDA block and thread distribution across wavelengths and pixels for BIL-interleave. For this particular GPU, threads are assigned in blocks of 160 subsequent pixels in a particular band to ensure memory coalescing. Blocks are arranged according to band. Each thread inverts its own absorption coefficient independently of the others, one for each pixel and band. Figure is modified from [10].

same time due to it requiring an uniform penetration depth. The melanin determining stage will also use unmixing and absorption inversion only at a specific wavelength range, for which inversion at any other wavelengths will be a waste. Choosing wavelength ranges for BIL interleave is convenient, as the pixels associated with a specific wavelength are grouped, and the pixels associated with the undesired wavelengths can be dropped with no repercussions for the coalescing of the memory reads. See figure 3.5 for thread distribution for this case.

The dropping of wavelengths will for BIP interleave lead to holes in the memory array. For coalescing to still be in place, the size of the wavelength interval must be a multiple of 32, and to ensure memory alignment, each interval must start at a multiple of 32. This severely limits the flexibility of the wavelength ranges, which will cause problems when the penetration depths for these possible choices is not uniform enough for good fits and parameter extractions. The goodness of fit will also be heavily dependent on the calibration of the camera as the specific wavelength associated with the possible wavelength index can be subject to change.

BIL interleave has similar problems when the free pixel range choice is desired, but this is less of a problem. It will be desired to drop outlying pixels to speed up computations, but here can entire blocks of pixels be dropped since a fine-grainedness is not desired. Dropping pixels in this way will also have no repercussions on the index calculations, as this is only dependent on how the CUDA kernel is called from the host and automatically propagated to the built-in **gridDim** variables.

Due to all this, BIL interleave is therefore the desired interleave for this particular application.

Model routines

Routines related to the diffusion model and the transformation from skin data to optical properties are

- **__global__ void calcSkinData()** - inputting skin data, optical properties are calculated
- **__global__ void ReflIsoL2InvertMuae()** - inversion of $\mu_{a,e}$ with respect to the input reflectance

- **__global__ void ReflIsoL2InvertMuad()** - inversion of $\mu_{a,d}$ with respect to the input reflectance

, and all defined in **inv.cu** and **inv.h**.

The function **calcSkinData()** uses the base arrays for the wavelength dependencies of the absorption and scattering functions of different chromophores to calculate the optical properties based on input oxygenation, blood volume fraction and melanin absorption at 694 nm. The melanin type may also be chosen, and the exact melanin absorption is calculated based on the value of the melanin type array, containing one of the values in

Code 3.2:

```
1    enum MelaninType{PHEOMELANIN, EUMELANIN, SVAASAND};
```

To avoid any if statements, the melanin calculation stage is written as

Code 3.3:

```
1 MelaninType melanintype = melanintype_arr[index];
2 float mua_melanin = \
3     muam694*((melanintype == SVAASAND)*powf(fdividef(694.0f,wlen), 3.46) + \
4     (melanintype == EUMELANIN)*expf(-kEu*fdividef(wlen-694.0f,694.0f)) + \
5     (melanintype == PHEOMELANIN)*expf(-kPheo*fdividef(wlen-694.0f,694.0f)));
```

This ensures maximum parallellization, warps are ensured since no thread-dependent if statements are used. The extra calculations will also not be evaluated when the statements evaluate to false. The current wavelength is read in as a shared variable across all the threads in the current block through

Code 3.4:

```
1 float wlen;
2 if (threadIdx.x == 0){
3     wlen = wlens[blockIdx.y+startblockind];
4 }
5 __syncthreads();
```

The alternative would be to let all the threads read their own copy of the current wavelength from an array containing multiple copies of the wavelength array as to ensure memory coalescing, but as it turns out, the above approach with a warp breakdown and a broadcasting of the current wavelength across all threads is a faster approach. The scattering functions are calculated using the same wavelength variable.

The function **ReflIsoL2InvertMuad()** uses the above calculated optical properties to invert the input diffuse reflectance. Originally, in [10], this function was split into two parts - one function calculating the diffuse reflectance and its derivative based on the optical properties, and one function calculating the next required dermal absorption based on the derivative and reflectance value. These kernels were repeatedly called from the host for a given amount of iterations. It became clear that this caused a time lag as the functions needed to read and write to global arrays to be able to communicate. The functions was instead combined in one function to do the iterations in a self-contained way and only read from the global arrays in the initialization, and write to a global array containing the dermal absorption values at the very end. Meanwhile are values read from and written to registers, which gives a performance boost.

To increase readability and reuse of the code, the reflectance calculation itself could have been moved to a device function, for example **__device__ float ReflIsoL2Calc()**, but optimizing calculation times was prioritized over readability. One inversion is compute bound, and register usage is maximized. Derivative calculations are especially fierce, and values are therefore reused between reflectance- and derivative calculations. It did in addition become clear that calculating the diffuse reflectance as a single code line into a variable made the numerical inaccuracies large due to truncation to zero, and the calculations had to be split into different variables to keep control over this zero truncation. Calculations related to the epidermal absorption coefficient were also placed outside of the iteration for loop.

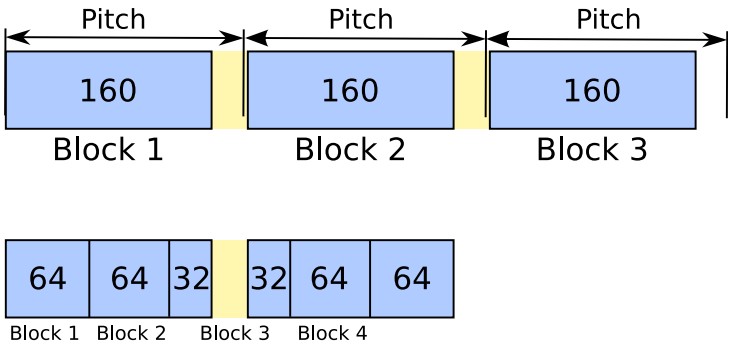


Figure 3.6: Pitch problems when moving from one threads per block size to another while optimizing the memory arrangement only for the first block size. Above is how the memory will be divided for the situation for which it is optimized for, below is the situation when the block size is smaller, say, 64 threads per block. Block 3 will be split across one pitch.

Similar optimizations are done also for **ReflIsoL2InvertMuae()**, the CUDA function for inverting the epidermal absorption coefficient with respect to the diffuse reflectance.

Calculating the derivative of the diffuse reflectance with respect to the epidermal absorption is more complicated than the dermal absorption, and requires a higher register usage still. The default threads per block was set to 160, but **ReflIsoL2InvertMuae()** required 64 to avoid register spilling. This would split a pitched chunk of memory in multiple parts, and one block of threads will necessarily have to access memory from two different blocks with the non-allocated pitch in-between. Some creativity was therefore applied in the index calculations. The situation is shown in figure 3.6, and the solution below.

Code 3.5:

```
1 int egBlockInd = (blockIdx.x*64 + threadIdx.x)/160;
2 int ind = ((gridDim.x*(blockIdx.y+startblockInd))*pitch*2)/5 + egBlockInd*pitch
      + blockIdx.x*64-egBlockInd*32*5 + threadIdx.x;
```

This particular solution is not applicable for any other threads per block combinations than 160 and 64. This is, as such, not a very dynamic solution. On the other hand would a "dynamic" solution be complicated as this type of solution would work only when the threads per block sizes are divisible and fulfill the alignment and coalescing requirements. It is also possible to convert arrays from one pitch optimization to another, but this was not tested as it was known that some milliseconds would be wasted on such an operation. It would however only have to be done once at the start every line processing.

Both inversion functions and **calcSkinData()** takes **startblockInd** as an input argument, which is used to specify at which memory block the calculations should start. This is used to specify the first wavelength in the wavelength range, and the size of the wavelength range is specified by the grid dimensions. The situation is shown in figure 3.7.

There are in addition two more functions, **__device__ float ReflIsoL2Calc()** and **__global__ void ReflIsoL2ErrorCheck()**. Though it was argued above that having a device function for calculating the diffuse reflectance is unnecessary, it still is implemented for use where appropriate to reduce the duplication of code. However, there is only one function where it is used, **ReflIsoL2ErrorCheck()**. This works on a block grid similar to the grids used for the unmixing routines, that is, it is initialized only across pixels and wavelengths are covered through for loops. It calculates the root squared mean error of the calculated diffuse reflectance as compared to the input, measured diffuse reflectance for a specified wavelength interval.

When other chromophores than blood and melanin are used, **calcSkinData()** is only used to calculate the scattering coefficients and epidermal absorption. For the dermal absorption, the values are calculated manually by multiplying the required chromophore coefficients by the chromophore absorption values and

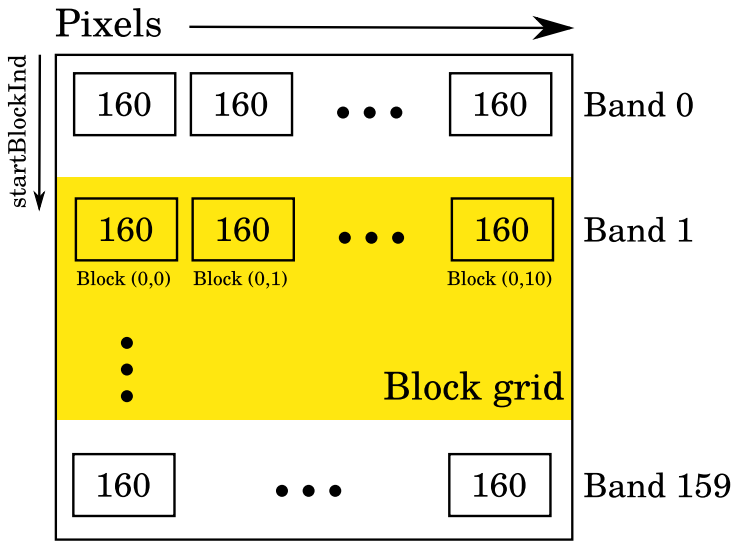


Figure 3.7: How the wavelength ranges are chosen within the larger memory grid. Grid sizes for CUDA are chosen to match the wavelength range, and an input variable startBlockInd is chosen to denote the grid displacement with relation to the memory blocks.

summed, either on the host or the GPU if the chromophore arrays are available. This will be described in detail below.

Unmixing

In the initial creation throes of this implementation was ordinary least squares used to unmix the absorption spectra, in particular through the use of the CUDA kernel **GPUlseunoptim()**. It takes as input argument the pseudo inverse of the chromophore matrix, $(S^T S)^{-1} S^T$, needing only a matrix multiplication for calculating the results. As matrix multiplication is a rather well-known problem for which optimizations extensively has been reviewed by others [2], no special optimizations using shared memory or the like were implemented for this function since it only was used for initial testing. It also uses the linearly interpolated absorption spectra, since it was thought that the spectra had to be interpolated to give out meaningful results, but this was only due to noise. Interpolation wavelengths and indices were determined beforehand.

Non-negative least squares fitting was implemented in

- **__global__ void ISRA()**
- **__global__ void SCA()**
- **__global__ void SCAFast()**

The ISRA is a modified version of the implementation presented in [34]. It was modified to use global memory instead of the registers for the hyperspectral data arrays, since the heavy register usage would only be possible for the Tesla GPU they used. Some indexing was also changed to match the implementation choices.

More resources were put into the development of **SCA()** and **SCAFast()**. The kernel **SCA()** is an implementation of the SCA algorithm for non-negative least squares. SCA's selling point is that each iteration of the algorithm requires few computations,. The matrices in use are of **numChromophores** × **numChromophores** size instead of the full **numBands** × **numChromophores** variant. To increase these selling points, the matrices and arrays are saved as shared memory arrays to throw away excess global memory access.

Code 3.6:

```
__shared__ float sh_x[MAX_ENDMEMBERS][MAX_BLOCKDIM];
__shared__ float sh_mu[MAX_ENDMEMBERS][MAX_BLOCKDIM];
__shared__ float sh_H[MAX_ENDMEMBERS][MAX_ENDMEMBERS];
```

The constant **MAX_ENDMEMBERS** is initialized as according to the upper limit of allowable shared memory on the device. Since every thread operates on a single pixel, each thread needs its own chromophore array, and therefore there are **MAX_BLOCKDIM** number of chromophore arrays. Both of these values are input as template arguments, for just in time compilation of the CUDA kernel. Three 10×160 float arrays will approximately be equivalent to 12 KB, which is 12 of the 14 allowable KBs of shared memory in the GPU. The GPU will be unable to assign a different block to a given multiprocessor when one block already has been assigned and had its shared memory allocated. The GPU will hence not be able to switch blocks in and out of the multiprocessor to reduce memory lag. Still, this solution is faster than reading from and writing to global memory arrays for a given number of iterations, as will be seen.

The arrays x and μ are initialized according to the SCA algorithm in the start of the kernel. Common matrix values across threads is loaded into shared variables before multiplication.

The $S^T S$ is read into the shared array **sh_H**, which is commonly used across all threads, and **sh_x** and **sh_mu** are iterated as according to the SCA algorithm for a given number of iterations **NUM_ITERATIONS**. The results are after this written to global memory.

As it goes, this solution does not scale well with GPU computing power since the amount of shared memory does not change much from GPU to GPU. A new CUDA kernel, **SCAFast()**, was therefore developed to use the registers instead of the shared memory for **sh_x** and **sh_mu**. The registers scale better with GPU power even if the registers also are a scarce resource. CUDA is not allowed to create arrays in register space unless all register use can be predicted at compile time. Otherwise, the arrays are allocated in shared memory space, with no unique array for each thread. It is possible to do this by for looping over constants and unrolling the loops, but the behavior was found too unpredictable. Each position in the array is instead named explicitly by **x1**, **x2**, etc. This would also imply an unmaintainable mess of copy-pasting of code instead of using for loops, but this was alleviated by encapsulating some of the redundant code within macros. For example is the calculation of μ in a given iteration given by the macro

Code 3.7:

```
#define CalcMu(Arg) k=Arg;\
    addition = (temp - xprev)*sh_H[k][j];\
    mu##Arg += addition
```

, which is called whenever the copying and pasting of the same code would be warranted. This is not a very maintainable solution, but the extra milliseconds gained on a powerful GPU with many registers warrant the hairy optimizations.

The kernel is defined to handle up till 7 endmembers by manually checking the supplied **MAX_ENDMEMBERS** template constant. The used if statement will have a common outcome on all the threads in the warp. When the kernel is called, the unnecessary register variables are optimized away in the just-in-time compilation. Several copies of the same kernel is compiled and used, each with a different register usage based on the number of endmembers in use.

There is one change in the algorithm compared to the generic algorithm. Methemoglobin is either included or dropped based on whether a supplied, previous methemoglobin value lies above a threshold, for the cases where it is to be a part of the fitting. The thought is that methemoglobin can be fitted at a wavelength range where methemoglobin has distinct spectral features, and included in the fitting where methemoglobin has severe crosstalk with melanin only when the presence of methemoglobin is certain. For the wavelength ranges where methemoglobin has distinct features, the function is initialized with a dummy methemoglobin array always evaluating above the threshold. The methemoglobin index of the previous array is supplied, since it is difficult to split the chromophore arrays into separate arrays for

the different chromophores. The exact, previous methemoglobin value is looked up in the previous array based on this in a coalesced way.

Code 3.8:

```
1 bool hasMethb = prevChrom[prevMethbind*gridDim.x*pitch + pixind] > METHB_TRESH;
```

The corresponding initial μ is either set to the correct value or zero according to this boolean value when the index matches the position which represents the methemoglobin value. This index is set to -1 and the below statements evaluates to true regardless, if methemoglobin is not a part of the chromophores to be unmixed.

Code 3.9:

```
1 float mustart = ((i != currMethbind) || (hasMethb))*(-1*temp);
```

Other chromophores were not treated the same way as they either were spectrally distinct or otherwise necessary and always present in normal skin.

Other CUDA kernels are used to fit the epidermal absorption coefficient to the melanin. The kernel **StraightLine()** is used to fit a straight line to the input absorption. It must be initialized across pixels, and will go through the wavelength interval in a for loop and calculate the straight line coefficients as according to well-known formulas [108]. The specific wavelength is needed, read in from a wavelength array and broadcast to all threads in a shared variable. The melanin coefficient is written to a result array as the straight line value at the supplied wavelength, usually 694.0f.

The kernel **MultVector()** is used to calculate the necessary melanin coefficient when the melanin curve is fitted to the epidermal absorption, as according to the formula in equation 2.50.

The kernel **calcOxyBvf()** is used to calculate oxygenation and blood volume fraction values from input derived chromophore arrays. It is assumed that oxygenated and deoxygenated must, respectively, be placed in the first and second array position of the chromophore array pointed to at the current pixel, which is ensured by the **Chromophores** class. The blood volume fraction is not divided by $\frac{H}{H_0}$ to ensure the correct physical meaning of the blood volume fraction as the difference is minimal and it is only used for the scattering calculation.

A typical work-flow for the inversion of a single hyperspectral data line using these CUDA functions is shown in figure 3.8.

Others

There are other CUDA kernels not strictly related to the inversions.

- **skindataDeinit()**
- **CheckReflectanceScaling()**
- **SetMelaninType()**
- **removeMuam()**

The kernel **removeMuam()** is used to combine absorption coefficients multiplied with their absorption values after an unmixing has been run, all the while removing one of the chromophores which originally was a part of the unmixing process (i.e. melanin). This is used after a dermal unmixing in the melanin inversion process to yield a dermal absorption which may be held constant while the epidermis is iterated. This is initialized across pixels and wavelength like the inversion kernels, and loops over all endmembers to sum the absorption coefficients. Wavelength dependencies of the chromophore absorption coefficients are read in from a chromophore matrix allocated in the same way as for the unmixing functions. This function is implemented in order to have a more flexible way of including other chromophores than having to reimplement **calcSkinData()** every time and gradually make it slower and slower due to increased register use, and also complicate the input arguments.

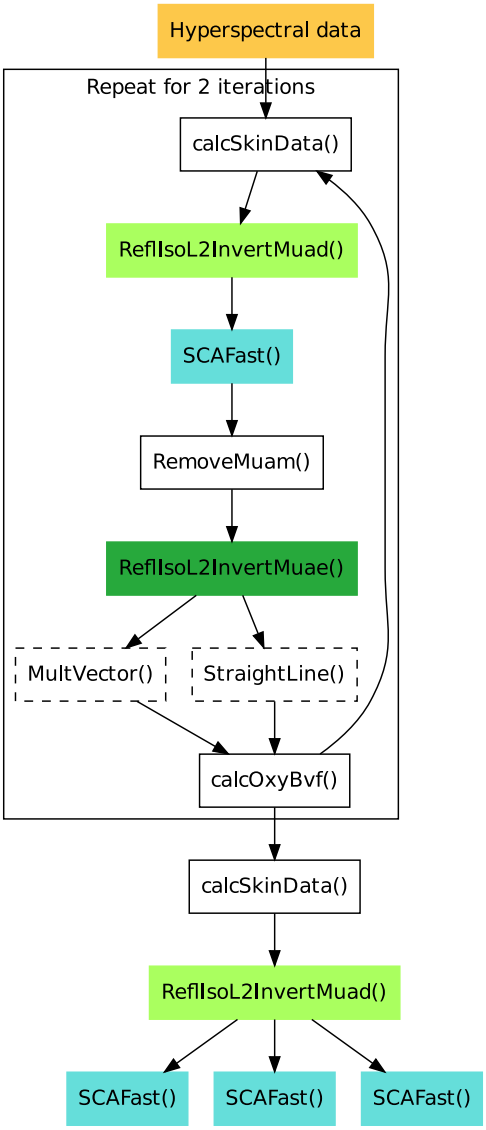


Figure 3.8: The sequence of CUDA operations for the inversion of a hyperspectral line.

Some preliminary support for changing the melanin type on the go is implemented, which will be discussed later. The kernel **SetMelaninType()** is used to set all the pixels to have the same, initial melanin type. The kernel **CheckReflectanceScaling()** calculates the scaling of the absorption at two wavelengths and increments the melanin type based on some threshold, as determined from the figure presented earlier in 2.4.

All arrays are meant to be reused for each line of hyperspectral data since the allocation stage is time-consuming. The kernel **skindataDeinit()** is therefore used to deinitialize the used arrays to the standard values. The melanin coefficient is set to 100 m^{-1}, the bvf and oxy to 0.01 and 0.8 and the melanin type to the Svaasand model.

3.5.4 Processing stages

Instead of calling the CUDA kernels manually, they are embedded in a hyperspectral streaming framework developed by FFI to ensure modularity and reuse of code. A non-disclosure agreement has been signed with FFI, and specific details around the implementation cannot be disclosed. The framework will have processing blocks. Data is streamed in and out of the processing blocks through input and output ports.

The inversion of the reflectance with respect to the chromophores is structured around a self-contained processing block. This takes the hyperspectral reflectance data as input and outputs either the unmixed result or the dermal absorption coefficient after a full inversion chain. Because of this, some of the methods contained within are not reusable as new processing blocks. The individual fitting routines are however implemented as self-contained classes which can, theoretically, be split off as separate processing blocks if desirable. The alternative was to split all sub-stages within the processing block as separate processing block and let GPU data be streamed between the blocks, however

1. it would become needlessly complicated and require much redundant framework code
2. as the GPU memory allocation takes too much time, it would need to pre-allocate GPU memory for the hyperspectral transferral between blocks, and this would cause potential deadlocks or the like when handling when to terminate a piece of memory or not
3. the individual stages required a specific order of calculation, not requiring the flexibility served by having individual processing blocks
4. the GPU would not be able to run any of the GPU-specific code in parallel, and as all the stages encapsulate GPU code, having separate blocks becomes less meaningful

The individual stages within the processing block were therefore written as flexible as possible for potential later breaking off into separate processing blocks instead of writing it all as individual processing blocks from the get-go. The latest CUDA compute capability has possibilities for running multiple streams in parallel. It will in the future be feasible to invert multiple lines in parallel even if the individual stages depend on the former stage. Still, it does not warrant for the processing stage to be split up in several processing stages yet and it is as such kept in one piece.

The processing block containing the inversion stage, from now on referred to as GPU-DM, is a C++ class derived from several classes in FFI's framework. There are some functions handling the framework business dealing with input and output ports, but the important parts are the member functions handling the inversions and the member classes handling the fitting routines.

There are also other processing stages for calibration, sample size massaging and visualization, shown in figure 3.9. These will in due course be presented. Noise removal is mentioned in the figure, but was beyond the scope of this thesis and never implemented. This particular processing stage will be implemented by others after this thesis' work is over. Noise removal was instead performed in advance on calibrated data.

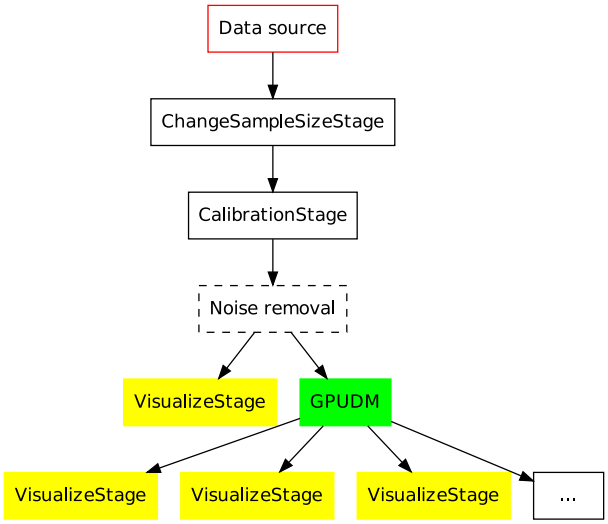


Figure 3.9: Real-time inversion chain from data source to visualization.

GPU-DM

GPU-DM is the only processing stage calling CUDA kernels. Its constructor therefore deals with much of the GPU memory allocation. Here, the wavelength ranges for which unmixing is to be performed are defined as input arguments into the class doing the unmixing itself, along with a definition of the chromophores to be used in the unmixing of each wavelength range.

The processing block functions and associated, although separate, helper classes will be considered together.

Inversion The different parts of the inversion chain are implemented as separate member functions.

- invertReflectance() - called from execute() to invert the reflectance into the dermal absorption coefficient
- invertMuad() - given the current optical properties, invert the dermal absorption coefficient. This calls the corresponding CUDA kernel, and the member function is mostly for encapsulating some grid initialization and the call to calcSkinData(), avoiding code redundancy.
- invertMuae() - same as the above, only for the epidermal absorption coefficient.
- detectMelaninType() - calls the CheckReflectanceScaling() kernel to decide melanin type.
- invertMuam694() - calls the combination of the above in the correct sequence to invert the melanin content.

The **execute()** function, upon receiving hyperspectral data, calls **invertReflectance()**. This function takes the reflectance to be inverted as an input argument, which makes it able to invert a given reflectance array also outside the **execute()** function. This is used to display the spectrum along with the fit when clicking on the **ImageViewer** widgets, to be later discussed. The **invertReflectance()** function uploads the data to the appropriate GPU allocated array, and calls **invertMuam694()** to find the melanin content. After this, **detectMelaninType()** is optionally called to detect the melanin type, and subsequently **invertMuad()** called to invert the dermal absorption, before the fitting function of each **SCAFitting** instance is called to unmix each desired wavelength interval.

The melanin inversion function calls **invertMuad()**, the SCA fitting function **doFitting()**, **invertMuae()** and single spectrum unmixing **doOneChromUnmixing()**. This is all done for a predefined amount of iterations. If a printing flag is set, for when the inversion function is called from outside the execution loop, are different properties printed to file in-between inversions.

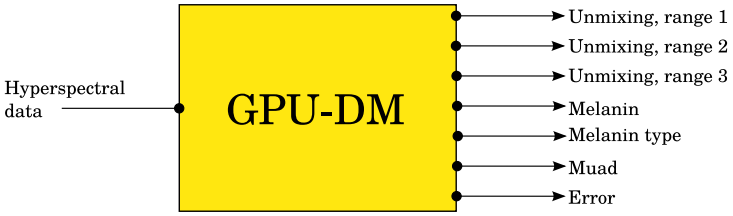


Figure 3.10: Input and output ports of the GPU-DM stage.

SCA is applied on each predefined wavelength interval when the dermal absorption coefficient has been found, and the output is downloaded to the host and sent through the output ports. The sequence of operations is the exact same one as previously described in figure 3.8, although now the various CUDA kernels are called from different member functions of the class, for code reuse in the initialization of the CUDA kernels.

Some potential for concurrency between the host and GPU is erased, since the CUDA kernels are not called from the main thread but from separate member functions. If this had not been done, the host could have prepared the initialization of the next CUDA function while the current CUDA function was running. It will now have to wait for the CUDA kernel to end before exiting the calling member function and moving on to the next, potentially creating some overhead between each CUDA kernel. However, the potential lag is small enough to be negligible as there is no major CPU work to be done other than the initialization. Readability can thus be prioritized over optimality.

SCAFitting Fitting using SCA is something which could have been implemented as a member function in GPU-DM. However, it needed a larger set of GPU arrays and one set for each wavelength range to be unmixed, warranting separation into a different class, **SCAFitting**. It takes care of allocating its GPU arrays as according to the space needed by the wavelength range to be unmixed, and creates the chromophore array as according to the supplied **Chromophores** class instance. It needs some knowledge of the pitch and threads per block used elsewhere, given as input arguments in the constructor.

The **doFitting()** function is used to call either of the CUDA kernels **SCA()** or **SCAFast()**, their template arguments correctly initialized based on the number of chromophores present. Based on some input flags is the result of the unmixing combined to a dermal absorption coefficient (minus melanin) using the kernel **removeMuam()**, used for later inversion of epidermis, and it may also call the kernel **calcOxyBvf()** to update the oxygen saturation and blood volume fraction in the input **GPUSkinData** struct based on the unmixed results. This behavior does not strictly belong in this class, but it was the most convenient with the chromophore absorption arrays being readily available in this class.

The member function **reflerror()** may, based on the presently unmixed results, calculate the reflectance error using the kernel **ReflIsoL2ErrorCheck()**. This does not strictly belong to such a class either, but as before, the chromophore arrays are readily available.

Framework sanity GPU-DM has quite a number of outputs, since unmixing is done at three separate wavelength ranges, in addition to the dermal absorption being a valid output by itself along with the melanin absorption and the melanin type. The input- and output ports are shown in figure 3.10.

All the other stages are rather rudimentary, but will still be presented. None of these were optimized much, as the focus point of this master's thesis was thought to be the above **GPUDM** class and the rest only for convenience when investigating the results. Ideas and code will can still be used for further work.

VisualizeStage

The class **VisualizeStage** is a stage for visualizing the results from **GPUDM**. This uses Qt [72], a graphical toolkit, for handling a graphical user interface. Classes with a name beginning with a **Q** are

standard Qt widget classes, for these, refer to the Qt documentation [72].

The **VisualizeStage::execute()** function sends hyperspectral data to the **VisualizeStage::imageStream()** function, which goes through the input data array, and assuming BIL interleave, writes the band values to band arrays. These are sent to the Qt widget **ImageViewer**. The class **ImageViewer** displays the input data as a **QImage** within a scrollbar area. The **QWidget::paintEvent()** function is reimplemented to zoom the image when the widget is resized. The **eventFilter()** function is installed as an event filter for the **QLabel** containing the displayed image. This catches mouse events. The mouse cursor position is translated into image pixel coordinates and displayed in a **QStatusBar** below, while mouse clicks trigger the **mouseClicked()** signal.

One of the **ImageViewer** instances across all **VisualizeStage** instances is chosen as the "main" image viewer, usually the **ImageViewer** displaying the RGB image reconstructed from the hyperspectral data. The **mouseClicked()** signals from all the possible image viewers are connected to the **forwardSimulation()** slot of this instance, which has, as opposed to the others, kept all the previous hyperspectral data saved. This tells the **GPU-DM** instance to reinvert the chosen hyperspectral line of data and plot the chosen pixel to a pdf file using gnuplot [31]. Keeping track of all previous hyperspectral data is not something which belongs in a pure image viewing class and the behavior is not optimal. Possibly should a new **QObject** derived class be created to receive all hyperspectral data and signals triggered by mouse click events.

The class **VisualizeStage** was developed to handle both full RGB images and chromophore maps. This is switched by a flag in the constructor, along with some vectors defining which bands to display within the chromophore arrays (i.e. which chromophores) and the maximum and minimum values which should be used to normalize the incoming values to unity. The RGB option can be used to display three different chromophores in the same image by assigning a different color to each. Monochromatic intensity values are transformed to RGB colors linearly. 0 to 0.5 is mapped to blue, 0.25 to 0.75 is mapped to green, 0.5 to 1.0 is mapped to red. Applications like gnuplot have been used in this thesis to display chromophore maps, not the visualization stage.

Some subclasses derived from the **VisualizeStage** class have been defined, **RGBVisualize** and **FourTypeVisualize**. The class **RGBVisualize** is used for showing the RGB image derived from raw hyperspectral data by setting the RGB value to be the square root of the corresponding band values, but this is by large obsolete as **Visualize** can show a good enough RGB image by using unity normalization on data output from the calibration stage. The **FourTypeVisualize** stage is used to assign an unique color to 0,1,2,3 from a given array. This is used to visualize the melanin types.

The basic logic in the **VisualizeStage::imageStream()** function is adapted from code shared by Norsk Elektro Optikk, as is parts of the **QImage** conversion in the **ImageViewer** class.

CalibrationStage

The class **CalibrationStage** is used to detect the reflectance standard present in the scene and calibrate the incoming hyperspectral data. Hyperspectral data is first collected in a buffer for a certain amount of lines, with the assumption that the calibration slab should be located within the area enclosed by the lines. Choosing some bands, are the images contained within each band segmented according to some chosen threshold values. The resulting binary images should in principle contain the calibration disc and the sheet containing the light source variation for calculation of both.

There are, however, some changes which will be made to the setup in the future. The images which were available used a paper sheet for light source variation calibration and a calibration disc for the flat field calibration, but these two will in the future be exchanged for a reflectance standard which is wide enough to encompass the whole field of view so that calibration and light source variation calibration is applied at the same time. Much effort was therefore not made to get the segmentation 100% correct, only something rudimentary enough to locate the positions of the calibration elements well enough with some tweaking of the threshold values.

A noise removal stage was not implemented. The images had to be de-noised before they could be

analyzed. This calibration stage was therefore used only for calibration of raw data before writing the calibrated, noisy data to file. These files were de-noised using other processing tools like ENVI, and the calibration stage was dropped altogether in the later processing of the de-noised data.

In the future may the calibration slab be detected as a square object using the Hough transform. The rudimentary methods for thresholding the image are discussed later.

The skin parts of the image are in addition segmented from the rest in the **CalibrationStage::skinThresh()** function. This is needed to clean away the spectra not being skin so that the MNF transform will not reconstruct the skin parts of the image from non-skin parts. It is difficult to segment the skin from the rest of the image purely based on the thresholding of some bands. This can, to a certain extent, be done by knowing that the skin will absorb the most of the light at the lower wavelengths due to the high melanin and blood absorption, but the light source used has a low signal to noise ratio here and areas close to the skin might lie partially in shadow. Misinterpretation will occur.

The pixels in the middle parts of the line was instead meaned to yield a spectrum theoretically representative of the whole line. Equation (2.46) was after this applied for each pixel using the obtained spectrum. The resulting binary image was used to either include or exclude pixels in the image output from the calibration stage.

The calibration itself is applied by dividing each radiance spectrum by the radiance spectrum obtained from the reflectance standard, divided by an appropriate factor defined by the reflectance standard.

ChangeSampleSizeStage

GPU-DM will assume that all images have a sample size that is a multiple of the block size. Technically, GPU-DM could take care of the up-sampling itself since it is presently the only processing stage calling any CUDA functions, but this would require some CPU-based memory shuffling before the CUDA kernels are called. For a processing stage consisting mainly of CUDA functions, it is desired that the sequential, CPU-based code takes as little time as possible and hinder the CUDA calling as little as possible. This was therefore moved to a different processing stage. More processing stages may be using CUDA in the future, which would require that these processing stages also receive appropriately up-sampled images.

The hyperspectral image array is moved into an array containing a sample size being a multiple of the block size. The parts of the memory now not containing any data is left uninitialized. In the real-time production scenario, the sample size will always be appropriate, this was mainly implemented to processed hyperspectral images saved on the hard drive.

PrintToFile

The **PrintToFile** stage is for saving the output from GPU-DM to disk. The chromophore values are saved to a plain ASCII file in a matrix format only for the convenience of this master thesis. There is no buffering of the data before writing it to the hard drive.

3.5.5 CPU-DM

An implementation of the algorithms for the host was implemented early on, for testing the algorithms on single spectra. The algorithms are more or less the exact same as for GPU-DM. This is used throughout this thesis for single spectra fits not extracted from GPU-DM. It also has a Qt GUI for fitting the diffuse reflectance manually. CPU-DM's automatic two-layered inversion facilities were used to invert a huge collection of diffuse reflectance spectra obtained from a psoriasis trial. These results will not be presented here, as the hyperspectral images presented served as a similar performance demonstrator for the inversion methods.

3.5.6 Libraries and compilation

Libraries in use are

- GSL [55]
- Qt [72]
- CUDA [2]
- FFI's hyperspectral streaming framework

The hyperspectral inversion framework is compiled through Qt's qmake system by employing some hacks to make the CUDA parts compile correctly against NVIDIA's **nvcc** utility and the other parts against **gcc**. There are some incompatibility issues between Qt and CUDA that warranted an "abstraction layer" between CUDA and Qt: A different function is called to hook the CUDA processing stages to the rest of the stages by inputting the input ports as input arguments and outputting the output port as the output argument. Signals from Qt are piped through functions calls with no Qt awareness. It is likely possible to mix Qt and CUDA in a better way.

3.6 Noise removal

Noise removal was applied using the MNF transform [36], with the noise being estimated by comparing neighboring pixel values. For the most of the hyperspectral images in this thesis, ENVI [24] was used for the forward and inverse MNF transform. The 12 first bands were selected for the inverse MNF transform as bands beyond that was seen to contain too much noise. Only a subset of the image was chosen for the MNF transform, if the image was not masked. Noise was estimated using the entire region of interest.

Chapter 4

Results and discussion

A fast skin inverse modelling routine is necessary for the derivation of skin parameters to be viable for real-time diagnostic instruments based on hyperspectral imaging. The results of the proposed method, timing results through verification on both real data and simulations, will here be presented.

4.1 General fitting results and choice of inverse model

Before the main results are presented, some intermediate results with regards to fitting will be presented.

4.1.1 Choice of melanin inverse model

The chosen melanin method is to fit dermis before fitting epidermis, and do this for two iterations. Here the results leading up to this choice will be presented.

Indices

The fact that any method involving the iteration with respect to the melanin index or slope of the reflectance spectrum will surely fail has extensively been investigated in [10]. One thing is deoxy crosstalk, another is the lack of one-to-one correspondence. An example of a potential mis-estimation is shown in figure 4.1. The two- and three-layered isotropic diffusion model has here been fitted to the measured spectrum manually. The measurement has an erythema index of 35.76 and a melanin index of 7.78. The "perfect fit" has an erythema index of 38.87 and a melanin index of 7.44. The less perfect fit has an erythema index of 37.37 and a melanin index of 7.83. The melanin- and erythema indices match for the both cases, but only one of the fitted spectra has an acceptable melanin absorption leading to fittable parameters. Even here is it slightly too large. In particular, the two-layered model will not be able to output the correct melanin using this method. The three-layered model can potentially output the correct parameters using the method, but seeing as the two-layered model is a possible solution for the three-layered model when the parameters are set equal, there will be multiple solutions. The fact that the indices match for the "perfect fit" is also no surprise as the spectra are the same in the relevant wavelength ranges. Iterating with respect to the indices will therefore at best be unpredictable.

It will later be seen (and has previously been stated) that the properties at the different layers will influence different parts of the spectrum. Stamatas and Kollias' method uses the linear fit of the reflectance spectrum to quantify the melanin, and corrects this melanin using a blood quantificator gained from the shorter wavelengths. Seeing as the properties will be gained from two different penetration depths, this should be difficult when the properties down to each penetration depth are not the same. Kollias' methods were also seen to not fare better in [10].

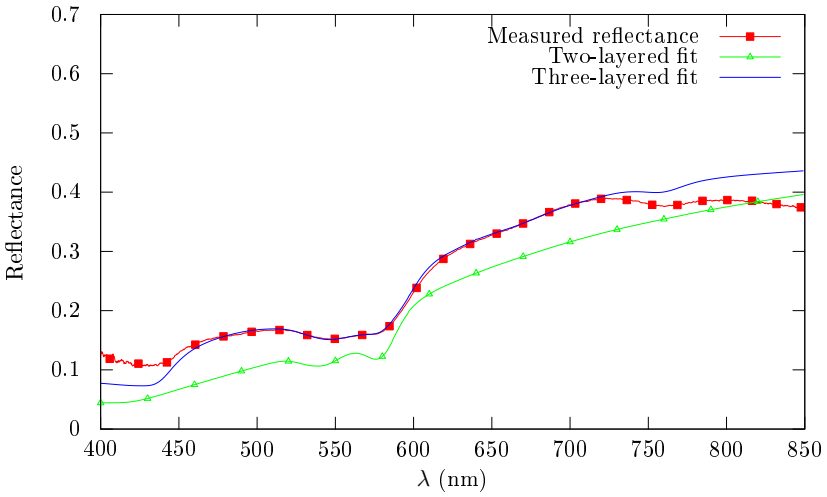


Figure 4.1: The inversion using melanin and erythema indices of a measured spectrum obtained from the back of the hand of an individual with Fitzpatrick skin type IV. The three-layered fit was simulated using $\text{oxy}_1 = 0.2$, $\text{oxy}_2 = 0.55$, $\text{bvf}_1 = 0.025$, $\text{bvf}_2 = 0.06$, $\mu_{a,m,694} = 700$ m^{-1} and the thickness of the upper dermis layer set to 390 μm. The two-layered fit was simulated using oxy $= 0.95$, bvf $= 0.02$ and $\mu_{a,m,694} = 1350$ m^{-1}. Both simulations were done using the isotropic diffusion model.

The index methods may therefore safely be laid aside for the advantage of other methods.

One-layered method

Just as the two-layered model may fit subranges of the spectrum, a one-layered model may also achieve the same. This method has the advantage of using a mathematically far simpler model. The melanin may be decoupled from the blood effects by inverting μ_a and performing a spectral unmixing. Instead of being contained in a thin slice at the top of a skin model the melanin will now be smeared across a semi-infinite layer and thus be severely underestimated. This method cannot be used for much else than providing a first, low estimate for the melanin absorption, if used at all.

Epidermis method

The next step is to fit only epidermis in a similar way to the one-layered method. This will ensure a decoupling of the blood from melanin in a similar way, although with the resulting melanin having the correct order of magnitude. The properties are now not fitted inside a semi-infinite layer, but in the thin slice at the top of the skin model. The problem with this method is that *all* the properties are fitted inside a thin slice at the top of the skin model, calling for a highly unphysical situation. It will be seen that the situation will be unphysical even when the fitting is done at the longer wavelengths where the absorption of the other chromophores is low.

In figure 4.2, the correct melanin absorption is subtracted from the derived, epidermal absorption and compared to a blood composition matching what should be present.

The required blood absorption when placed in epidermis instead of dermis is off, slightly sloped in the opposite direction of the slope of melanin. The method should in theory therefore consistently underestimate the melanin, as is seen in figure 4.3, since the melanin will be forced to be lowered in order to fit the blood absorption coefficients. However, for real data, the melanin would be overestimated using this method. This would happen in the presence of a different scattering than the assumed scattering, but also with no obvious telltale signs of anything being wrong. It did in particular fail for spectra obtained from psoriasis-afflicted skin. The method was therefore abandoned due to instability and uncertainty. The diffusion model will also be less than valid when this much absorption is placed within a thin slice.

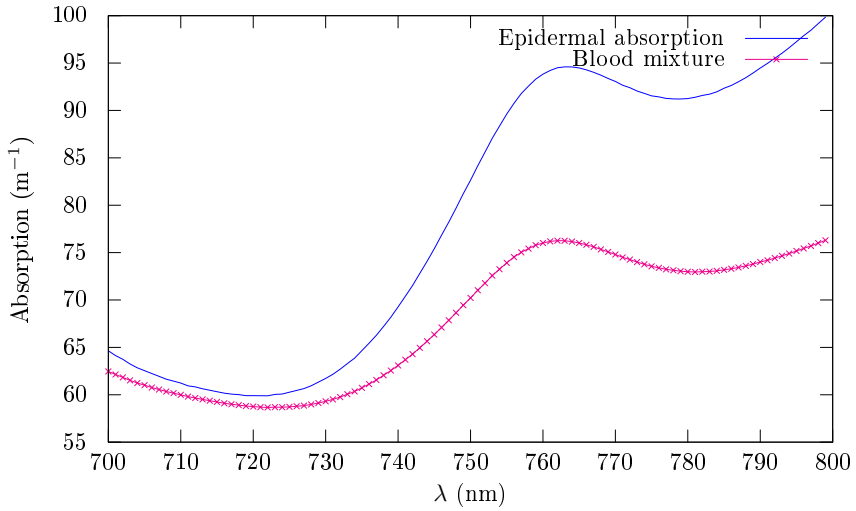


Figure 4.2: Resulting absorption spectrum when all absorption is fitted to epidermis. The epidermal absorption coefficient is fitted to a simulated diffuse reflectance spectrum from the isotropic two-layered diffusion model, oxy = 0.8, bvf = 0.04 and $\mu_{a,m,694}$ = 300 m^{-1}. The dermal absorption was set to 25 m^{-1} in the inverse model. The melanin absorption has been subtracted from the derived epidermal absorption coefficient, and the blood mixture represents a typical blood absorption.

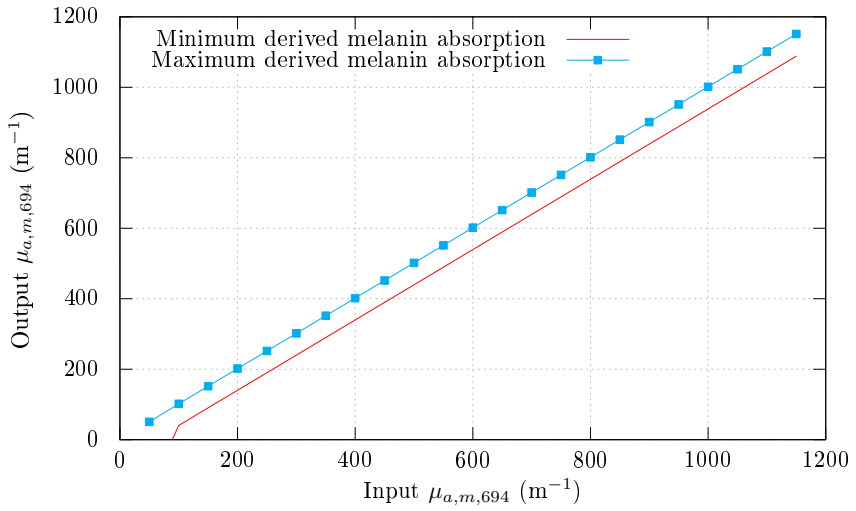


Figure 4.3: Melanin output results using the epidermal fitting method. All chromophores are fitted to the derived epidermal absorption. The forward model to be inverted was simulated using the three-layered isotropic diffusion model by varying the oxygenation at the first and second dermis layer between 0.40 and 0.80, the blood volume fractions between 0.01 and 0.06 and the thickness of the first dermal layer between 200 and 500 μm. Fitted at 700-800 nm. The dermal absorption was set to 25 m^{-1}. The maximum and minimum is plotted to show the stability of the inverse approach for a set of different blood parameters for the same input melanin.

Apparently, these results will seem to be better than the later presented dermal method results, but this will partially be due to the varied parameters being far less extreme in figure 4.3.

Dermis method

The next ad-hoc upgrade of the method must

1. Decouple the melanin and blood using spectral characteristics where deoxy crosstalk is avoided (solved by the one-layered approach)
2. Place the melanin in a thin slice at the top to ensure a valid order of magnitude (solved by the epidermal method)
3. Place other chromophores in a semi-infinite layer to ensure a valid order of magnitude

The next proposal is to use the two-layered model and fit melanin and other to chromophores to dermis before moving the melanin to epidermis. This will theoretically ensure that each individual chromophore will be placed in the correct layer and have the correct order of magnitude during the fitting.

The epidermis method will, as indicated above, have problems due to the strong presence of other chromophores in the wrong layer. The dermis method will, on the other hand, not experience a strong presence of the dermal chromophores in the wrong layer but initially instead experience the missing presence of melanin in the upper layer. Melanin is essential for the situation to become physical. Melanin will block light from reaching down into the layers and will cause less light to be scattered and absorbed down in the lower layers. With the melanin initially set to a low value, the dermal absorption coefficients will represent a less physical situation in which absorption coefficients of the different chromophores may not be fitted in a meaningful way. Melanin is fitted along with the rest and then removed before the epidermis is fitted, but the melanin may be mistaken as something else and thus lead to a too-high estimate of the melanin for which the algorithm easily may not escape.

There are mainly two concerns:

- Too much light is let through epidermis, causing the absorption curves to be affected by a higher scattering
- The melanin curve is corrupted by being placed in the wrong layer and a simple least squares approach is no longer viable

These concerns were mitigated by performing two iterations of the method, using the obtained, increased melanin estimate in the next iteration. The problem is to correct for the erroneous chromophore curves in dermis, likely to not be as bad as when placed in the wrong layer, and correcting for the corrupted melanin curve. The problem is reduced to correcting for the corrupted melanin curve, given that the other chromophore curves are not too corrupted by the scattering errors. This is an easier problem than correcting for all the chromophores placed wrongly in epidermis. Using this method, only one chromophore has to be corrected, and it may be that the corruptions of the other chromophores can be moved into an accumulated corruption of the melanin curve.

Figures 4.4 and 4.5 show the derived dermal absorption for simulated data when less melanin is assumed in epidermis and the correct blood volume fraction is assumed for scattering. The derived absorption is higher than the actual absorption input into the model. This is expected since the absorption in epidermis is set lower. The difference, on the other hand, is not the simple melanin curve (refer to figure 2.5) but something that seems to be convolved with the blood absorption curves. The bump reminiscent of deoxy hemoglobin is present in what should be the melanin contribution to the absorption spectrum (refer to figure 2.2). It is also seen that dividing the difference by the wavelength-dependency of the melanin curve results in a wavelength-dependent melanin coefficient. This is strongly in favour of the melanin curve being convolved with the blood absorption curves when placed in the wrong layer.

It is likely that the blood will be over-compensated in the first iteration because of the presence of deoxy hemoglobin in the absorption curve, and the melanin likewise under-compensated when placed in the top layer. The resulting wavelength-dependencies of the melanin coefficient is shown in figure 4.6 when

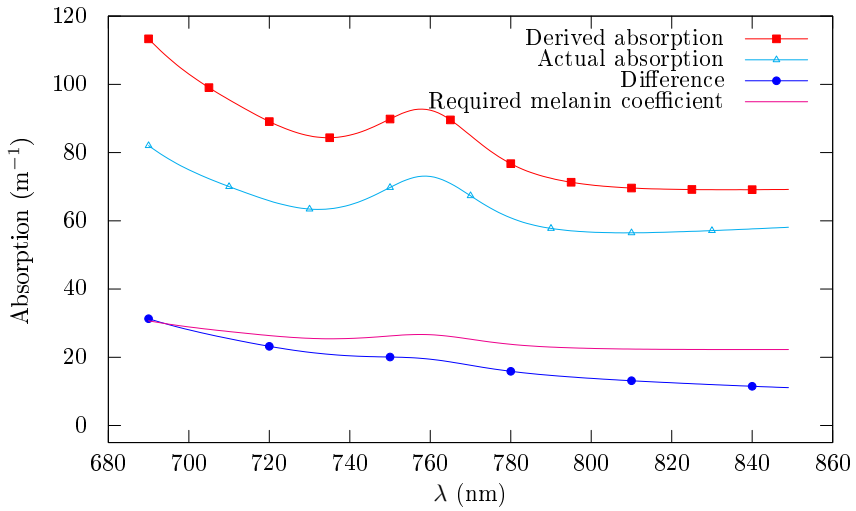


Figure 4.4: Derived dermal absorption when initial melanin content is low, low oxygenation. Input reflectance spectrum was simulated using the isotropic diffusion model from the two-layered skin model using oxy = 0.4, bvf = 0.10, $\mu_{a,m,694} = 300$ m^{-1}. The melanin coefficient $\mu_{a,m,694}$ is initially assumed to be 100 m^{-1} in the epidermal layer of the inversion model.

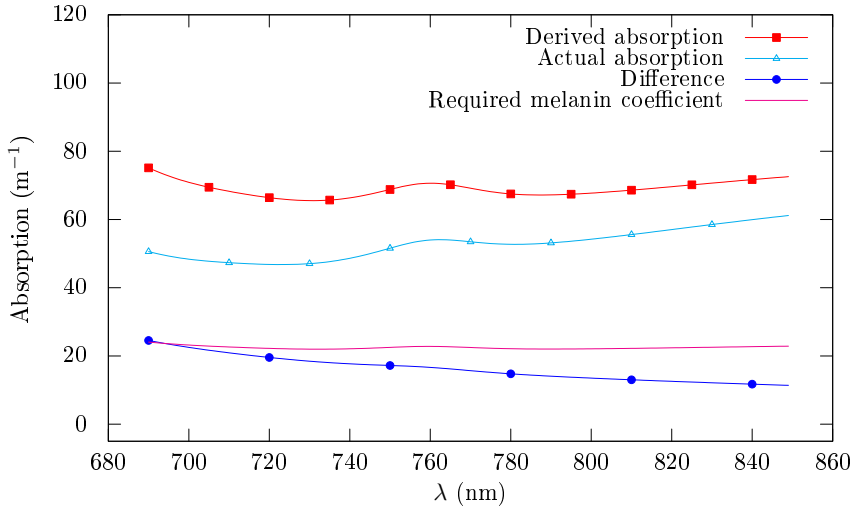


Figure 4.5: Derived dermal absorption when initial melanin content is low, high oxygenation. Input reflectance spectrum was simulated using the isotropic diffusion model from the two-layered skin model using oxy = 0.8, bvf = 0.10, $\mu_{a,m,694} = 300$ m^{-1}. The melanin coefficient $\mu_{a,m,694}$ is initially assumed to be 100 m^{-1}.

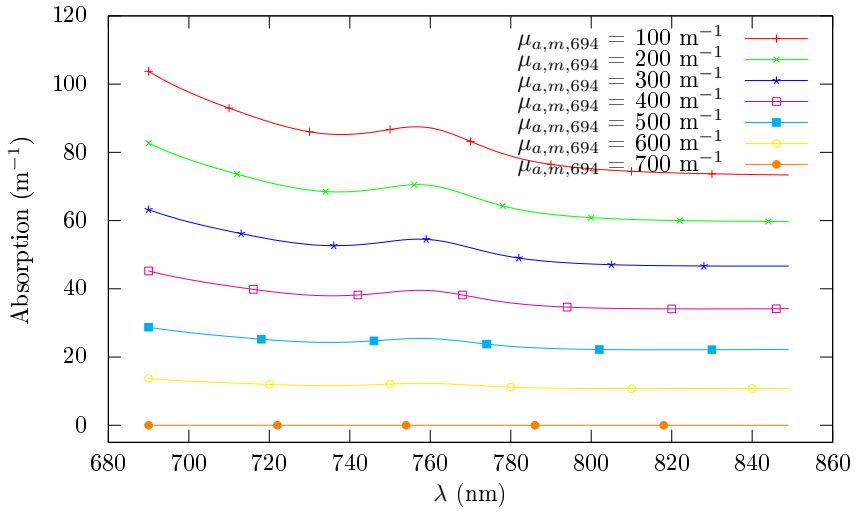


Figure 4.6: The wavelength-dependency of the required melanin coefficient when parts of the melanin is placed in the dermis instead of the epidermis for different melanin coefficients for epidermis. Simulated using an input reflectance derived from the isotropic diffusion model, oxy = 0.5, bvf = 0.10, $\mu_{a,m,694} = 700$ m^{-1}.

the actual absorption is subtracted from the derived absorption, after the epidermal absorption is set to be less than the actual epidermal absorption.

The required melanin coefficient in dermis displays less similarities to the hemoglobin absorption curves as the epidermal melanin absorption approaches the actual value. The fit should therefore become far better after the first iteration of the method.

The melanin absorption curve placed in the wrong layer will mostly, at this wavelength interval, be affected by the deoxy and oxy hemoglobin absorption. Each will influence the melanin to be underestimated: the oxy absorption will influence it to be lower, as the oxy hemoglobin absorption curve slopes in the opposite direction of the melanin absorption curve, while the deoxy absorption curve technically can influence the melanin to be higher than it actually is. However, the deoxy absorption curve has a distinct peak at 760 nm which should influence the fitting algorithm to compensate with deoxy hemoglobin rather than melanin, although they both slope in the same direction.

In the fitting, the erroneous dermal absorption is assumed to follow a linear scheme, $\mu_{a,d}(\lambda) = \mu_{a,oxy}(\lambda)f_{oxy} + \mu_{a,deoxy}(\lambda)f_{deoxy} + \mu_{a,mel}(\lambda)f_{mel}$, but it is clear that $\mu_{a,mel}(\lambda)$ must be exchanged for a convolution $\mu_{a,mel}(\lambda) * h(\lambda)$ for it to be strictly correct. The function for which the melanin has to be convolved with is not known, although the simulations above indicate that it should be some combination of the deoxy and oxy absorption curves. The simplest solution will be to multiply the absorption curves against each other to generate two "new" chromophores to be fitted instead of the pure melanin curve.

This did not work, the dermal absorption was fitted as 100% melanin, and simple unmixing was therefore used.

For the choice of wavelength ranges, it is clear that 730 to 830 nm is the best choice. This is among the only wavelength ranges with no obvious crosstalk potential between the parameters, as seen in figure 2.2. The wavelength range above 600 nm is better than the shorter wavelengths since the absorption of all other chromophores except for melanin is uniform and low enough for the presence of melanin to be noted. It will be seen that 730 and 830 is problematic due to an erroneous scattering assumption or other errors. The other choice, 620-700 nm is documented to have too severe crosstalk between melanin and deoxy hemoglobin and is no better. Given that the error someday is resolved, 730-830 nm will also represent the range for which the diffusion model is the most valid due to the low absorption. On the other hand does this wavelength range represent a high penetration depth. The two-layered model will have difficulty fitting the parameters here when the properties at the different layers the light penetrates are extremely different. The range 530-590 nm has a very distinct spectral difference

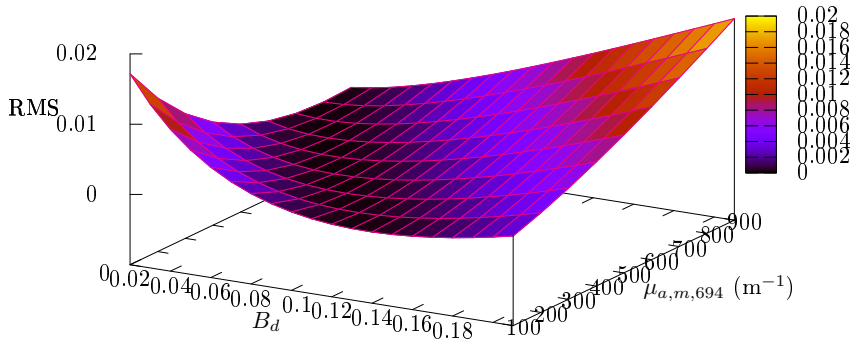


Figure 4.7: The root mean square error between the measured spectrum and simulated two-layered, isotropic reflectance spectrum as a function of B_d and $\mu_{a,m,694}$ for the spectrum shown in figure 4.1. Oxygen saturation is set to 60%.

between the blood absorption spectra and the melanin absorption in addition to a lower penetration depth, but methemoglobin is on the other hand too similar to melanin for those cases where there is a methemoglobin presence. 450-530 nm does seemingly have spectral features for all chromophores being distinct enough for melanin decoupling, but with betacarotene and bilirubin being present and the fact that the melanin presence in dermis will be convolved by the other chromophores is it uncertain how much better this wavelength interval would fare. The diffusion model will also be less valid here.

The wavelength 730 nm is chosen as the lower wavelength as an even lower wavelength would give more emphasis on the parts of the deoxy absorption curve being similar to the melanin absorption curve instead of the deoxy maximum. Crosstalk resulting in melanin overestimation was seen for a slightly lower start wavelength, eliminated when the deoxy maximum was given higher emphasis. The wavelength 830 nm is chosen as the upper wavelength to keep the interval large enough. For this wavelength range is the water absorption also constant enough to be included in a freely varying constant.

The performance of the chosen melanin method will be evaluated later in this thesis.

Summed up, this method will still have deoxy hemoglobin crosstalk, although in the opposite direction. Melanin will be underestimated instead of being overestimated. This should be easier to control as such cases can be detected by the prominent deoxy hemoglobin absorption maximum.

Multidimensional fitting

With the ad-hoc methods above and their obvious and noted problems, the question remains why one does not use some multidimensional fitting scheme to fit both blood and melanin at the same time in their respective layers.

In figure 4.7 is the root mean square error between the simulated and a measured reflectance plotted as a function of the blood volume fraction in dermis and melanin absorption in epidermis. The oxygenation is not varied, which it would in a multidimensional fitting setting, making the above plot slightly misleading. On the other hand is the chosen oxygenation likely close to the value it must attain. Seemingly are there multiple solution in a sadle point, but minute differences along the sadle point will make a fitting algorithm converge to around 500 m^{-1} for the melanin absorption, provided the error surface as a function of the oxygen saturation is well-behaved. Judging from this is such a multidimensional fitting entirely possible and even "easy" as there are many extensively researched solutions to the multidimensional fitting problem and even entire fitting libraries. For instance may Levenberg-Marquardt optimization [71], conjugate gradient methods [71], simplex methods [71], simulated annealing [71], genetic algorithms [71, 112] or similar be used. There is no question that such methods will be able to find good results, especially if box constraints can be implemented. Such methods may fare less well when implemented for the three-layered model since there are too many parameters to vary, and the root mean square error is not a good enough objective function. However, for a smaller interval and using the two-layered model, the best fit for the melanin can be found.

The main reason is the implementability in CUDA and real-time requirements. Most methods with fast convergence will require the calculation of the derivative with respect to all parameters. For every iteration and for every wavelength, the derivative of the diffuse reflectance with respect to both the scattering in dermis, the absorption in epidermis and the absorption in dermis must be calculated, in addition to the reflectance itself. It is known that these operations require much register space, warranting that each operation is implemented as a global kernel. Data must therefore be read in and out of global arrays several times for each iteration. The iterations must be parallelized across pixels, since the iteration will be done with respect to the accumulated error. This will increase either the number of derivatives that must be calculated within each kernel or increase the global memory reads. The number of iterations must hence be initialized from the host. The number of iterations required for a good estimate will depend on the starting point. Some of the methods will also require line searches for each iteration, which will be done differently for each pixel. The most of the gradient based methods will require large matrix multiplications or even matrix inverses [71], though matrix inverses can be approximated using other methods [71].

In short, multidimensional fitting will be difficult to implement for this particular problem, and therefore has ad-hoc methods been used. If something is wrong or something is under- or overestimated may this also be detected midway through the proposed methods by evaluating the dermal and epidermal fits. This would not be as possible for a multidimensional fitting. Though this has not been used for the chosen method as it stands, the method can potentially be modified to use spatial information in addition to the spectral information in the individual, intermediate spectral unmixings using common methods present in the hyperspectral community. This is far more difficult to achieve for the multidimensional fitting.

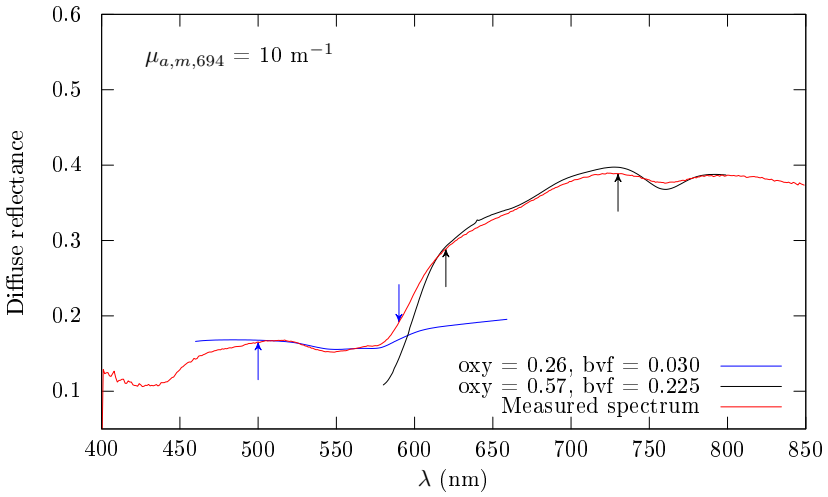
The ad-hoc methods are chosen not because they necessarily are better methods giving better answers, but rather because of performance issues. Tests were also run using the three-layered model in a fitting algorithm using the root mean square error as an objective function (not shown here), and while the spectra seemingly matched, there were tell-tale signs that some parameters either were under- or overestimated.

4.1.2 Effects of too high/too low melanin

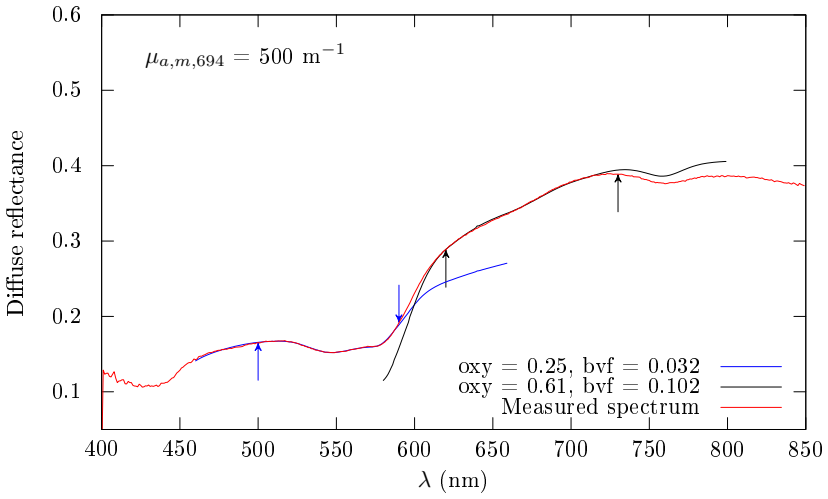
In figure 4.8 are two-layered fits for the spectrum seen in figure 4.1 shown for a too low melanin estimate, the approximately correct melanin estimate and a too high melanin estimate. The overestimation- or underestimation will have a discernible impact on the simulated spectrum. It is most easily spotted for the shorter wavelengths, where the simulated spectrum has to intersect the measured spectrum due to the scaling of the absorption maxima not being fittable. For the longer wavelenghts, the discrepancy cannot be spotted within the fitting range, although the extrapolation towards the even longer wavelengths will show whether the fit is good or not. For the lower melanin estimate is deoxy hemoglobin overcompensated to rectify the missing absorption. For the higher melanin estimate is deoxy similarly underestimated in order to fit the absorption to the reduced dermal absorption, as oxy hemoglobin has a lower absorption than the deoxy absorption. It should be noted that apart from the differences in blood volume fraction are the oxygen saturation values comparable, but it has been seen for other spectrum fits that the oxygenation would attain extremely (close to 1) large values for melanin overestimation, and extremely small (close to 0) for too low melanin underestimation.

4.1.3 Three-layered fitting

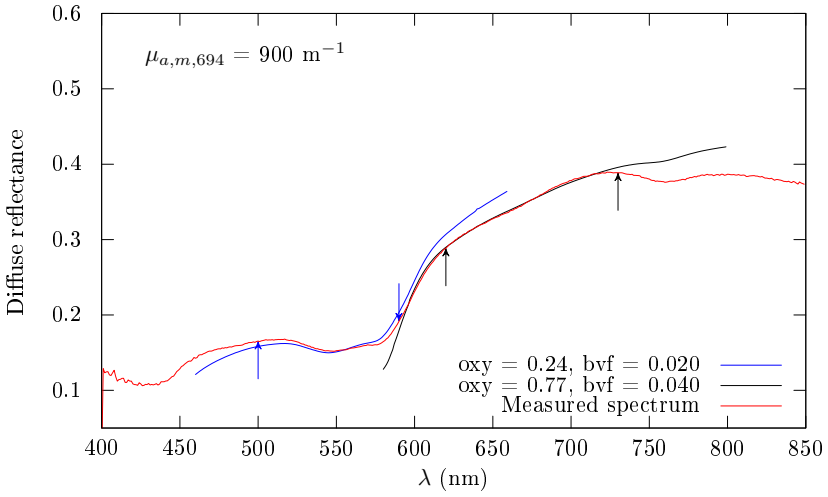
In figure 4.9, the isotropic, three-layered diffusion model is run for different oxygen saturations while keeping all other parameters constant. The different oxygenation values at the different layers are affecting different parts of the spectrum due to the penetration depth in each part, as was expected. It must however be noted that the depth of upper dermis is set exceedingly large for demonstration purposes. The oxygen saturation values are also extreme. Such a clear divide between the parts of the spectrum representing different penetration depths will not be seen for other spectra with a more shallow depth of the upper dermis.



(a) Too low melanin assumed.



(b) Approximately the correct melanin.



(c) Too high melanin assumed.

Figure 4.8: The effect of having a wrong melanin estimate on the rest of the two-layered reflectance fitting. The input spectrum is the same as the one presented in 4.1, Fitzpatrick skin type IV.

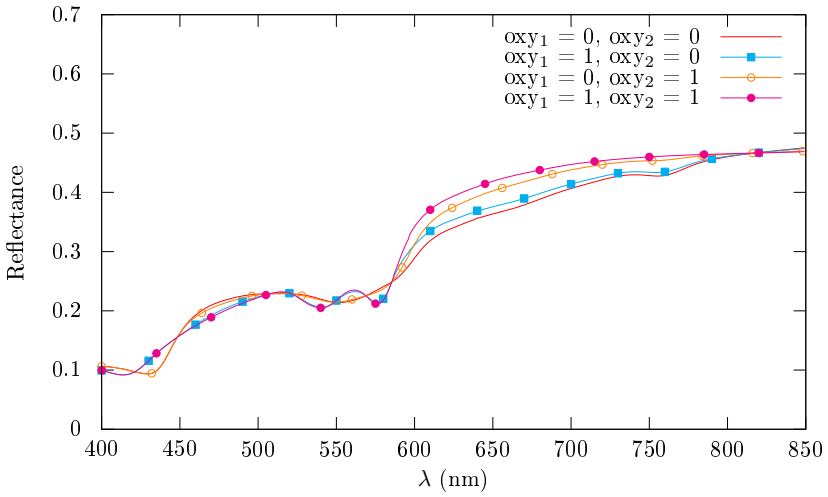


Figure 4.9: Simulations using the three-layered model, $\mu_{a,m,694} = 550$ m^{-1}, bvf$_1 = 0.01$, bvf$_2 = 0.05$ and $d_2 = 840$ μm.

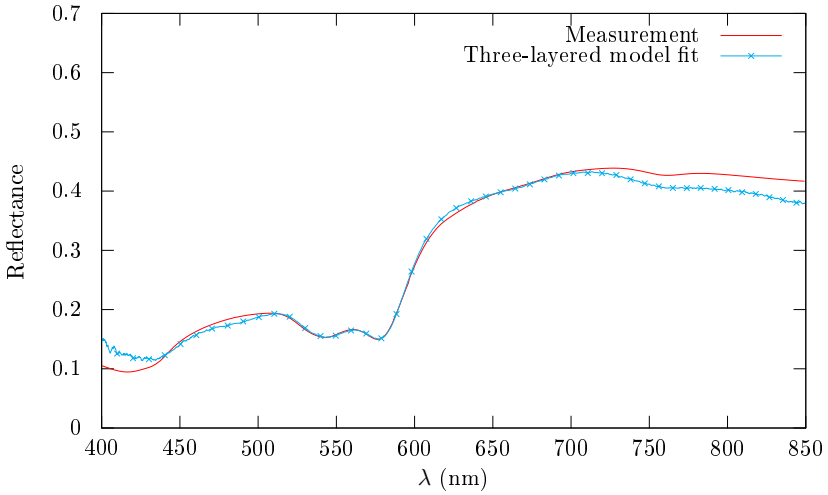


Figure 4.10: Manual isotropic diffuse reflectance model fit to a spectrum from the palm of the hand of an individual with Fitzpatrick skin type IV. Parameters are $\mu_{a,m,694} = 350$ m^{-1}, oxy$_1 = 0.71$, oxy$_2 = 0.83$, bvf$_1 = 0.035$, bvf$_2 = 0.13$, $d_2 = 230$ μm.

A typical three-layered model fit is shown in figure 4.10. It must be stressed that this is fitted manually. The fit is not the objectively best fit possible as the reflectance curves intersect at 650 nm, but it can be used for demonstration purposes. It is possible to get a good fit at the shorter wavelengths, but at the longer wavelengths is the fit far worse. Neither is it possible the bridge the remaining gap. This is a paradox as the diffusion model is more accurate at the longer wavelengths. Water absorption is not included, but had it been included would the fit have become even worse as the wavelength-dependency of water absorption does not match the remaining absorption exactly. This can be due to the water absorption being very low at this wavelength range and the exact spectral characteristics being difficult to measure, but as the water absorption is low should it not have much impact on the reflectance spectrum.

A more likely explanation is the scattering. The scattering model is the one presented by Saidi et al. [82]. This scattering function was mainly found by fitting an experimentally found scattering function to a model from around 400 nm to 700 nm - exactly in the area where the fit is optimal. Above 700 nm, the fit becomes worse. In addition, the Saidi scattering follows a parabolic function, but several sources [116, 83, 58, 103, 8] will report the reduced scattering coefficient to be monotonically decreasing with increasing wavelength from 400 to 1400 nm. Saidi's reduced scattering coefficient will, too, exhibit this behavior for the chosen Rayleigh and Mie coefficients, but above a certain wavelength this is no longer valid. When converting to unreduced scattering coefficients, they will also no longer be monotonically decreasing. There is less literature available with regards to whether or not the non-reduced scattering coefficient should be monotonically decreasing.

Similar behavior will be seen when using the two-layered model to fit the measured spectrum across this entire wavelength range. Is the wavelength range split in two parts are the fits apparently better, but this will not necessarily be due to the situation being more physical. The point being, the success of the two-layered inverse model cannot be evaluated only by comparing the fit against the measurement for the specific wavelength range, but it should also be extrapolated to the neighboring wavelength ranges to see how well the model fares outside of the fitting range. Anything can be fitted to a short wavelength range.

4.1.4 Wavelength ranges for two-layered fitting

The chosen wavelength ranges for the two-layered absorption fitting were 500-590 nm and 620-730 nm, alternatively 690-820 nm. In addition was 460-530 nm used for comparison with 500-590 nm and for betacarotene- and bilirubin determination.

The wavelength range 620-730 nm was chosen primarily because the wavelengths above this range will have the problems noted above, even though it is a paradox that this wavelength range still is used for the melanin determination due to it being the only relatively crosstalk-free range. The wavelength ranges were elsewise chosen purely based on how well the data was fitted by inspecting several spectra. A plot of the penetration depths for two typical spectra is shown in figure 4.11.

For the interval 500-590 nm, the spanned penetration depth from maximum to minimum penetration depth will respectively be 95.1 μm and 68.9 μm for spectrum 1 and 2 displayed in the above figure. For the interval 620-730 nm for spectrum 1 and 2 respectively, this is 194 and 168 μm. For the interval 690-820 nm, the penetration depth range for spectrum 1 and 2 is respectively 152 and 160 μm. Fitting within a wavelength range across a penetration depth range of 200 μm should generally give a good fit, and can serve as an explanation why it is not possible to fit the entire wavelength range for the longer wavelengths, in addition to apparent scattering problems. This penetration depth range will be far above 200 μm, and apparently not be uniform enough.

Each wavelength range has several fitting problems. 620-730 nm has severe crosstalk with melanin, and when melanin is slightly underestimated, the deoxy hemoglobin will be severely overestimated, as was seen. With one of the methemoglobin peaks being within the wavelength range should methemoglobin have been easily detectable, but it seems to be that methemoglobin even here will overcompensate for something. The shorter wavelength ranges will experience crosstalk between methemoglobin and melanin when melanin is underestimated, and the spectrum fits will apparently be "good" due to the

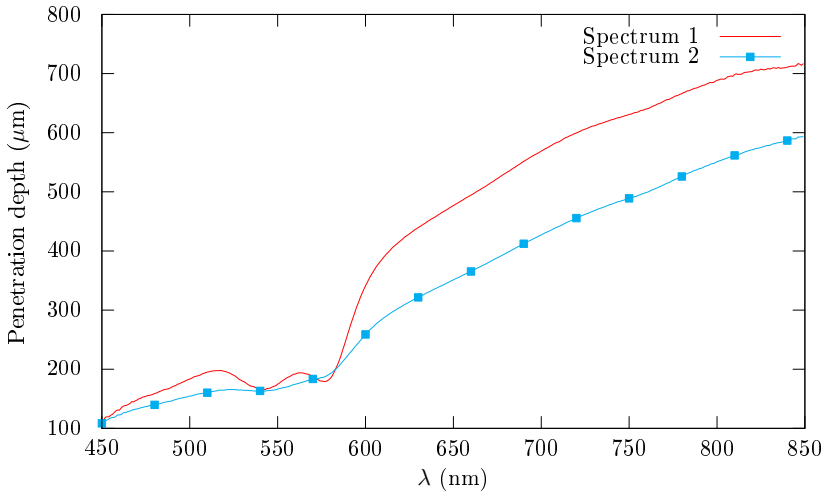


Figure 4.11: The penetration depth as a function of wavelength for the measured spectra shown in figure 4.10 (spectrum 1) and 4.1 (spectrum 2).

mis-compensation of methemoglobin. The constant, otherwise set to 25 m^{-1} in the forward model, was allowed to vary freely, and would vary from wavelength range to wavelength range in a non-consistent manner. This may be a way for the two-layered model to approximate the semi-infinite layer which should have been present beneath the properties down to the penetration depth. For example may betacarotene only be fitted to the spectrum when the constant is set to something else than 25, and betacarotene will generally be have the largest impact while being present in epidermis.

Lastly, the scattering is not allowed to vary, and this *will* introduce errors in the fitting which may be mistaken as something else. How this should be rectified when this method and not a multifitting approach is used is not clear. The derived parameters will be influenced by the erroneous scattering and cannot be used to rectify the scattering. In addition, the refraction index at the different layers will vary spatially and from skin sample to skin sample. In the diffusion model is this assumed to be constant throughout all layers for simplicity in the boundary conditions, and this will introduce fitting errors. This will be investigated in more detail later on.

4.2 Time analysis

Here are the timing results presented.

4.2.1 Optimization

The running times for some CUDA kernel calls for the two GPUs in use are presented in table 4.1. The time reduction is given as the difference between the new and old running times as percentages of the old running time, in order to have a common number of merit for all kernels regardless of original running time. This is not a very good figure of merit, but can be used for comparisons.

As is seen, the running time, not unexpectedly, is lower on the newer graphics card but not by a large amount. The total running time on the older GPU was 25 ms, while the higher-end graphics card has a running time of 7 ms. The age gap between the two graphics cards is about 2 years, and though they are developed for different applications, for such a large upgrade the differences in running time should be far more.

The new graphics card has a larger amount of registers. This is well received by ReflInvMuad and ReflInvMuae as these have a performance boost of around 80% as seen in the table. The limiting

Table 4.1: Comparison of running times for some CUDA kernels across graphics cards.

Operation	Quadro FX 3700M (ms)	GeForce GTX 670 (ms)	Time reduction (%)
Invert $\mu_{a,d}$, 28 bands	0.68	0.10	85
SCA, 28 bands, 4 endmembers	1.10	0.51	54
Invert $\mu_{a,e}$, 28 bands	2.51	0.64	75
Monochrom. unmixing, 28 bands	0.02	0.02	0
Invert $\mu_{a,d}$, 160 bands	3.74	0.50	87
SCA, 31 bands, 5 endmembers	1.52	1.09	28
SCA, 20 bands, 7 endmembers	2.42	1.83	24
SCA, 26 bands, 5 endmembers	1.49	1.08	28

factor for these two were the reuse of computations and temporary saving of variables and the register overhead, which is lessened on the new GPU. There are not many ways to optimize this further as all the computations are needed. The only way to optimize is to rearrange and try to reuse as much of the calculations as possible, but major optimizations have already been put through.

Performance was not found to increase by increasing the number of threads per block. This is due to the amount of registers in use compared to the amount of registers being available still being high.

SCA, however, does not have the same performance boost. The amount of shared memory has not changed much for the newer GPU [2]. All of the arrays are saved in shared memory for faster memory access, but evidently, this strategy is not scalable. Saving the matrix multiplication $S^T S$ to shared memory is reasonable as this is used often and across threads. Saving the fractions to shared memory is dubious. This is used to reduce overhead across iterations, but the global memory lag it is supposed to hide will be exchanged with a shared memory lag that cannot be hidden away. Too much shared memory is used on a per-block basis for the GPU to be able to exchange one block with another to hide away the memory lag. It is possible that the GPU, on the other hand, would be able to reduce global memory lag with a higher thread occupancy when the shared memory strain is reduced. The investigation of this is shown in table 4.2.

Table 4.2: Comparison of running times for different variants of SCA either allocating all arrays in shared memory or using the global memory. 20 bands and 7 endmembers.

Variant	Running time per line (ms)	Theor. occupancy (%)	Occupancy (%)	Shared memory/block (kB)
w/ shared memory	1.83	23.4	11.1	12.9
w/ global memory	4.81	93.8	11.2	0.40
300 threads/block	4.82	93.8	15.6	0.40
800 threads/block	5.11	78.1	38.7	0.40
800 threads/block, 4 lines	1.53	78.1	44.6	0.20

Less use of shared memory increases the occupancy, but the running time increases. This will be due to the global memory being far slower. Are the number of threads per block increased is the running time not readily decreased, but this is due to lower multiprocessor utilization, as the number of blocks will not match the number of multiprocessors. Are multiple lines inverted, in order to have a number of blocks matching the number of multiprocessors, is the running time actually decreased compared to the version using shared memory.

The above results were run for compute capability 1.1. The newer GPU also has support for compute capability up till 3.0. The running times for the different compute capabilities is shown in figure 4.3. The main thing that is the most apparent here is that the thread- and block distribution is no longer optimal, and the running times are higher with increasing compute capability.

Table 4.3: Comparison of running times for SCA (20 bands and 7 endmembers), ReflIsoL2InvMuad (160 bands) and ReflIsoL2InvertMuae (28 bands) for different compute capabilities.

sm	*InvMuad	*InvMuae	SCA
1.1	0.644	0.503	1.827
1.2	0.644	0.503	1.827
1.3	0.644	0.503	1.827
2.0	1.351	3.803	3.191
2.1	1.357	3.809	3.177
3.0	1.611	3.780	3.075

For now has there been no need for features available from higher compute capabilities, and the program has been kept to 1.x regardless that the GPU in use has support for a higher compute capability.

A version of SCA using the registers instead of shared memory was proposed. The result is shown in table 4.4.

Table 4.4: Comparison between SCA and SCAFast, 20 bands and 7 endmembers

Function	Time (ms)	Shmem (kB)	Registers	Occ. (%)	Theor. occ. (%)
SCA	1.824	12.9	18	11.1	23.4
SCAFast	0.321	0.43	63	10.9	46.9

The register usage becomes very large with this change, comparable to the register usage of InvMuae. The performance boost is on the other hand huge, but the maintainability of the code becomes worse.

There are two different extremities present - one of the methods uses excruciatingly much shared memory, while the other method puts a huge strain on the registers. Straining the registers seem to result in a more optimal function than straining the shared memory. This will partially be due to the fact that the registers are far faster than the shared memory.

Some concern may also be directed towards the fact that the thread distribution might not be optimal. The performance boost of the monochromatic unmixing (**MultVector()**) will for example be negligible since the workload in each kernel is low and the most of the computation time will be due to global memory access. This would scale better with hardware if there was a higher threads per block, as was seen in table 4.2.

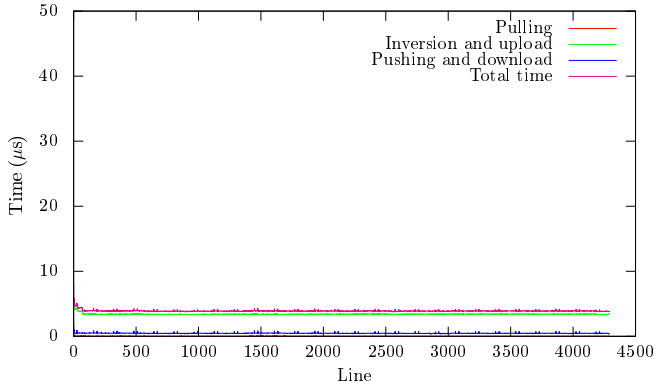
ISRA was found to use 2.5 s for the unmixing of one line at all three intervals, using 1000 iterations.

4.2.2 Real-time analysis

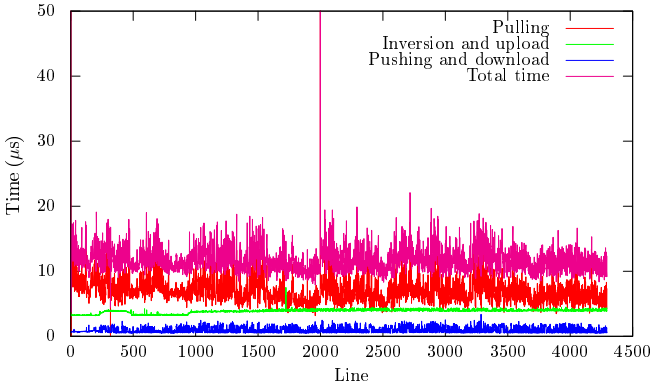
The total running times, after optimization and on the newer GPU is shown in table 4.5.

The total running time is 3.5 ms, leaving 26.5 of GPU time for other use.

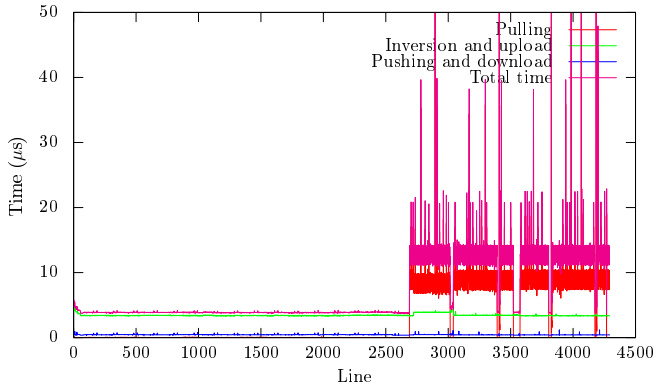
Total timing results as measured from the host are shown in figure 4.12. The inversion by itself is apparently stable, being able to invert the hyperspectral line of data well within 10 ms every time, as seen in figure 4.12a. Posed by some interference from other application does the inversion itself stay constant, but the pull requests for data from the input port becomes unstable, see figure 4.12c. With this many cores is it unlikely that the interference is due to CPU usage, especially as the total inversion time stays constant as if it does not have to wait for kernel launches. It will rather be due to hard drive interference as the setup reads data directly from the hard drive instead of pulling it over the TCP/IP protocol from a different computer. This causes interference and lags in the data streaming from the



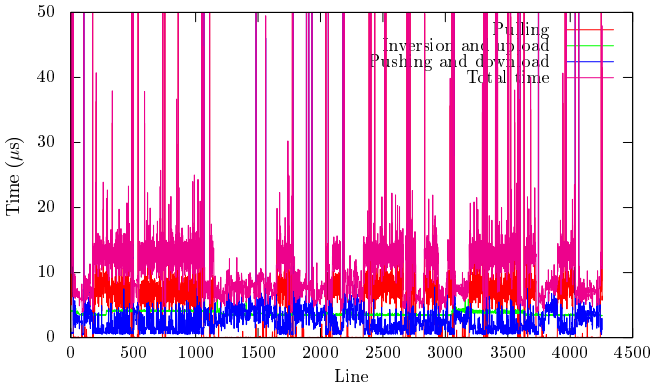
(a) Inversion as the only processing block



(b) Inversion, calibration and full visualization



(c) Inversion as the only processing block, interference



(d) Inversion, full visualization and saving to hard drive

Figure 4.12: Timing results, time difference measured from the start to the end end of the inversion block for each line.

reading block to the inversion block. Memory usage is almost maximized and swap is likely to be used, straining the hard drive even more when posed by interference.

This behavior is also seen in 4.12d. Here, the data is saved to hard drive in addition to being visualized. The data saving is inefficient since it goes through the arrays one by one position as it writes them to separate files. This strains the CPU. There will be a lot of hard drive read/write requests when data thus is written from separate threads all the while being read from the same hard drive. The file writers will have to wait for the hard drive, the hyperspectral reader will have to wait for the hard drive, and all the while must the inversion block wait for data to appear in the buffer of the hyperspectral reading block and wait for the file reader to eat away at the inversion block's output buffer. Some of this will be alleviated when data is streamed from TCP/IP instead of being read from the same hard drive some block also is writing to, but some kind of buffering will still be necessary.

The times are slightly more stable with no file writing in place, seen in figure 4.12b. There is one spike, due to a switch between the accumulated buffer of the calibration block to pulling data from the hyperspectral file reader. Push requests are far smaller than pull requests. This will also be due to the implementation of the calibration block. It runs through some slow for-loops. This is handled by at most one CPU core. The CPU core will lag behind the speed of the inversion block, and pull request times become significant.

The CUDA inversion implementation is, even with uploading and downloading to and from the GPU, well within the deadline. The problems are the other processing blocks as they are unstable and some of them too easily dependable on the response times of the hard drive. The implementations were mainly developed for demonstration purposes and convenience, but before they can be included in a real-time environment, they must be optimized in better ways. Either may some of the nested for loops be exchanged by non-nested for-loops, or the for-loops may be exchanged by other strategies altogether like multi-threading on the CPU using openMP [66] or even using the GPU. CUDA should in principle be reserved for the heavier activities like inversion, but as is seen is the inversion finished well within 4 or 5 ms, and this is one of the most heavy operations possible. Any other operation will be far faster. Most of the unpredictability is also due to hard drive response times, alleviated when data is streamed over the TCP/IP protocol and memory use is more optimized.

A similar implementation on the host, though using Lawson-Hanson for fitting instead of SCA and with other differences enough to not make the situation comparable, uses 172 ms for the inversion of exactly one spectrum. For the inversion of 1600 spectra would the CPU version use about 4 minutes. Even if the implementation somehow was optimized down to 10 ms, it still would use 16 s per hyperspectral line of data.

It should be apparent that this is no real-time system, it is far too unpredictable in its behavior. The framework is partially at fault since it is no real-time system for ensuring real-time behavior, but the implementations are also at fault since they are not warded against hard drive interference. The interference on the CUDA kernel launches is minimal, although it does exist. This is to be expected, as the CUDA kernels are called from a non-real-time host where kernel launches must wait for scheduling even though the GPU is free for computation. There does exist implementations [45, 23, 81] for ensuring proper real-time behavior for CUDA, as such behavior is not inherent in any operating system, but as the framework itself is not real-time will not this help. With an arbitrary number of processing blocks can real-time behavior never be ensured, as the speed of each processing block will be limited by the slowest processing block due to the processing blocks being chained together. Using hard real-time behavior will also be difficult as long as visualization and hard drive writing is performed on the same computer as the computations itself. Hard real-time behavior will make the computer prioritize the processing blocks over trivial matters like graphics processing.

The framework can on the other hand easily be made to forward the data to and from a TCP/IP protocol, and visualization and hard drive saving need not be done on the same computer as the processing computer. Provided that more proper real-time behavior is desired, is this the only way to go other than employing gigantic buffers to alleviate the hard drive stress. Other than this is the processing requirements far within the power of 8 CPU cores, even in its unoptimized form. Of course, even if the processing is moved to a different computer are the hard real-time requirements not fulfilled automatically, but the hard real-time requirements *can* be fulfilled provided that hard real-time functionality is

implemented in the framework and the work described in [45, 23, 81] is used.

While all tasks technically are event-triggered, they will in reality be periodic, which should make them schedulable. All tasks will, however, have the same deadline, as each task is dependent on the former and all will have the same release times. The deadline requirements are with respect to the visualization results, which is dependent on all former stages. The problems described above will mainly be due to input jitter, as a new line of data will not necessarily arrive consistently at every 30 ms interval. This will, as already mentioned, be alleviated when data is delivered directly from the data over TCP/IP. The task at hand is schedulable, but the real-time framework is not able to give any real-time guarantees other than guaranteeing that the slowest processing block will slow down the entire chain. Generally will no operating system guarantee hard real-time behavior in any case. Such operating systems will generally be designed for embedded systems which cannot be allowed to fail, not hard hyperspectral applications like this.

Still, hard real-time is not a requirement. The requirement is a fast computing requirement. The lines will only be buffered if the deadline is missed. The visualized results will still be shown to the medical personnel and experienced as being as fast as the camera is scanning. The behavior seen above is therefore well within the requirements, even if it is hairy and not strictly real-time. The methods in use are deterministic and the total GPU time will stay the same regardless of image difficulty.

Table 4.5: Total running times for the inversion of one hyperspectral line of data

	Function	Time (μs)	Accumulated time (μs))
Calculate melanin coefficient	calcSkinData	20	20
	InvMuad	97	118
	SCAFast	198	316
	RemoveMuam	13	328
	InvMuae	643	972
	MultVector/StraighLine	19	991
	calcSkinData	17	1007
	InvMuad	98	1105
	SCAFast	198	1303
	RemoveMuam	13	1962
	InvMuae	647	1962
	MultVector/StraightLine	17	**1979**
Detect melanin type	setMelaninType	3	1982
	calcSkinData	7	1989
	InvMuad	13	2002
	calcSkinData	6	2008
	InvMuad	12	2020
	CheckReflectanceScaling	4	2024
	calcSkinData	5	2030
	InvMuad	12	2042
	calcSkinData	5	2047
	InvMuad	12	2059
	CheckReflectanceScaling	3	2063
	calcSkinData	5	2068
	InvMuad	12	2080
	calcSkinData	5	2086
	InvMuad	12	2098
Invert $\mu_{a,d}$, unmix	CheckReflectanceScaling	3	**2101**
	calcSkinData	75	2176
	InvMuad	503	2679
	SCAFast	253	2932
	SCAFast	349	3281
	SCAFast	233	3513
	skindataDeinit	3	**3516**

4.3 Numerical accuracy

4.3.1 Singularities

A potential singularity in the denominator of the expression for the isotropic diffusion model function will pose problems when numerical accuracy is low. The singularity lies at $\delta_1 = 3\xi_2$, countered by a numerator which approaches zero when δ_1 approaches $3\xi_1$. This has no repercussions for the calculation of the reflectance or the variation of dermal absorption coefficient with respect to the reflectance as this happens only for one exact value governed by the epidermal absorption coefficient. Analytically, the singularity does not exist since it is countered by the numerator, but numerically, it can pose problems when the epidermal absorption coefficient is varied in the region of the potential singularity. Additionally, as the expression for the analytical derivative is complicated, the numerical errors will accumulate and result in a worse singularity-like phenomenon around the potential singularity if the numerical accuracy is unsatisfactory.

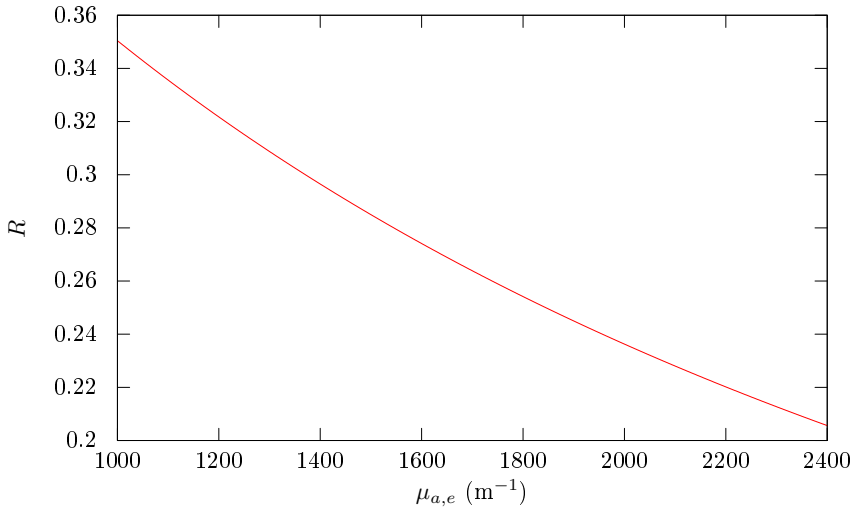


Figure 4.13: The reflectance of the isotropic diffusion model as a function of the epidermal absorption coefficient. 80% oxygenation, 1% blood in dermis.

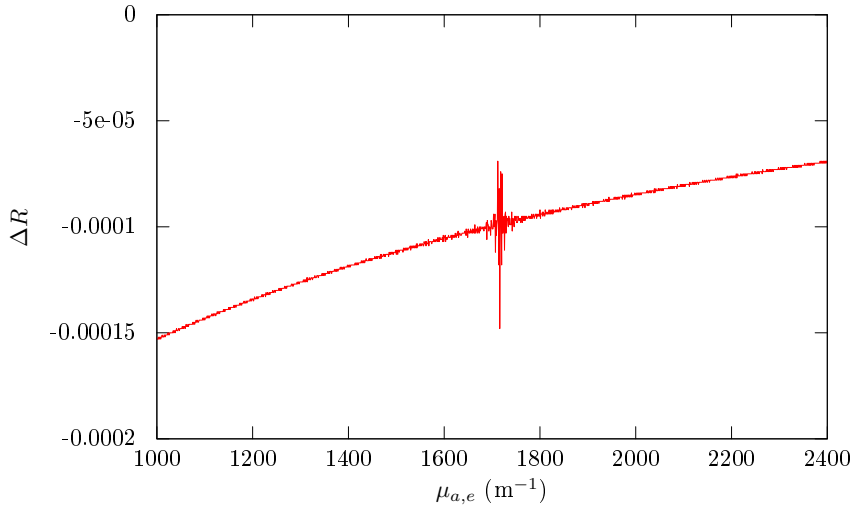


Figure 4.14: The numerical derivative of the reflectance shown in figure 4.13 with respect to the epidermal absorption coefficient.

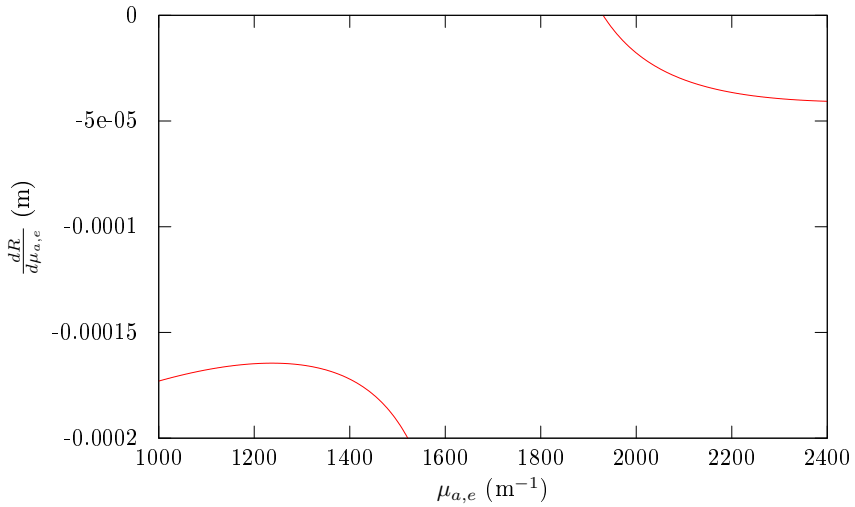


Figure 4.15: The faulty analytical derivative of the isotropic diffusion model as a function of the epidermal absorption coefficient. 80% oxygenation, 1% blood in dermis. When the sequence of the arithmetic operations increases the error.

The reflectance calculation itself is apparently safe, as shown in figure 4.13, although the numerical derivative in figure 4.14 will show that there are some discrepancies around the potential singularity. The analytical derivative does, however, have an obvious singularity at the absorption coefficient in question, as shown in figure 4.15. This is for a naive implementation. A less naive implementation will lessen the singularity, as shown in figure 4.16. The singularity is still present, but it is far lessened.

4.3.2 GPU versus host computing

GPU and host was found to not vary significantly with respect to the numerical accuracy. In figure 4.17, the epidermal and dermal absorption coefficients derived from the same reflectance spectrum and using the same initial parameters are displayed. While there is a difference, this difference is negligible for this application. The inversion is a stress test as any errors will be accumulated, and a sequence of derivative and reflectance calculations are done for a wide array of scattering and absorption coefficients. The numerical accuracy is made possible by the sequence in which the reflectance and derivative calculations are performed. A naive implementation caused far larger numerical inaccuracies for the GPU compared to the CPU for the exact same sequence of arithmetic operations, where calculations particularly for small absorption coefficients were truncated to zero. Derivative calculations were also quite off as compared to the CPU even for higher absorption coefficients. The potential singularity presented in the previous subsection has no repercussions.

4.3.3 The performance of SCA

While numerical accuracy is good enough for the inversion, does the unmixing fare rather worse using SCA. In figure 4.18 is the convergence of SCA compared against the root mean squared error of the result output by the Lawson-Hanson algorithm. For a low number of endmembers or low amount of bands to unmix does the algorithm fare almost as good as Lawson-Hanson with respect to the residual error, although the performance decreases for an increasing number of endmembers and bands. Convergence is very slow after the initial 50 iterations. For the hyperspectral application with the two-layered approach are the wavelength intervals small, consisting of 30 bands. Especially for the melanin determination is the number of endmembers small, typically 3 (two types of blood plus melanin). For the melanin determination stage should SCA therefore serve as a good approximation to the non-negative least squares problem, but the subsequent unmixing would suffer from the use of SCA. Apparently has the error been decreased, but closer investigation of the unmixed parameters will show that there is a large

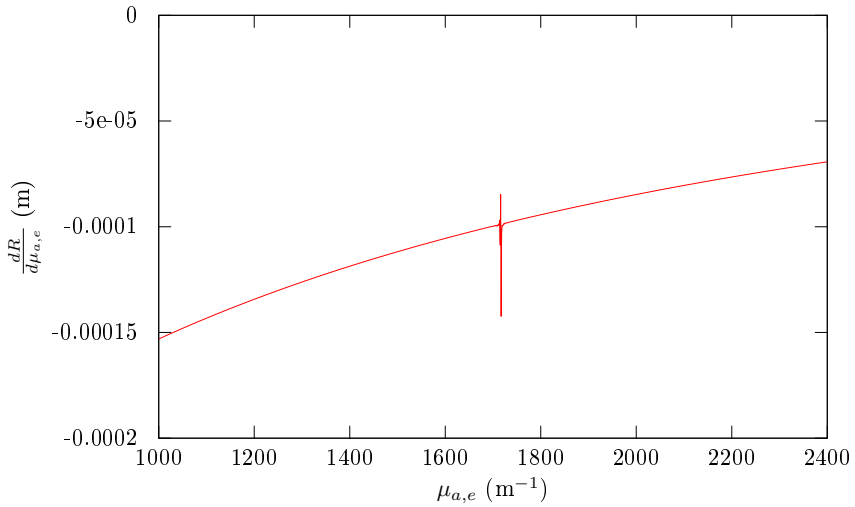


Figure 4.16: The analytical derivative of the isotropic diffusion model as a function of the epidermal absorption coefficient. 80% oxygenation, 1% blood in dermis. The arithmetic operations have been rearranged to minimize the propagation of precision errors.

difference between Lawson-Hanson and SCA. For a low number of bands and endmembers are the results equivalent down to 2 or 3 significant siffers. For 6 endmembers and the same number of bands, Lawson-Hanson can for instance output a dermal melanin coefficient of 0, while SCA would likewise output 182. The discrepancy is too large.

Once the initial iterations are finished does SCA seem to be caught in some kind of lock-in. The first iterations will have established a non-negative estimate for all parameters, and SCA will not be able to push back the parameter if some parameters in reality should be equal to zero. All other parameters already have forced down the error. This is a weakness of varying only one parameter at the time. Convergence was not proven for the algorithm, and it can be likely that there is no convergence proof as the algorithm does not converge. On the other hand, the objective function was seen to decrease, albeit slowly, and after a large amount of iterations may it be able to converge. To say that 300 is enough is a stretch, however, though the error is close to the desired minimum error. The convergence would also depend on the order of variables fitted.

In conclusion can SCA be used for the melanin determination stage as the number of endmembers is low, but further unmixing would not necessarily output good estimates for the parameters.

PQN-NNLS has been shown to have a good performance, with convergence proofs. Still does this not easily parallelize. The line search algorithm it uses will also cause the GPU warps to break down. Some matrix inverses are approximated using numerical methods. The algorithm will quit once reaching a pre-specified numerical accuracy, but it is unknown whether it will diverge or continue to converge after meeting a maximum numerical accuracy when run for a pre-defined number of iterations. Still does this algorithm seem to be better than the alternatives, though whether statistical methods should be used instead of these more classical, deterministic methods is still a question to be answered.

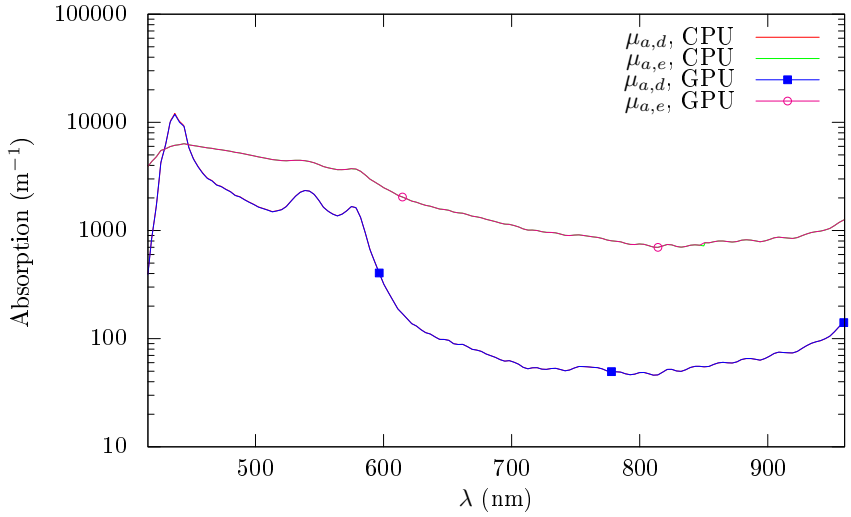


Figure 4.17: Derived dermal and epidermal absorption for CPU and GPU-versions of the same implementation. The dermal absorption was found assuming bvf = 0.01 and $\mu_{a,m,694} = 800$, same for the epidermal absorption except for the melanin coefficient. The spectrum inverted was extracted from line 0, pixel 100 from the hyperspectral image represented by figure 4.19b.

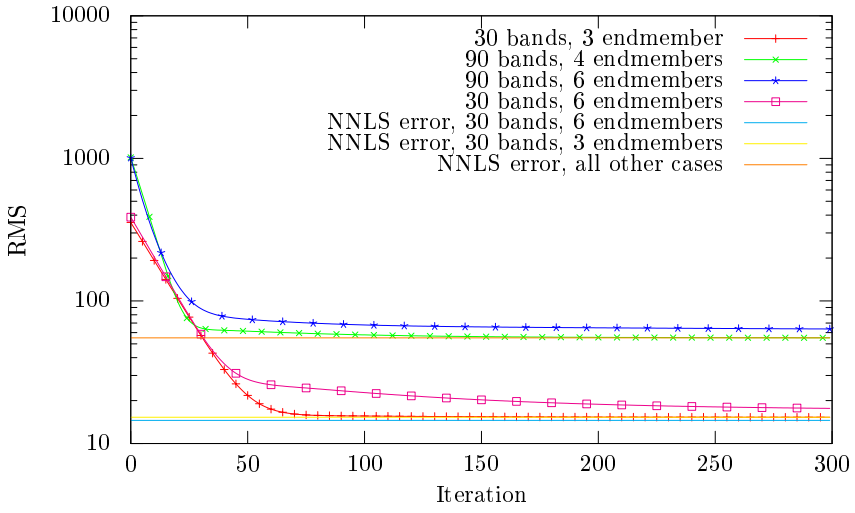


Figure 4.18: Convergence of SCA.

4.4 Verification on real data

The method was first developed for real data. The verification using real data will be presented before the verification using simulated data. The reasoning is that any method may output the correct answer for simulations due to the controlled nature and assumptions being present, but only a few methods will be able to output correct results for real data. Even the melanin and erythema indices may be able to output the correct melanin values for simulated data, and only for real data is the method able to fail.

The hyperspectral images analyzed are

- Image of a healed wound from a chronic ulcer trial [20], shown in figure 4.19b.
- Image of a wound from the same trial, shown in figure 4.48a.
- Image of a normal hand, shown in figure 4.45a. Fitzpatrick skin type I or II.
- Image of the underside of an arm, shown in figure 4.46a. Fitzpatrick skin type IV.

The patients with wounds in figures 4.19b and 4.48a are hyperpigmented due to the healing process of the wounds [12]. High pigmentation can therefore be expected to be seen in these areas. No pigmentation is caused by sun exposure, and other areas should have a low melanin content as the patients were North-European.

All inverse simulations have been done using the two-layered isotropic diffusion model applied on various wavelength intervals in the spectrum.

4.4.1 Melanin verification

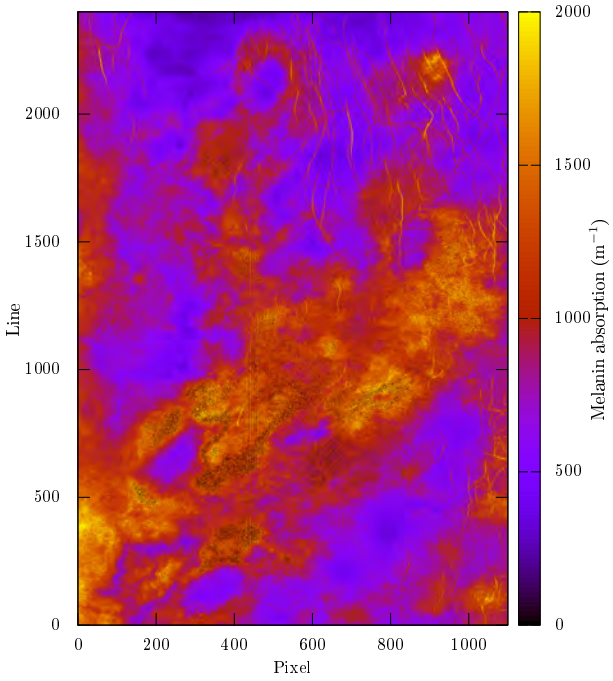
The chosen melanin method will be verified in this section. It will first be shown to overestimate the melanin content without the proper modifications. The use of incorrect melanin type will be investigated as the cause of remaining discrepancies, and lastly, the method is compared against the melanin index.

Overestimation

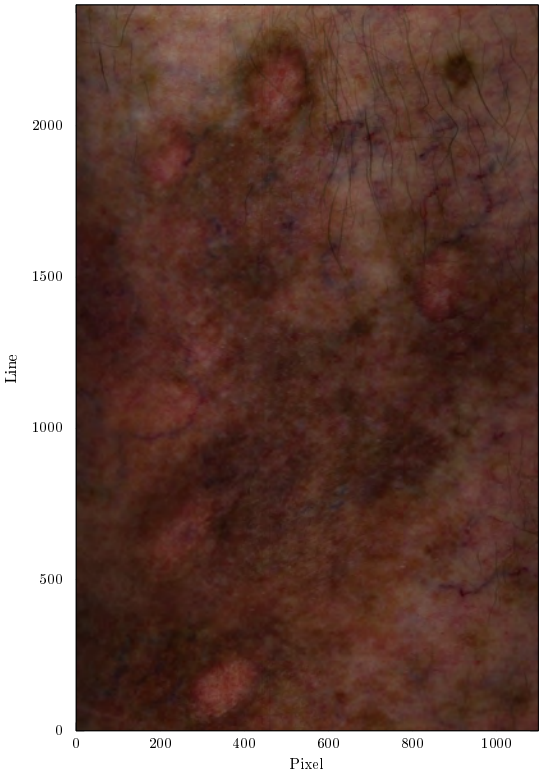
The melanin method, fitting the dermis before fitting epidermis in two iterations, is first verified on a difficult hyperspectral image of a healed wound. The imaged area has spots of hyperpigmentation, and spots within the hyperpigmentation representing healed tissue. Around the old wound will there be scar tissue, with the scattering assumption not being correct. In the picture are there blood vessels present, going in and out of the hyperpigmented spots. The reconstructed RGB image is shown in figure 4.19b. A map of the derived melanin coefficient is shown in figure 4.19a. Here, the epidermal melanin has in each iteration been fitted using Svaasand's melanin curve. The melanin map corresponds well with the places in the RGB image where a high melanin absorption is warranted due to the skin color. It also corresponds well with the spots with less melanin content where wounds previously have been present. The lower melanin values correspond with what is to be expected from this type of skin [61], while it is unknown whether the high values in the hyperpigmentation is within a valid range. Norvang et al. [61] will report tanned, white skin to be 800 m^{-1} and African skin to be 2500 m^{-1}. From visual inspection only should the hyperpigmentation be higher than normal, tanned skin but lower than African skin. Hair can be seen as spots with a high melanin content. This is expected due to the anatomy of hair.

Some discrepancies are clearly present. At some blood vessels, the melanin is set too low compared to the neighboring values. This is due to the blood vessels requiring different boundary conditions than what the diffusion model has to offer, and the melanin is lowered as a result. This will be verified through simulations. There are also spots within the high melanin areas where the melanin is set to zero. Closer investigation revealed these to likely be attributed to erroneous scattering assumptions.

The spectra from three different pixels at line 1000 will first be presented, with an increasing, derived melanin content with increasing pixel index. These are shown in figures 4.26, 4.20 and 4.23, respectively extracted from pixel 200, pixel 440 and pixel 840.



(a) Derived $\mu_{a,m,694}$ for the hyperspectral image represented by figure 4.19b. Melanin has been derived from 730 nm to 830 nm using blood and water in the dermal absorption fit and the melanin curve from equation 2.40.



(b) RGB image reconstructed from a hyperspectral image. RGB image was constructed using band 55 (red), band 41 (green) and band 12 (blue) after reflectance calibration against the reflectance standard.

Figure 4.19: Melanin content and RGB image of a healed wound.

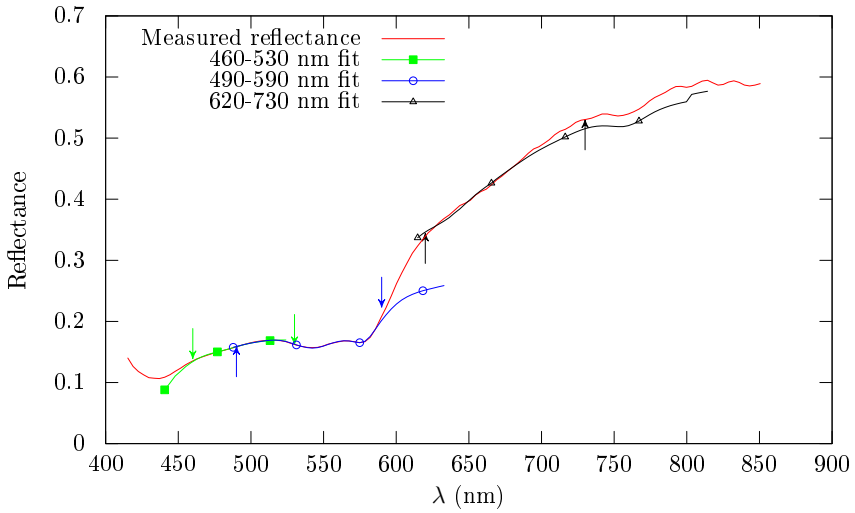


Figure 4.20: Inverse simulation where too low melanin is set or penetration depths at the longer wavelengths are too steep. The spectrum is located in line 1000, pixel 440 of figure 4.19b. Derived parameters are shown in table A.1. Fitting ranges are indicated by arrows.

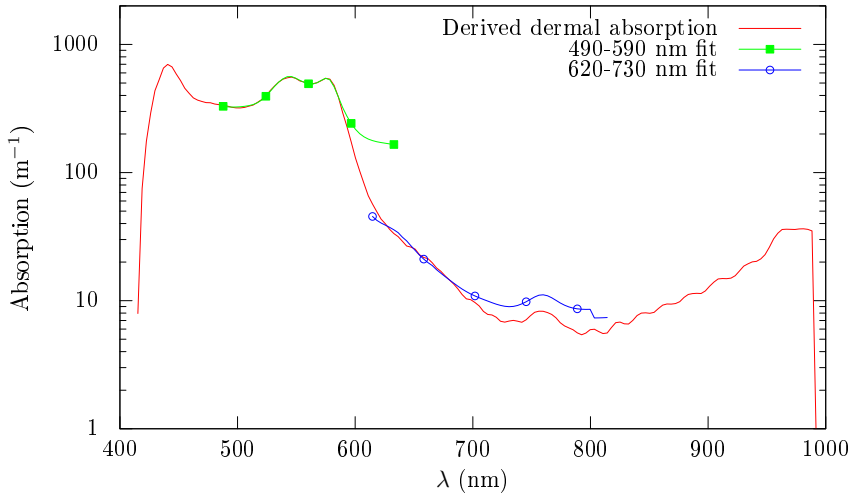


Figure 4.21: Dermal absorption fit for figure 4.20.

It was discussed earlier that deoxy hemoglobin would likely be overestimated, and melanin in turn underestimated. Figure 4.20 shows a situation where deoxy *apparently* is overestimated and melanin underestimated, but more likely a failed unmixing using SCA and a penetration depth error. The fit at the shortest wavelengths does not display any misfits spectrum-wise, but the oxygenation does not match the oxygenation obtained from 490-590 nm and is unphysical. In addition is the constant absorption set very high, methemoglobin has been set to something non-zero and there is a high dermal melanin coefficient. Apparently has the melanin been underestimated and the fit is trying to compensate. This can, however, be explained away by the fact that SCA has been used in the unmixing and SCA was in the previous section proven to be unreliable in the presence of too many endmembers. Especially is this proven by the next wavelength interval, where a lower amount of endmembers is used in the unmixing. There is no obvious misfitting which should have been present had the melanin been set too low. The spectra match, and neither melanin nor methemoglobin are overcompensated in dermis. The fit from 630 nm and up is however far worse. The simulated reflectance and measured reflectance intersect. The simulated reflectance has a pronounced deoxy absorption dip around 750 nm which indicates that it is trying to compensate for something with more deoxy hemoglobin and methemoglobin. There are no obvious spectral characteristics present in the spectrum to indicate that methemoglobin is present.

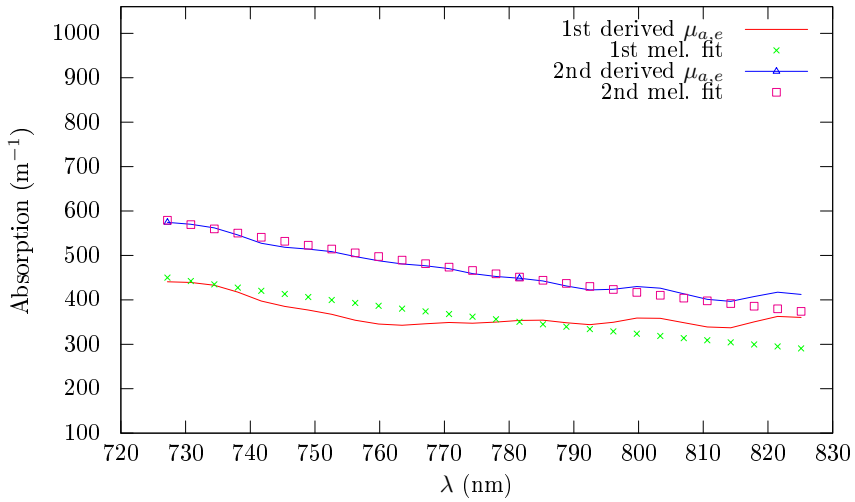


Figure 4.22: The melanin fitting procedure of the spectrum displayed in figure 4.20. Dermal fitting is omitted. Note the change in wavelength ranges.

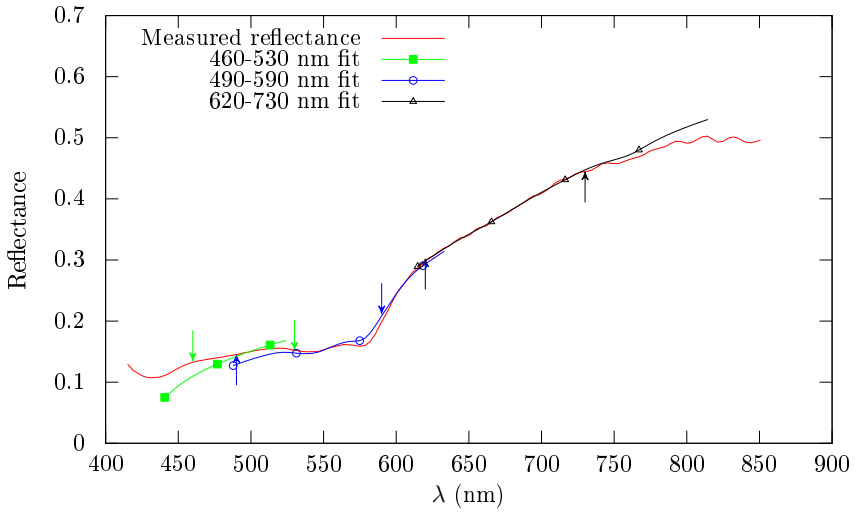


Figure 4.23: Inverse simulation where too much melanin is being set. Line 1000, pixel 840 of figure 4.19b. Derived parameters are shown in table A.1.

The melanin fitting of the required epidermal absorption is shown in figure 4.22. Only the epidermal absorption fit is shown as the dermal absorption fit shows no indication of any misfitting and is therefore uninteresting. The first iteration of the derived epidermal absorption shows a distinct dip where the deoxy absorption maximum usually is located, indicating a deoxy hemoglobin overestimation and melanin underestimation. This does seem to have been eliminated in the next iteration, verifying that two iterations will rectify the underestimation. Still, something is wrong, and it may be that the wavelength interval at the longer wavelengths vary across a too high penetration depth gap, leading to misfitting. Comparison of the derived dermal absorption in figure 4.21 with later dermal absorptions in figures 4.30 and 4.28 will show this to be far more sloped in the wavelength range in question.

Figure 4.25 shows the process of a melanin overcompensation. The dip in the derived epidermal absorption shows that the deoxy hemoglobin was overestimated in the corresponding dermal fit. In the next iteration is the epidermal fit, as above, more or less perfect. Still, the melanin is obviously being overestimated as is seen in the final spectral fit in figure 4.23 and the absorption spectrum in 4.24. This can be due to a circular loop. The first melanin estimate is so high that the subsequent dermal fitting will mis-estimate, and the mis-estimation results in the same epidermal absorption as the input epidermal

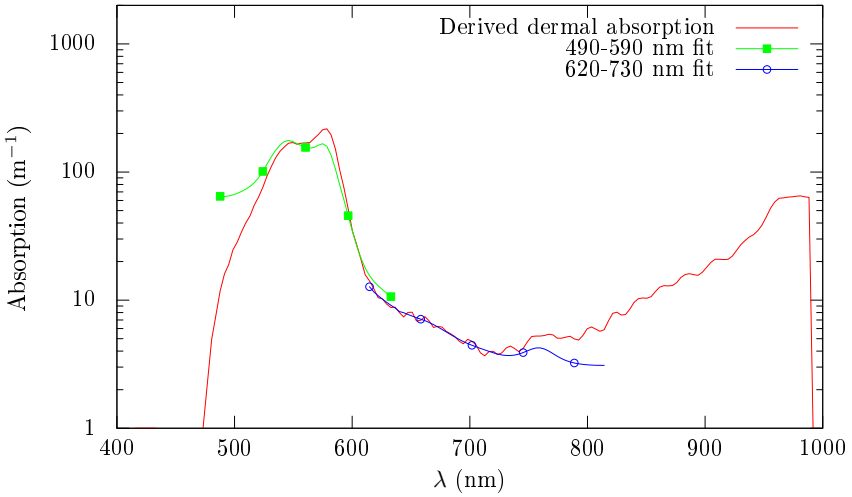


Figure 4.24: The dermal absorption fit for figure 4.23.

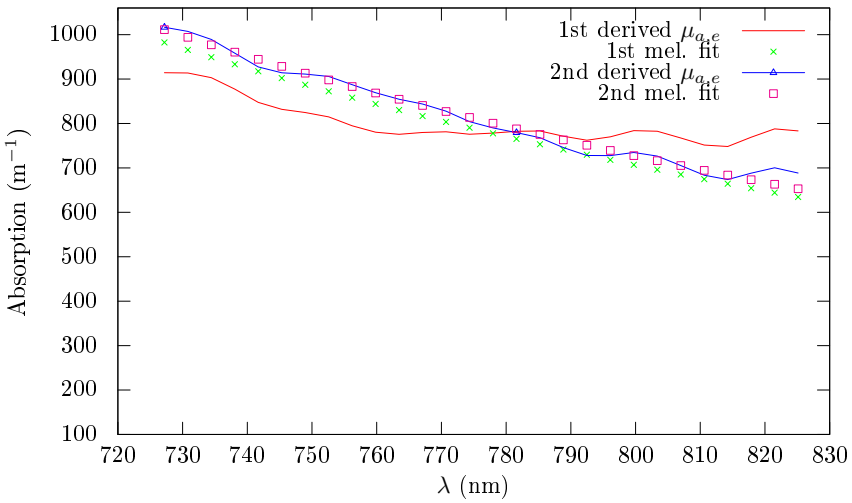


Figure 4.25: The melanin fitting of the spectrum displayed in figure 4.23. Dermal fitting is omitted.

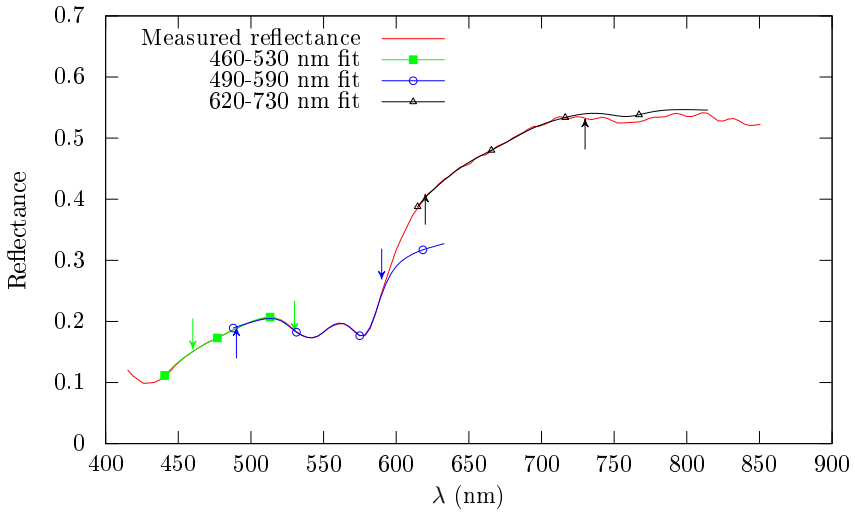


Figure 4.26: Inverse simulation for approximately the correct melanin. Line 1000, pixel 200 of figure 4.19b. Fitting parameters are shown in table A.1.

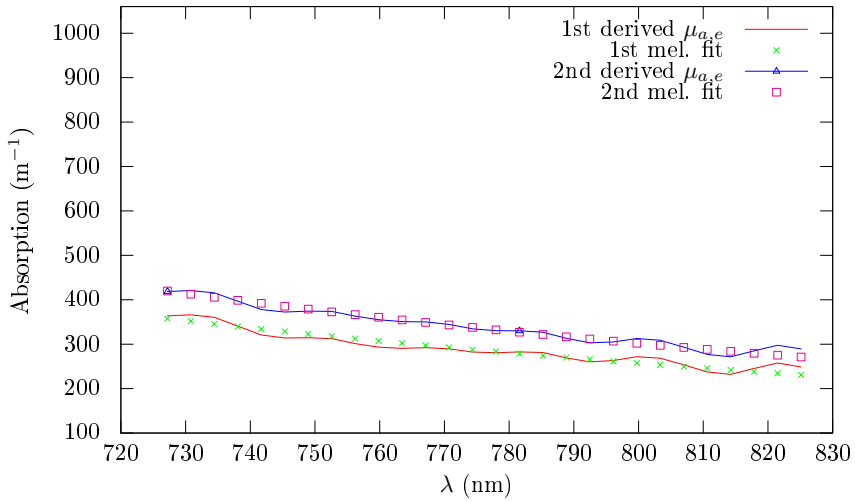


Figure 4.27: The melanin fitting of the spectrum displayed in figure 4.26

absorption. There may also be something wrong at the lower wavelengths, like the wrong melanin type being used.

Moving back to figure 4.20, it was seen that the melanin estimate above 620 nm was apparently too low, although the same melanin estimate at the lower wavelengths resulted in no overestimation of other parameters. The lower wavelengths would have suffered, had the longer wavelengths been allowed for a good melanin fit. This might be readily demonstrated by 4.23. Here is the fit for the lower wavelengths indeed worse when the higher wavelengths are allowed for a good melanin fit. Given that the melanin absorption is too high for the last presented spectrum, which it is, as will be seen, is this strongly in favour of the penetration depth variance being the cause of discrepancies in figure 4.20.

A better melanin fit is shown in figure 4.26, with the melanin fits present in figure 4.27 and the full dermal fit in figure 4.28. Nothing is overcompensated to correct for a missing melanin at any wavelength. The blood parameters derived for 450 nm and up and 490 nm and up match to a certain extent. The simulated reflectance does not match the measured reflectance exactly above 700 nm, but this is expected as the water absorption is not fitted along with the rest of the parameters. The general shape is the same, the reflectances do not intersect at any point, and most importantly does the simulated reflectance lie above the measured reflectance. The difference will therefore be rectified by more absorption, not less.

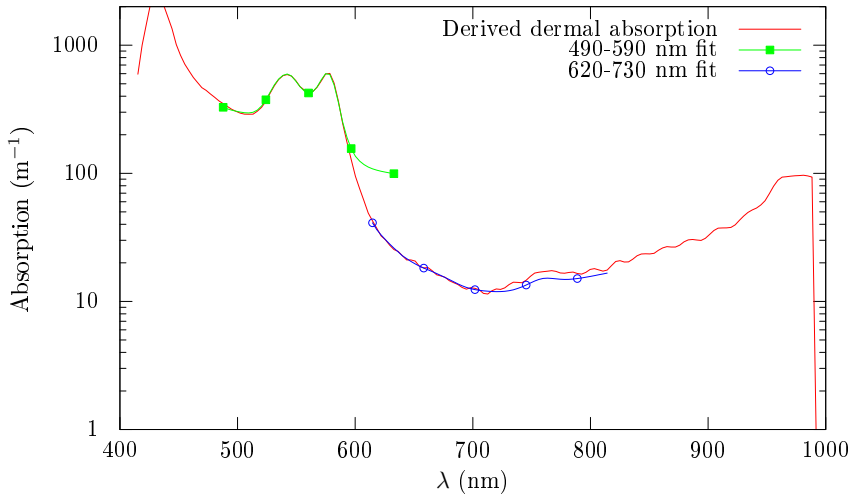


Figure 4.28: Dermal absorption fit for figure 4.26.

This spectrum, however, represents a rather low melanin content. The first dermal fitting error is expected to be less when the difference between the actual melanin absorption and start melanin absorption (100 m^{-1}) is low, as was seen from simulations. The first derived epidermal absorption coefficient has certain spectral characteristics reminiscent of earlier fits, suspecting a deoxy overcompensation also in this case, but this has evidently no dire ramifications. The next iteration is able to correct for this, as was expected from the simulations.

The way other melanin methods previously behaved was that a high deoxy hemoglobin content was mistaken as the presence of a high melanin content. In this case, the deoxy hemoglobin is evidently being overcompensated but the melanin is in turn also overcompensated. It is apparent from the individual melanin fits that the first iteration of the melanin method can, in part, show when a discrepancy is expected. Looking at figure 4.25, the first melanin curve fit to the epidermal absorption is non-optimal. The derivative is completely different and the curves intersect. This behavior is not seen for figure 4.27.

There are several possible explanations for this behavior.

- Oxy hemoglobin is rectifying the deoxy behavior. When deoxy hemoglobin is overcompensated is oxy hemoglobin somehow overcompensated, and although the dermal, initial melanin is low is the slope of the resulting epidermal absorption curve very off. The melanin curve is not able to fit the resulting curve, off-shooting it.
- The melanin curve used is not representative of the melanin *types* present in this particular skin, indicating that a different melanin curve should be used in the fit.
- Numerical inaccuracies, especially in the context of SCA.
- General misfitting. Although the dermal melanin is set in dermis is the dermal melanin not correct since the curve should have been convolved, and the resulting epidermal absorption should not be reminiscent of the melanin curve in any case.
- The pixel with a too high melanin estimate is right at the boundary of a high melanin region, and the surrounding high melanin properties are bleeding into the pixel.

Numerical inaccuracies are likely to be small, SCA or no SCA. While the GPU version gave a melanin absorption of 1188 m^{-1}, the corresponding CPU version using the same wavelengths and chromophores in the fitting gave a melanin absorption of 1192 m^{-1}. The CPU version had the same misfitting problems as the GPU version around 530 to 590 nm. The discrepancy is small and can be attributed to other differences in the implementations than the use of SCA versus Lawson-Hanson.

The used melanin curve from (2.40) is thought to be a combination of eumelanin and pheomelanin. This will not always be correct, different individuals will necessarily have different combinations of melanin

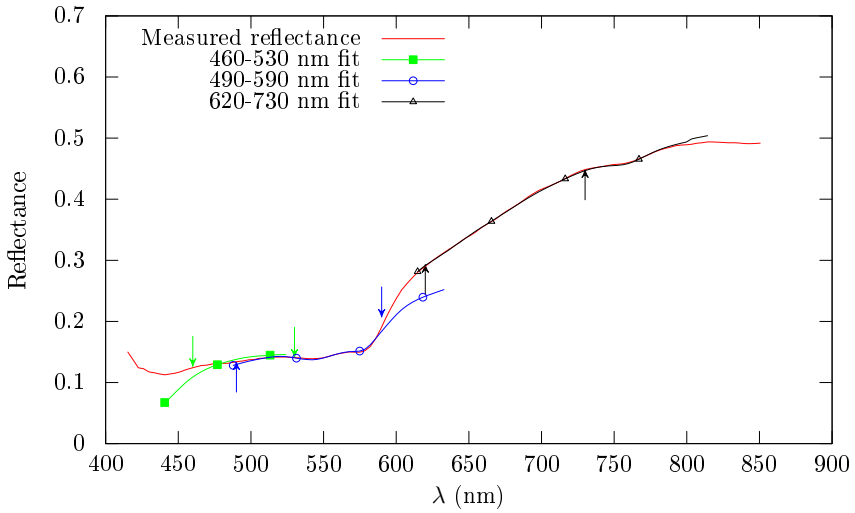


Figure 4.29: The same spectrum as in figure 4.23, though now with a straight line being used in the epidermal fitting. Fitting parameters are shown in figure A.1.

types in their skin. Eumelanin was described to be brown and black, while pheomelanin is more yellow-reddish. This obviously cannot be discerned from a reconstructed RGB image using only three bands from a hyperspectral image, but purely by speculation does the hyperpigmented parts of 4.19b seem to be mostly brown-black, indicating eumelanin. The hair is also black, more strongly indicating eumelanin. There are however some lighter brown parts being more yellow than brown. Had the melanin type present been eumelanin, would this have explained the good fit at the longer wavelengths and worse fit at the shorter, as eumelanin has a lower absorption at the shorter wavelengths when compared to the longer wavelengths and the behavior of pheomelanin and Svaasand's model.

On the other hand is it unlikely that the slope of the derived epidermis absorption in this case is affected by the presence of pheo- or eumelanin. The absorptions are fitted to the reflectance using the two-layered model. The derived epidermal absorption is calculated after the dermal absorption coefficient has been derived and calculated. The epidermal absorption coefficient will be influenced by the dermal absorption coefficient's wavelength-dependency. It was shown by trial of simulations that for a perfect match would the melanin have to be convolved in the dermis fit, and when this is not done must the resulting epidermal absorption be different than the melanin curve. The wavelength-dependency at this stage can therefore not be influenced by the presence of the different melanin types.

The bleeding between pixels and melanin types will be important, but not in this case. This is purely a misfitting problem.

The general melanin level is a correct, lower estimate, but the wavelength-dependency is wrong and not fittable to any melanin curve. The result is a too high melanin estimate, as the curves must intersect. The difference will become worse with increasing melanin absorption. The better way to fit this epidermal curve will therefore be to fit it using a straight line and extrapolate it to 694 nm. The required melanin level will be obtained without major overestimations, as the straight line may adjust its derivative, as opposed to what the melanin curve is able to do. The result of the linear epidermal fit for the spectrum in figure 4.23 is shown in figure 4.29. This is far better than the first proposed melanin estimate, though a very close look will reveal that the fit is slightly, almost not noticeably, off for 530-590 nm. At the shorter wavelengths is the fit even more off. From 400-450 nm is the signal to noise ratio low, and the spectra here cannot be expected to be fittable. Still should the dermal absorption not approach zero, as is seen in the dermal absorption fit in figure 4.30.

The melanin error is best discerned looking at the error at 530 to 590 nm. The squared error at this interval is therefore used to indicate whether or not the melanin has been well fitted to the reflectance. A map of the error when the melanin curve has been used in the fitting is shown in figure 4.32a, while the error using the linear fit is shown in figure 4.32b. As is seen is the error lessened, and in some places

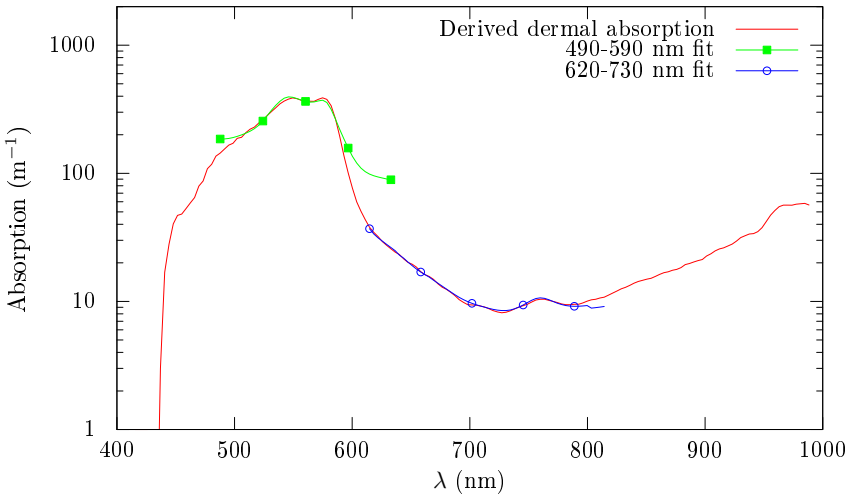


Figure 4.30: Derived dermal absorption for the spectrum in figure 4.29.

removed entirely. At other places is it not. The error estimate is not a perfect, absolute estimate and will only calculate the error within the fitting range, but it can used for comparison between areas. A map of the melanin coefficient using straight line fitting is shown in figure 4.31.

The change of methods fixes some overestimation error, but the rest must be attributed to something else.

Melanin type

The choice of melanin type will be investigated in this section. Investigating single spectra fits, it is apparent that using eumelanin is warranted for some pixels, and using pheomelanin is warranted for others. Still can this also be attributed to other errors.

Figure 4.33 shows an ill fit using Svaasand's model. Eumelanin, on the other hand, is able to give a better fit with no gross overestimation of melanin or methemoglobin in dermis. Pheomelanin gave approximately the same results as Svaasand's model (not shown), which is not surprising as Svaasand's model seem to be closer to pheomelanin than eumelanin.

Though Svaasand's model fails, the eumelanin model will not necessarily give a better estimate, as shown in figure 4.34. This is explained away by the fact that this pixel lies exactly in the middle of a hair straw which will make the skin model incorrect.

Figure 4.35 shows a slight, not easily discernible misfit for the Svaasand model case around 530-590 nm. The eumelanin model gives a far too low estimate, seen from the overestimation of dermal melanin around 530-590 nm and the longer wavelength fit lying below the spectrum. The absorption spectra for this particular case are shown in figure 4.37. The scale of the blood absorption peaks around 530-590 nm is wrong for the pheomelanin case. Pure pheomelanin is not warranted. Pure eumelanin is not warranted either due to the scaling of the absorption peaks being sloped in the other direction and much melanin being set in dermis. Either will this be due to a wrong scattering assumption, or a combination of the two types is warranted.

Figure 4.36 shows a case where there is only a small difference between the goodness of fit of eumelanin and pheomelanin. The explanation is that the melanin absorption is low, and the difference between pheomelanin and eumelanin will also be low and the error in assuming either types is lessened.

The RGB image around the pixels displayed above is shown in figure 4.38. The good eumelanin fits shown in figures 4.33 and 4.34 are dominated by hair. The fitting here is done assuming independent pixels, but there will be light pollution between pixels. The hairs are black, which would mean eumelanin. The image representing the fit in figure 4.35, which was an ill fit either way, shows hyperpigmented skin.

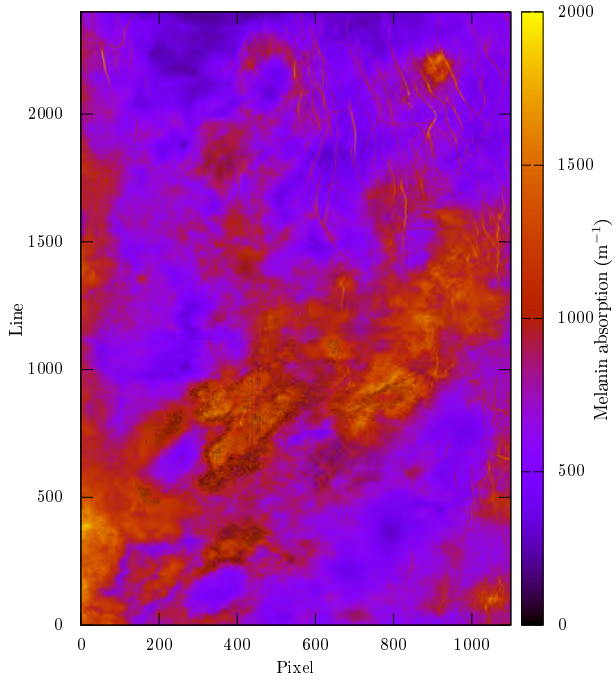


Figure 4.31: Map of $\mu_{a,m,694}$ after substituting the melanin curve fitting with a straight line fitting.

Referring back to the RGB image in its entirety in figure 4.19b, it is seen that this is close by a small area with low melanin content in a sea of hyperpigmentation. This is therefore expected to be an old wound with scar tissue, and the scattering assumption will be incorrect.

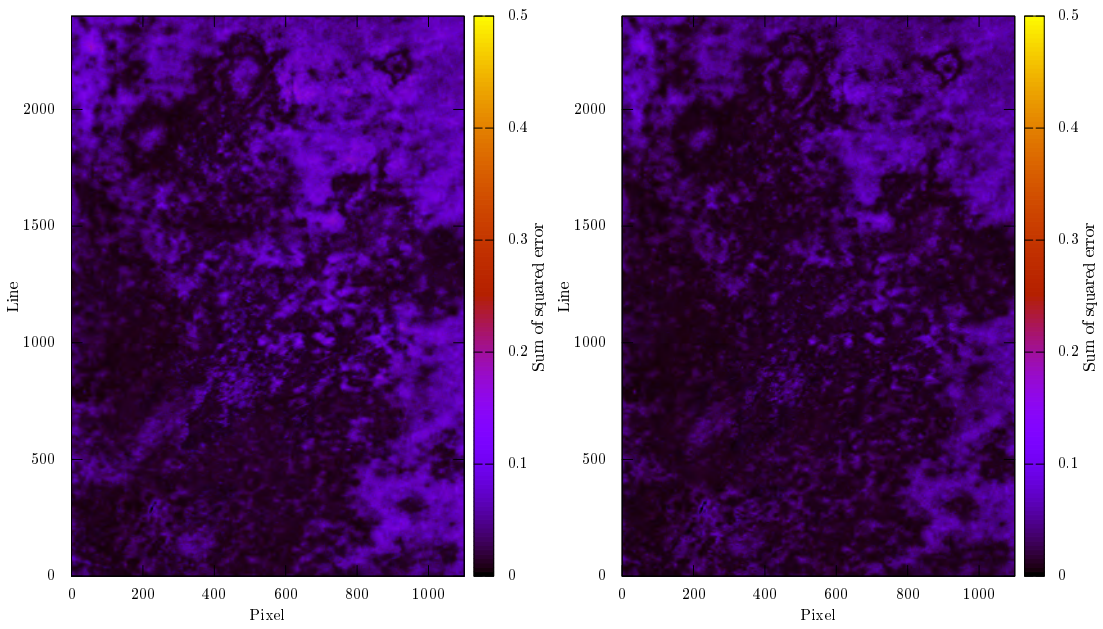
For each chosen model, the model was used throughout all iterations and in both epidermis and dermis. It is worth to note that all three models will output approximately the same melanin absorption at 694 nm, at least the same order of magnitude, and the subsequent goodness of fit is only due to the wavelength-dependency of each melanin model. Detecting the correct type of melanin should therefore not be dependent on what melanin curve was used in the melanin iteration itself, since the melanin coefficient is approximately the same regardless of the melanin curve used, but only be dependent on the later choice of the melanin type. This should ease the melanin type detection.

A map of the necessary melanin types is shown in figure 4.39, generated by thresholding the errors by assuming different melanin types. It can be seen that the spots representing where neither melanin type is a good fit do largely appear where old wounds can be expected to be located, although the spots around line 1000 do not seem to be so. The pheomelanin type does also appear only at places where a high melanin content is expected, while eumelanin gives a good fit only where the melanin content is low. This is strange, as the melanin in these parts is very dark. Eumelanin should be expected to be found in the darker parts when eumelanin is found in the lighter parts.

This may be yet another overestimation error, since eumelanin has a lower absorption at the lower wavelengths compared to pheomelanin and might be able to rectify overestimation errors. Figure 4.33, however, showed that only eumelanin was able to give a good fit regardless of the absorption values at the upper wavelengths. There was also no obvious underestimation errors for the eumelanin case. Any underestimation errors have in figure 4.39 been detected by using the error value obtained from the upper wavelengths, as these would give a large error due to deoxy hemoglobin overestimation when melanin was underestimated.

Eumelanin was able to give an apparent good fit at the lower wavelengths also for the apparent pheomelanin areas, but this was only due to overestimation of other parameters like methemoglobin and melanin in dermis. This was detected from obvious deoxy hemoglobin overestimation at the upper wavelengths. Only using eumelanin at these areas will therefore apparently not be correct.

One of the fits at the apparent pheomelanin areas is seen in figure 4.40. The fit itself is worse for the



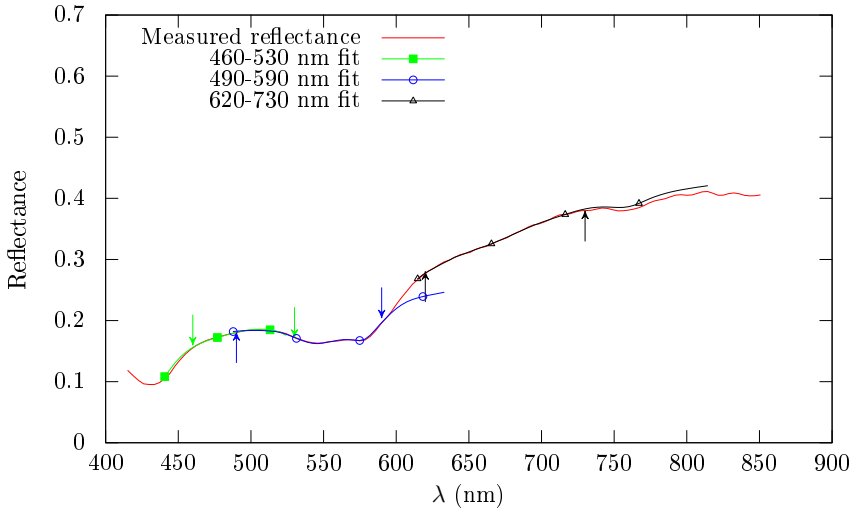
(a) Melanin curve.

(b) Straight line.

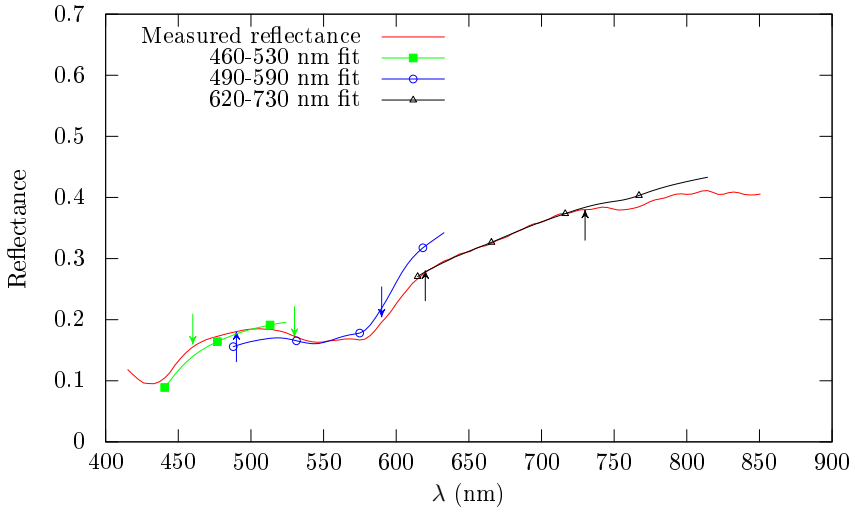
(c) Straight line, eumelanin.

(d) Straight line, pheomelanin.

Figure 4.32: Comparison of sum of the squared error at 500-590 nm using the melanin curve and a straight line to fit the epidermal absorption, and use of pheomelanin and eumelanin.

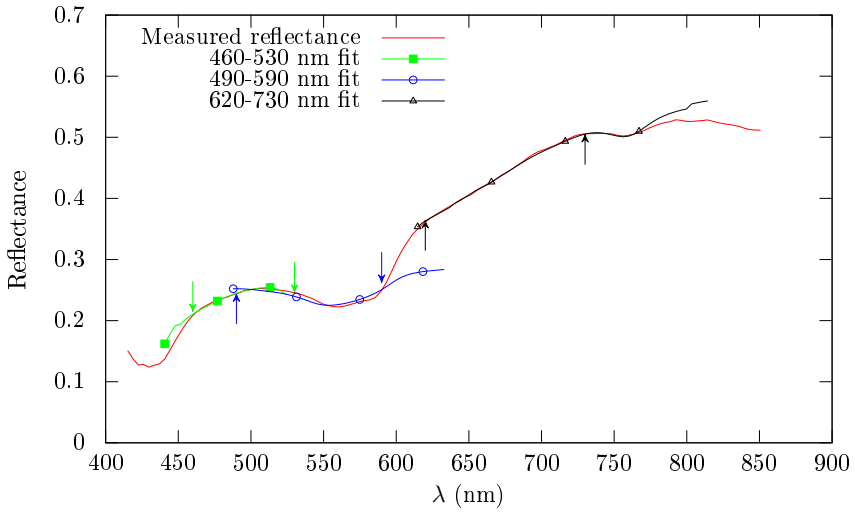


(a) Fit using the eumelanin model.

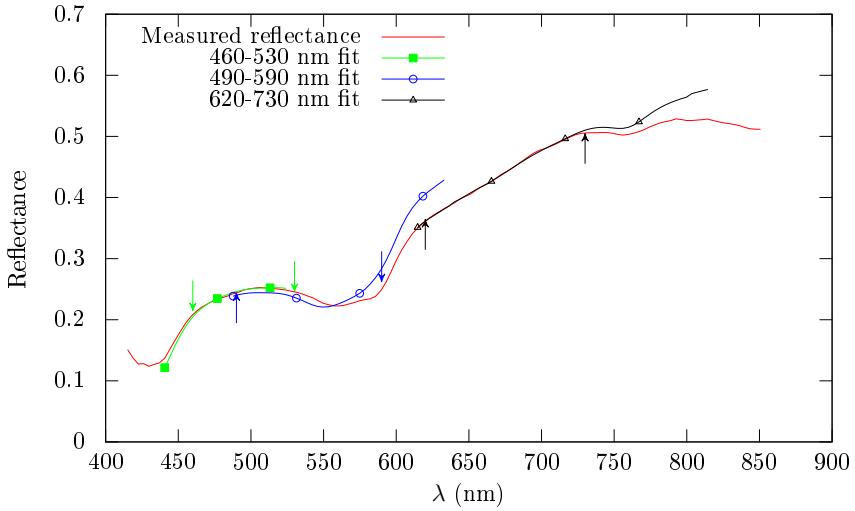


(b) Fit using the Svaasand melanin model.

Figure 4.33: Eumelanin giving a better fit. Comparison of Svaasand's melanin model and eumelanin for an isotropic, two-layered diffusion model fit using the GPU on the spectrum located in line 1653, pixel 1029 of figure 4.19b. Parameters in table A.1.

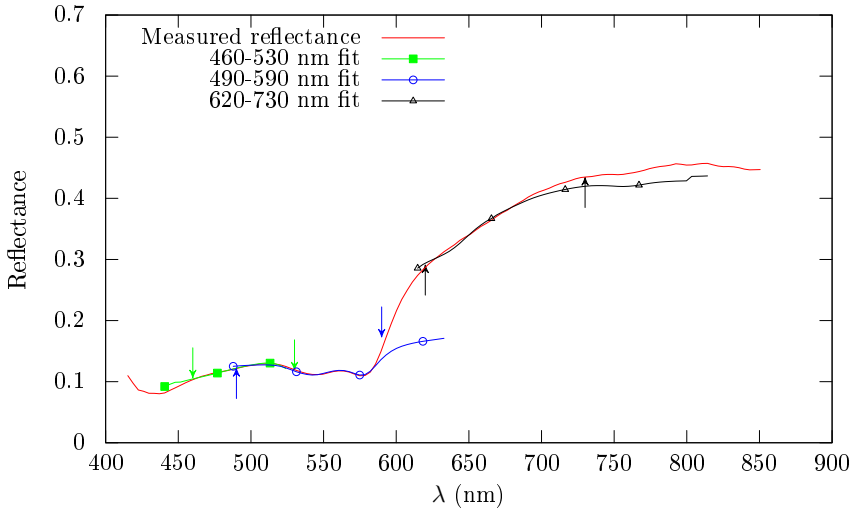


(a) Fit using the eumelanin model.

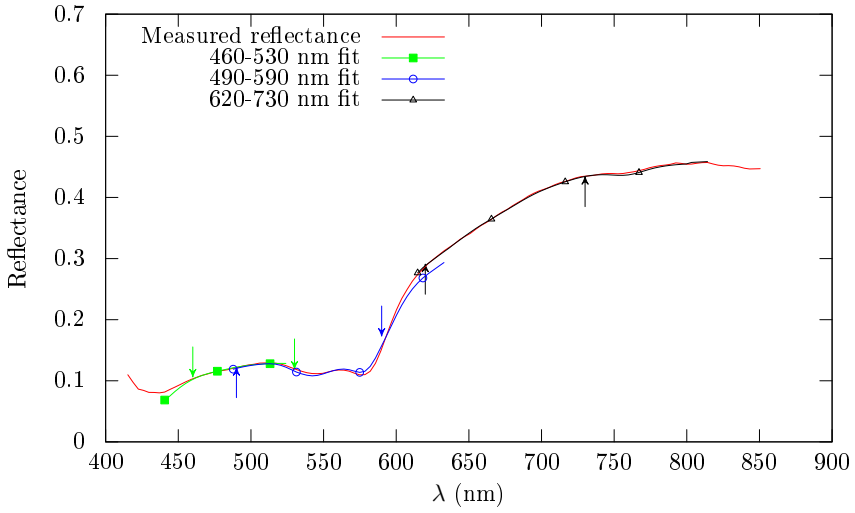


(b) Fit using the Svaasand melanin model.

Figure 4.34: Svaasand's model giving a non-optimal fit, eumelanin only slightly better. Spectrum from a straw of hair. Comparison of Svaasand's melanin model and eumelanin for an isotropic, two-layered diffusion model fit using the GPU on the spectrum located in line 2015, pixel 847 of figure 4.19b. Parameters in table A.1.

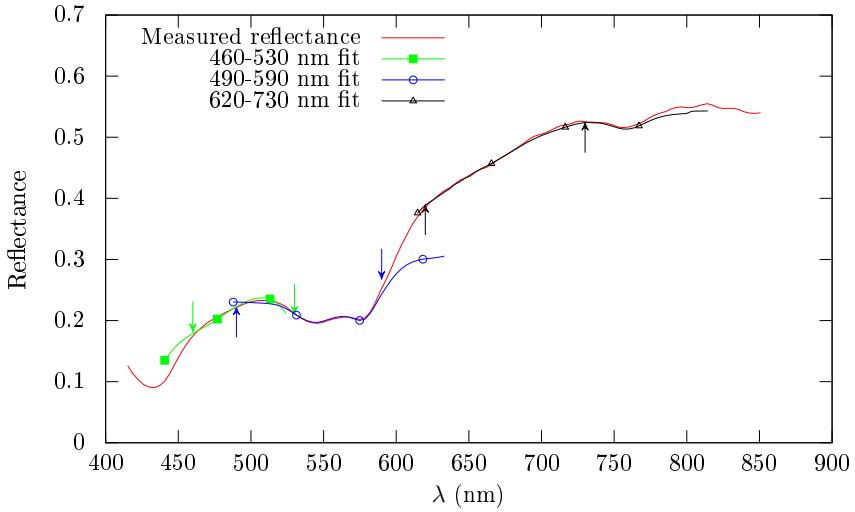


(a) Fit using the eumelanin model.

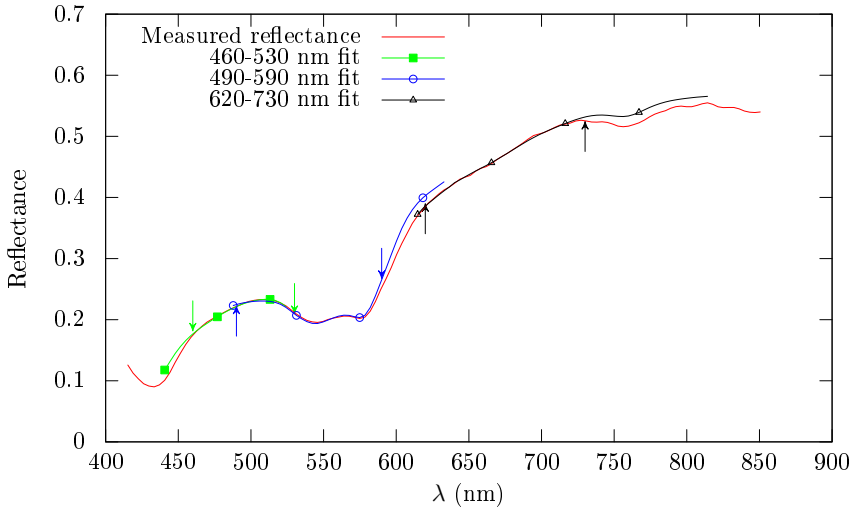


(b) Fit using the Svaasand melanin model.

Figure 4.35: Misfit for Svaasand's model not easily discernible. Comparison of Svaasand's melanin model and eumelanin for an isotropic, two-layered diffusion model fit using the GPU on the spectrum located in line 2015, pixel 847 of figure 4.19b. Fitting parameters in table A.1.



(a) Fit using the eumelanin model.



(b) Fit using the pheomelanin model.

Figure 4.36: Small difference between eumelanin and pheomelanin for a low absorption case. Comparison of pheomelanin and eumelanin for an isotropic, two-layered diffusion model fit using the GPU on the spectrum located in line 756, pixel 933 of figure 4.19b.

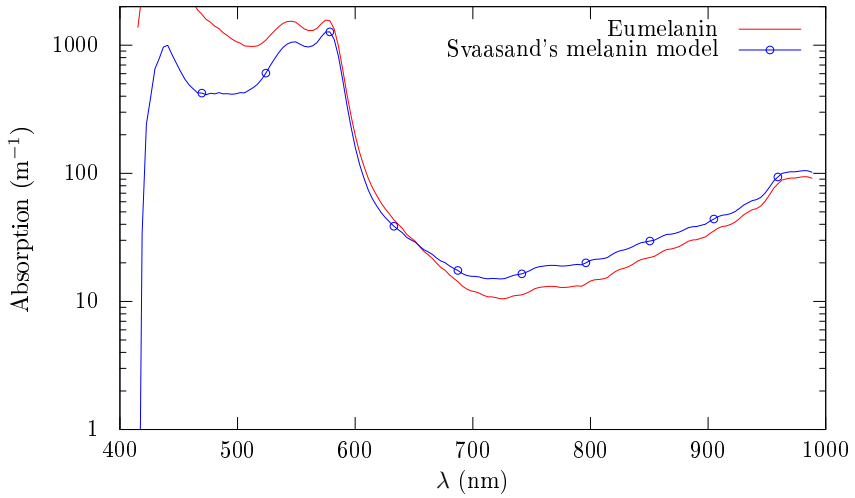


Figure 4.37: The derived dermal absorption using the two layered diffusion model for the spectrum shown in figure 4.35.

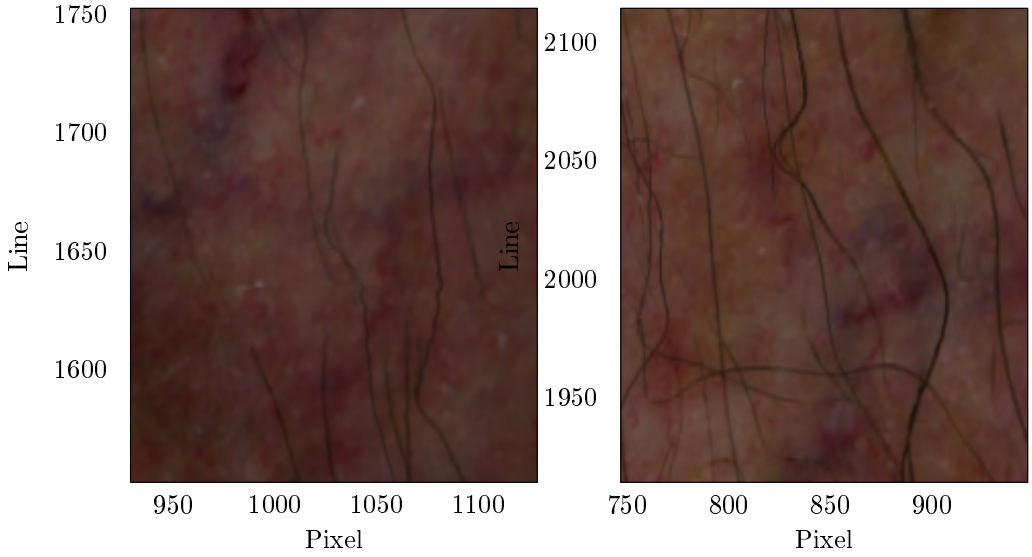
eumelanin case, where deoxy hemoglobin and other parameters have been overcompensated to compensate for the lacking melanin absorption, and apparently better for pheomelanin. But a closer look yields slightly unphysical values - for one thing has the melanin coefficient at 694 nm been underestimated for pheomelanin (600 m^{-1}) as compared to eumelanin (800 m^{-1}), but again compensated for using melanin in dermis. The output parameters for oxygenation are also not physical. Using the two-layered diffusion model, oxygenation values equal to 1 should never be possible due to being a mean value of different layers. However, this can also be attributed to a slight scattering error causing the scaling of the absorption maxima to crave a very low deoxy value. In any case, this particular pixel is inconclusive with regards to the melanin type.

The spectra fits from one of the larger black spots representing neither melanin type is shown in figure 4.41. This shows an apparent bad fit using pheomelanin, but an underestimation using eumelanin. There are no obvious scattering errors, pointing towards the conclusion that the melanin type is neither, but rather a combination of the two.

Apparently, the presence of eumelanin and pheomelanin can be detected using the derived dermal absorption values around 530-590 nm, but it can only detect the extrema - where there is so much pheomelanin present that eumelanin is negligible, or when the presence of pheomelanin is negligible and only eumelanin can be assumed. It will be more difficult to detect how much should be present of each melanin type without resorting to tiring iteration when both eumelanin and pheomelanin are present in non-negligible amounts. Svaasand's model, a mix of the two, will not always be able to correct this as it represents a different ratio between eumelanin and pheomelanin than might be present in the skin at hand.

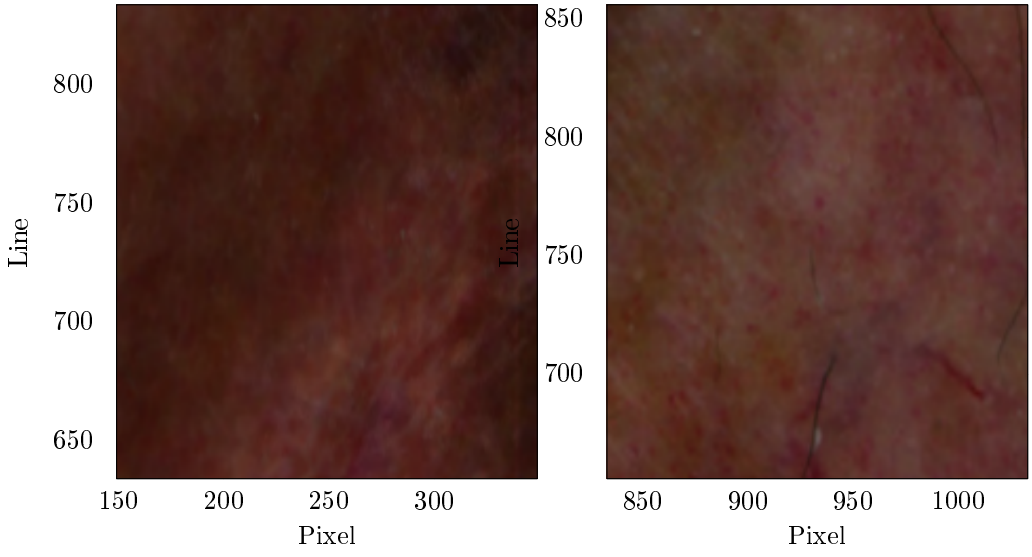
The result of deciding the melanin type from the blood absorption peaks is shown in figure 4.42. Sample spectra from each possible area is shown in figure 4.44. The melanin type map does correspond to a certain extent to the earlier melanin map composed from the error maps. Most notably, the spots in the middle of the error-constructed image representing neither melanin type has not been recognized as spots with erroneous scattering. Figure 4.44 shows that the spectrum matching none of the melanin types is correctly identified as erroneous scattering, but this is not as obvious for the spots which earlier were identified as having none of the melanin types. It is also seen in 4.44 that the melanin types are switched with increasing melanin content.

The melanin being set too low is one likely explanation for the pheomelanin requirement as the method is steering towards doing exactly that. The underestimation would be worse for increasing melanin content, explaining the switch in melanin type for increasing melanin content. In figure 4.43 is the melanin being underestimated, and pheomelanin has been chosen to rectify it. Other pixels will not show such grossly obvious underestimation, especially for the higher melanin contents. Other pixels where pheomelanin is chosen will show behavior more reminiscent of the plots where the penetration depth is called into



(a) Subset 1

(b) Subset 2



(c) Subset 3

(d) Subset 4

Figure 4.38: Zoomed subsets of the RGB image shown in figure 4.19b.

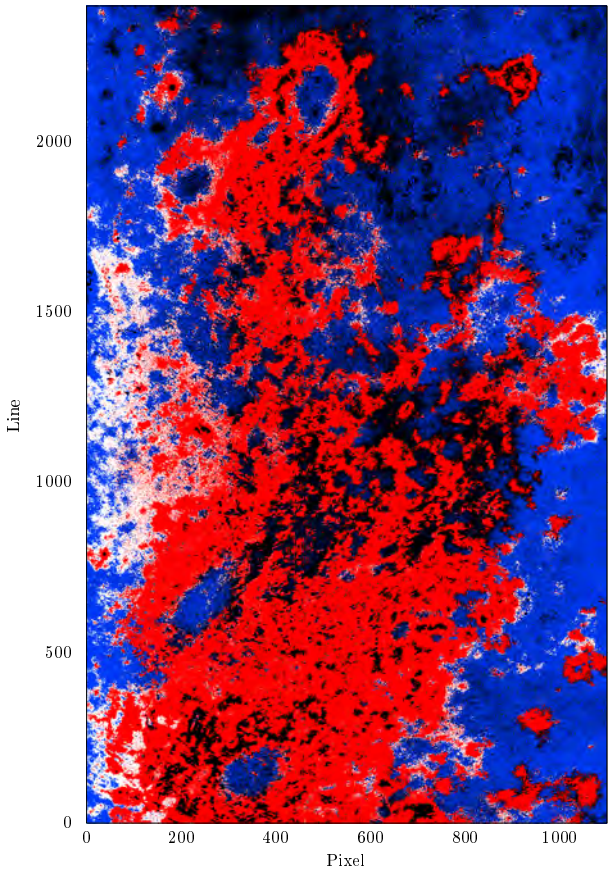
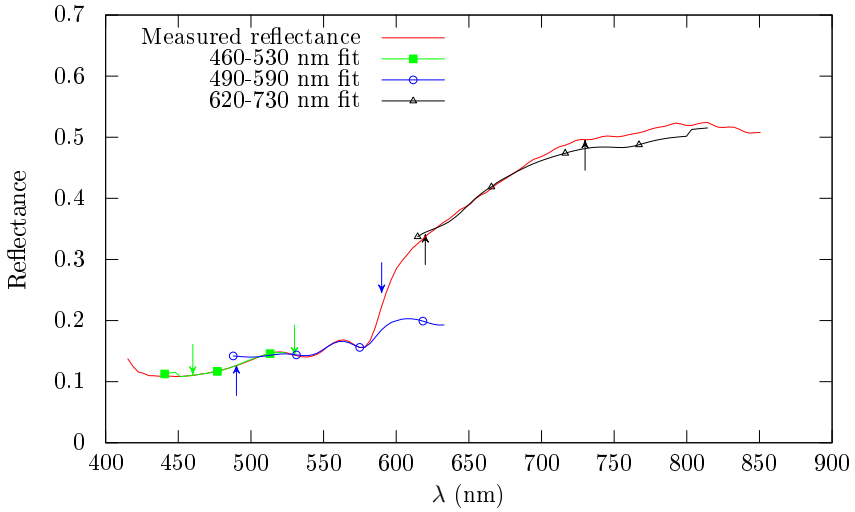
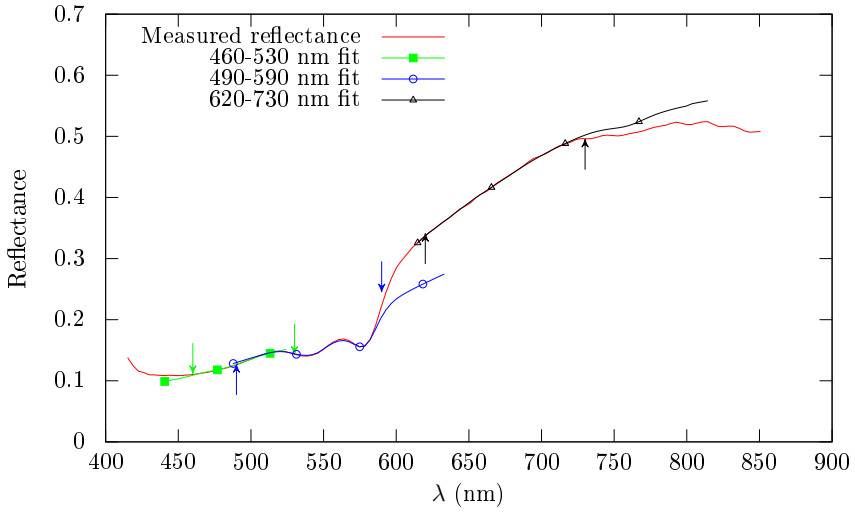


Figure 4.39: Map of the necessary melanin types. Red is pheomelanin, blue is eumelanin, black is neither and white is both. The map was generated comparing the root mean squared error at 530-590 nm and 620-720 nm of either fit against an arbitrary, low threshold value (0.02).

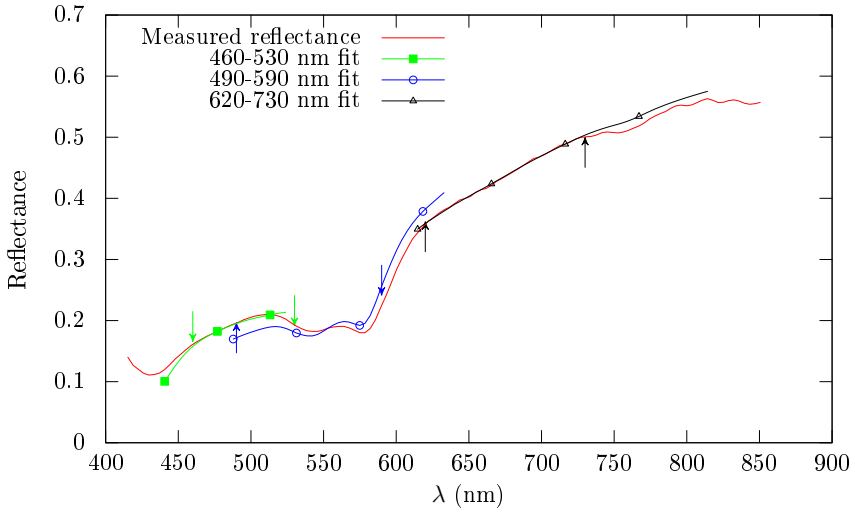


(a) Fit using the eumelanin model.

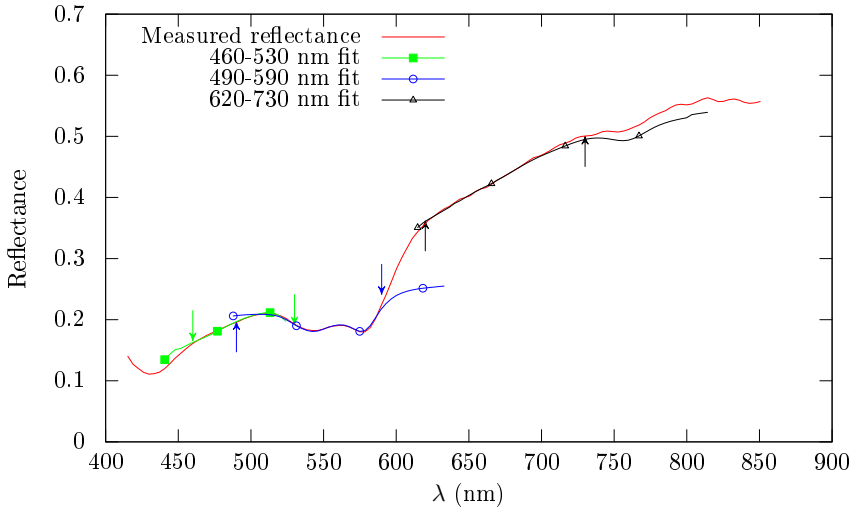


(b) Fit using the pheomelanin model.

Figure 4.40: Fits using pheomelanin and eumelanin in an area detected as pheomelanin. Comparison of pheomelanin and eumelanin for an isotropic, two-layered diffusion model fit using the GPU on the spectrum located in line 389, pixel 527 of figure 4.19b. Parameters in table A.1.



(a) Fit using the pheomelanin model.



(b) Fit using the eumelanin model.

Figure 4.41: Overestimation using pheomelanin, underestimation using eumelanin. Comparison of pheomelanin and eumelanin for an isotropic, two-layered diffusion model fit using the GPU on the spectrum located in line 1163, pixel 719 of figure 4.19b. Parameters in table A.1.

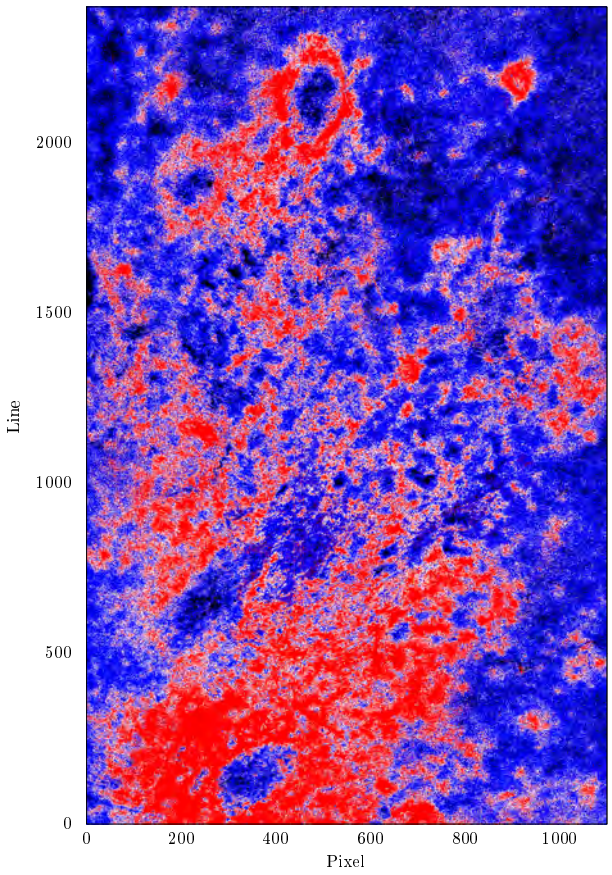


Figure 4.42: The melanin type distribution as decided from the scaling of the blood absorption peaks around 530-590 nm. Red is pheomelanin, white is Svaasand's model, blue is eumelanin, black is neither.

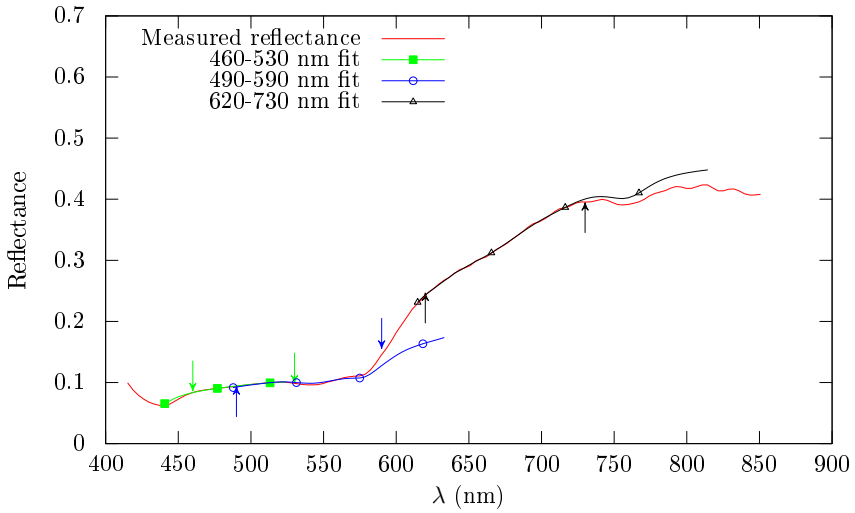


Figure 4.43: The use of pheomelanin for a detected pheomelanin region. Isotropic diffusion model on line 76, pixel 228. See table A.1 for parameters.

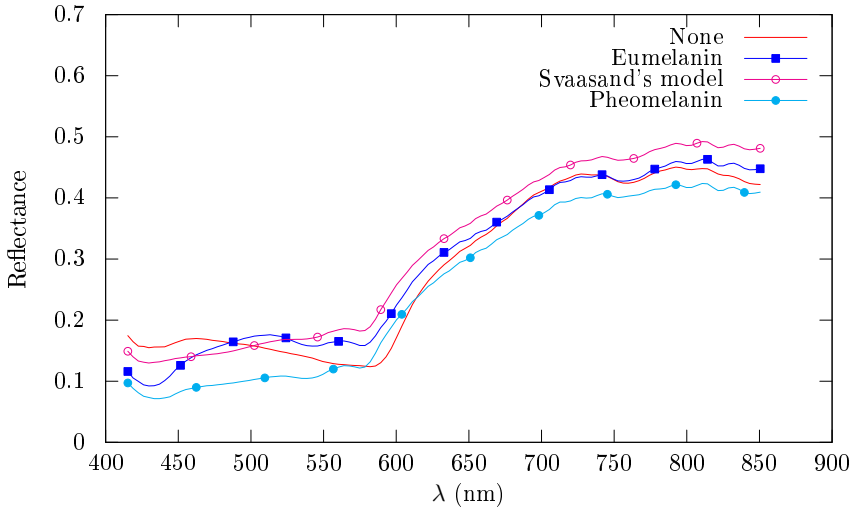
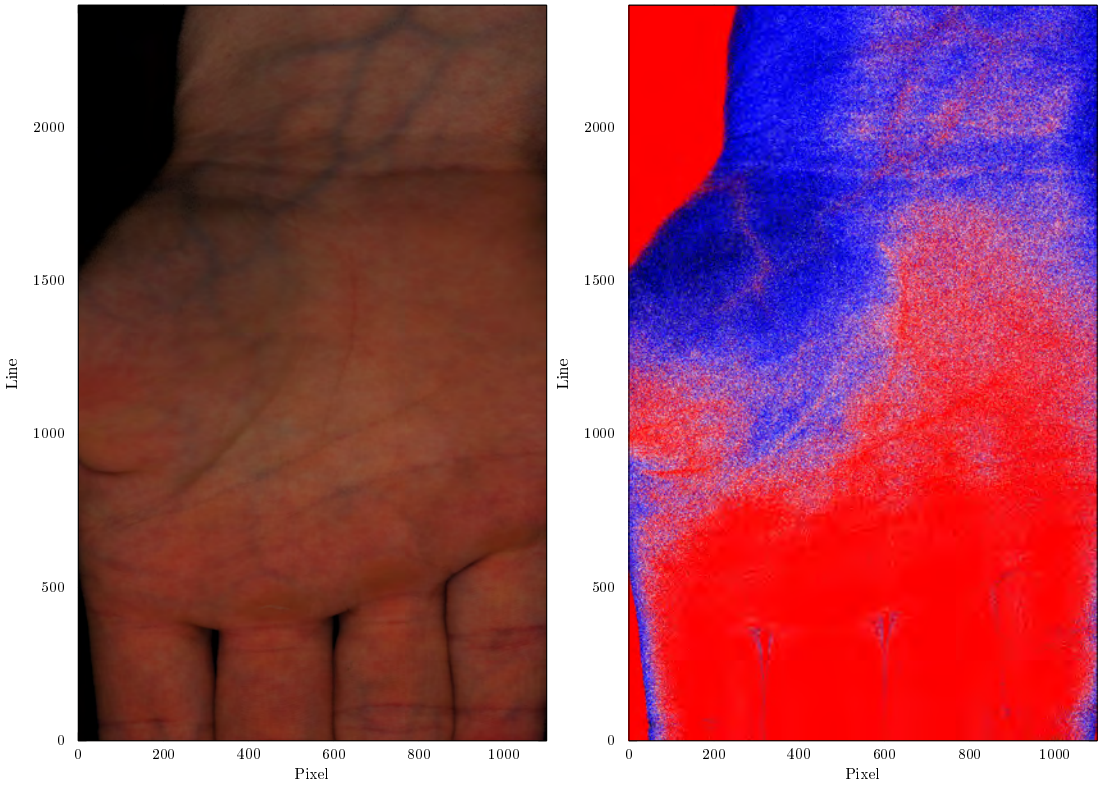


Figure 4.44: Extracted spectra from different melanin type regions.

question.

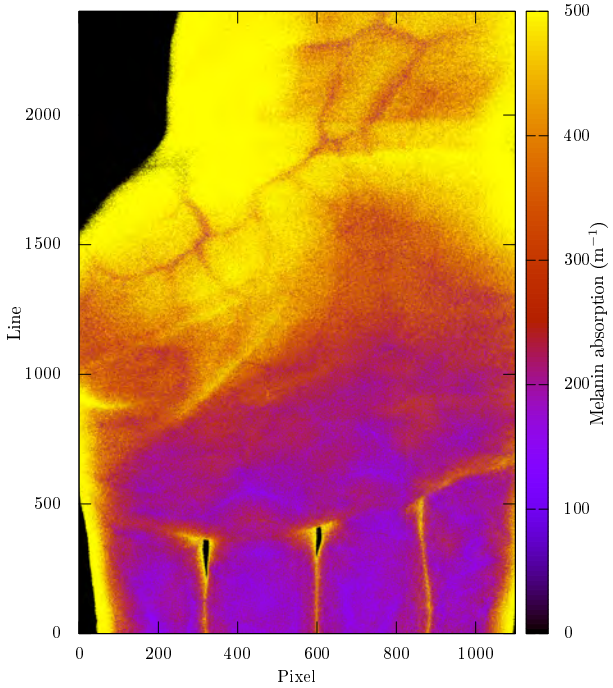
Whether pheomelanin actually is warranted is difficult to say, as the image is difficult. It is possible for different melanin types to be present in the same type of skin [102], but localization and heightening of different melanin types in different situations does not seem to have been investigated much. Whether this is the cause here is not verifiable.

In figure 4.45 is the derived melanin coefficient, melanin type and RGB image shown for a hyperspectral image taken of the hand of a red-haired individual with freckles. This image was taken as part of an arthritis trial. Only the fingers and the upper parts of the hand were therefore in focus during the image scan, as the autofocus system was not in use. The melanin type, where the image is in focus, is correctly detected as pheomelanin. As the image gets more blurry is the melanin type switched over to eumelanin. This is also seen at the boundaries of the fingers and where the hand lies partially in shadow. The melanin here seems to be overestimated. The eumelanin requirement can therefore be seen as a measure to alleviate the too high melanin estimate due to the image not being in focus. It should also be noted that the melanin is determined from the longer wavelengths. Here are the penetration depths high enough for the light to penetrate all throughout the fingers.



(a) The RGB image reconstructed from a hyperspectral image.

(b) The melanin type in figure 4.45a. Red is pheomelanin, white is Svaasand's model, blue is eumelanin and black is neither.



(c) The melanin absorption in figure 4.45a. Note the difference in scaling as compared to earlier melanin images.

Figure 4.45: RGB image, associated melanin absorption values and apparent melanin types for an individual with Fitzpatrick skin type I or II.

The eumelanin requirement for the former hyperspectral image was primarily at the edges of the image. Mostly will only the central parts of the image be in focus as the leg is curved. The same explanation can, technically, therefore be used for the former hyperspectral image. The melanin is overestimated due to being out of focus, and the eumelanin is needed to compensate. This still does not verify why pheomelanin is warranted for the hyperpigmented areas rather than the ordinary Svaasand model, but Svaasand's model and the pheomelanin model are in any case close. Differentiating between the two will be sensitive to other factors.

The reconstructed RGB image from a hyperspectral image of the underarm of an individual with Fitzpatrick skin type IV is shown in figure 4.46a, the melanin map in 4.46c and the melanin type in 4.46b. The melanin coefficient is correct for such an individual by comparing to tanned variants of white skin [61], but the melanin type is not. When Svaasand's model is forced as the melanin type will the spectrum give appearances of the melanin being underestimated. An example is shown in figure 4.47. The method will therefore try to use pheomelanin, though here it is obvious that this melanin type is not correct at least in its pure form. On the other hand, the melanin is apparently underestimated at the shorter wavelengths, while it for the longer wavelengths has a perfect match with no obvious overestimation of any kind. The underestimation can also be discerned only at close examination.

Therefore may the pheomelanin misdetection be attributed to some calibration error, as the different wavelengths warrant different melanin absorptions. It may also be that the threshold is too sensitive.

In conclusion can the results of the melanin type detection for the most cases be attributed to erroneous calibration and to melanin underestimation. The chosen method of observing the scaling of the blood absorption peaks will also be sensitive to errors in the scattering assumption. It just is not reliable. It will however cause the spectra to be fittable, but whether the parameters yielded are correct remains to be seen.

Melanin index

The melanin method has problems, but it will characterize the relative variations of the melanin content well enough. The melanin index has been said to do the same thing. Here will it be shown that despite this will the developed melanin method still output a better estimate of the relative variations than the melanin index once did.

In figure 4.55 is the melanin map from a wound shown, along with the melanin index, RGB image and melanin type. The method is able to correctly estimate the melanin to be low in the wound, although it is not set to zero as it should. This is not because of any crosstalk but rather because the initial melanin is set to 100 m^{-1} and to move further downwards will be difficult.

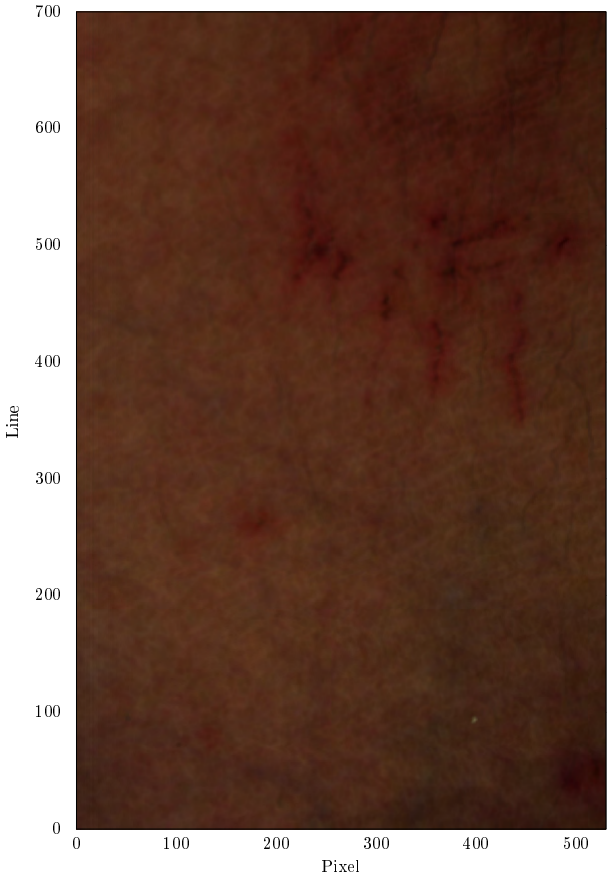
The corresponding melanin index is shown in figure 4.48d. This also characterizes the melanin variations within the image well enough, but the index values themselves are off. A melanin index map is shown in figure 4.49, for the hyperspectral image represented by figure 4.45a. The melanin index values in the wound should be comparable or less than the values in figure 4.49. They are not. The absolute values yielded by the proposed melanin method does therefore fare better than the melanin index ever would. It should be noted that the melanin absorption at 694 nm and the melanin index are strictly not directly comparable, but comparing melanin index against melanin index should be correct.

4.4.2 Parameter verification

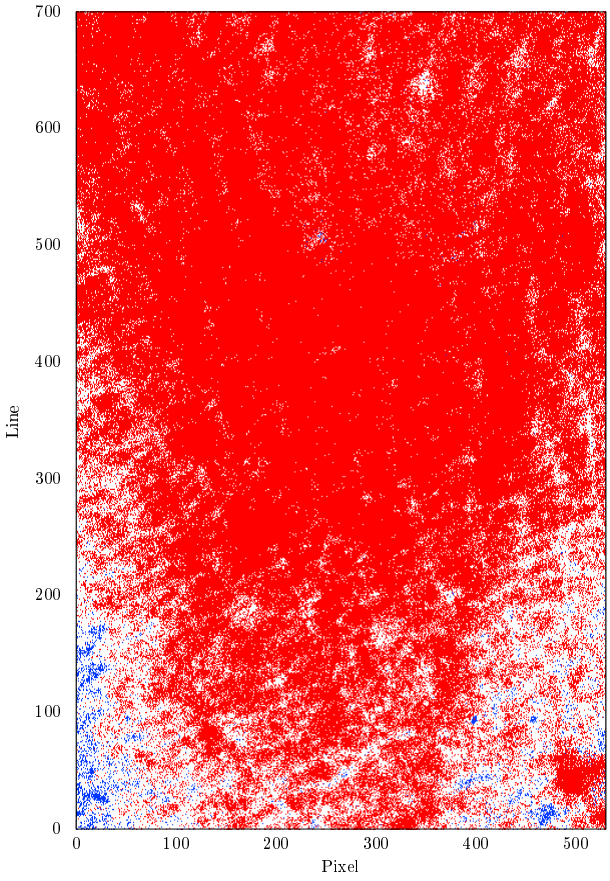
Here the other parameters than the melanin content are verified.

The parameters extracted from the image shown in figure 4.19b is shown in figure 4.50. Some of the parameters extracted when the melanin type is free to vary is shown in figure 4.51.

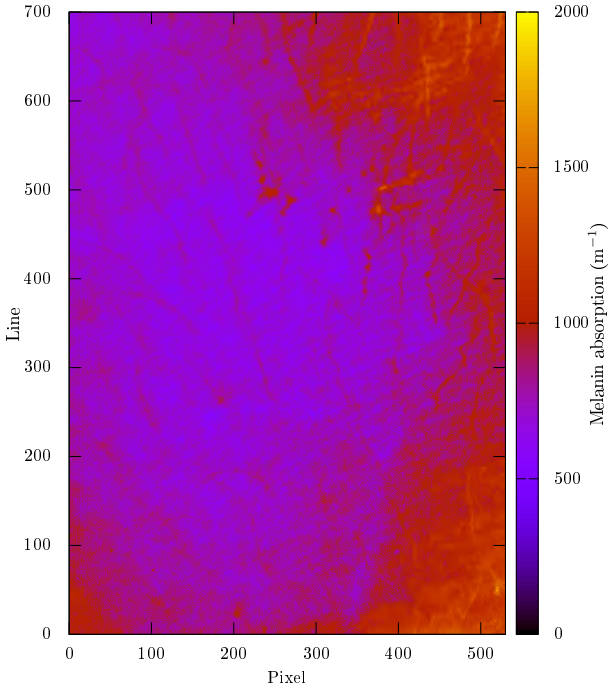
When the melanin is free to vary is it apparent that there will be extra artifacts in the oxygenation and blood volume fraction maps as compared to when the melanin type is constant, but other than the few artifacts do the relative variations within each image correspond.



(a) The reconstructed RGB image.



(b) The apparent melanin type. Red is pheomelanin, white is Svaasand's melanin model, blue is eumelanin and black would be neither.



(c) The melanin absorption.

Figure 4.46: Melanin absorption, apparent melanin type and RGB image from the hyperspectral image taken from the under arm on an individual with Fitzpatrick skin type IV.

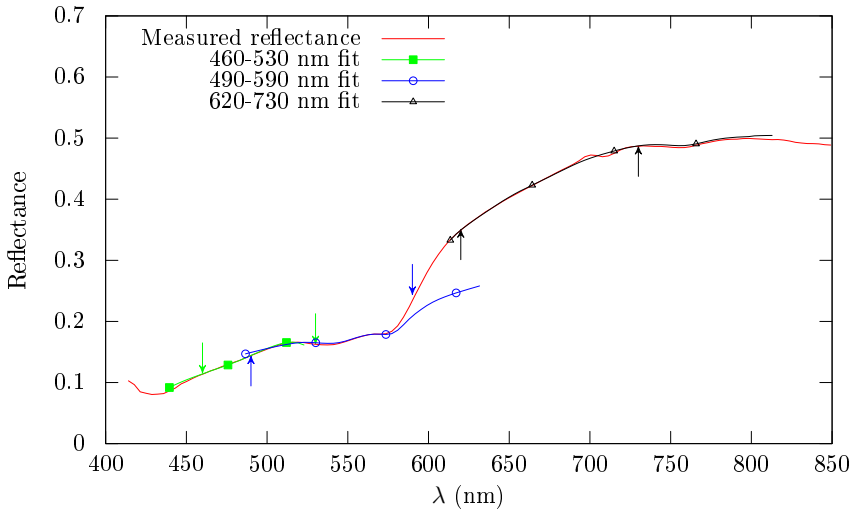


Figure 4.47: The isotropic diffusion model fit for the spectrum located at line 344, pixel 260 in figure 4.46a when Svaasand's melanin model is used. Parameters in table A.1.

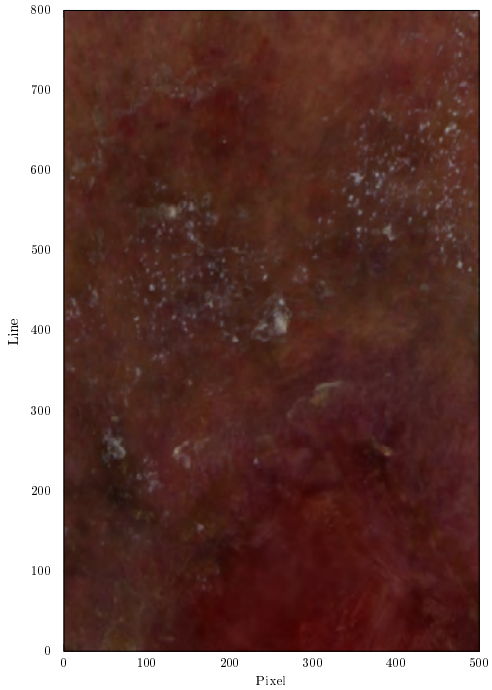
The oxygenation values for the shorter wavelengths correlate somewhat with heightened melanin. This can be due to underestimation of melanin, the diffusion model no longer being correct (similar behavior will later be seen for simulations) or a high oxygenation being warranted due to the hyperpigmentation. Other than the hyperpigmented parts are the oxygenation values mostly within the physical values, with the oxygenation being somewhat lower for the shorter wavelengths and somewhat higher for the longer wavelengths. This is to be expected due to differences in vascularization in the deeper and more superficial layers [53]. The parameters will be discussed more in a later section.

Above fits were all done using SCA. SCA's accuracy has been called into question in an earlier section, and the method was also run using the ordinary Lawson-Hanson algorithm. The results are shown in figure 4.52. It is easily seen that there is a difference, but both images characterize approximately the same relative differences. SCA yields slightly higher values and Lawson-Hanson is more prone to failure in infeasible areas. The difference is discernible, but not large. Still, there is a difference and an uncertainty, and SCA should not be used for this unmixing later. The use of SCA can also explain some of the discrepancies, but not all. SCA will still yield a relatively good fit with the error being set low. Still is this no proof that SCA always will fare this well.

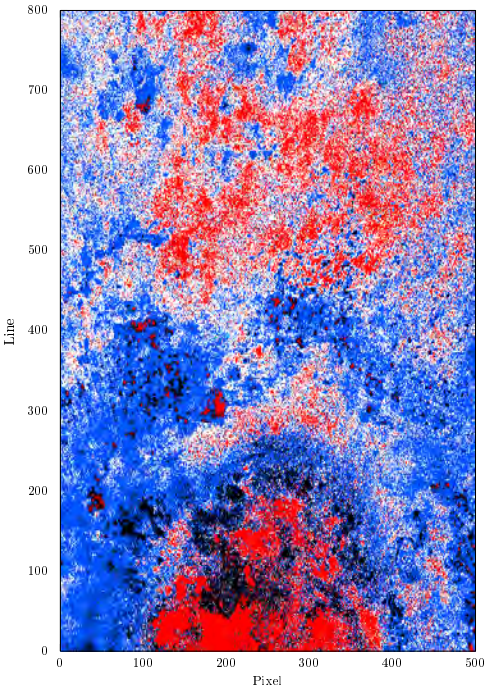
The parameters for the image shown in 4.45a are shown in figure 4.53. The oxygenation values are very high, but this is, after all, in the palm of a hand. High oxygenation is to be expected. The blood volume fraction is overestimated when the image goes out of focus or lies partially in shadow. This correlates with the melanin overestimation. Same behavior could also be seen earlier in figure 4.50, with a very high blood volume fraction at the edges. This is incidentally where the melanin type was chosen to be eumelanin.

The parameters from the image shown in figure 4.46a are shown in 4.54. The oxygenation is extremely low in areas, likely because of the image being out of focus or lying partially in shadow. The image was obtained before autofocus was developed. The bruise is detected correctly using the shorter wavelengths, and the longer wavelengths detect the blood vessels lying beneath the more shallow bruise.

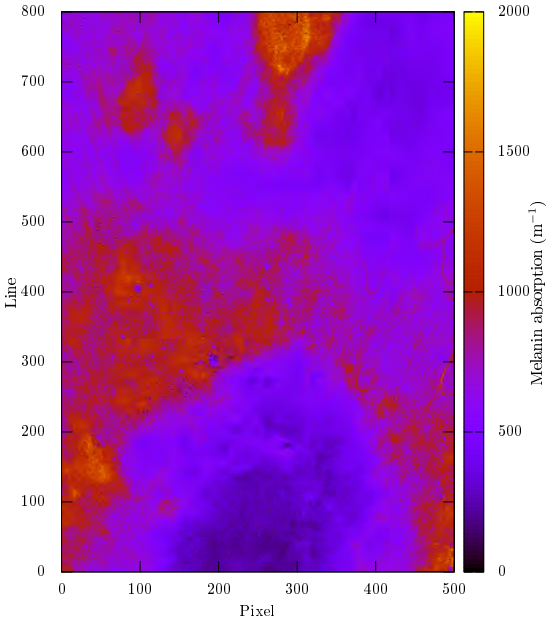
In figure 4.55 are the parameters for the image shown in 4.48a shown. The oxygenation values for the normal skin are within physicality. The blood volume fractions correlate, likely because of the penetration depth not being deep enough, as is also seen for the other images. The blood volume fraction and the oxygen saturation both have problems in the wound for the shorter wavelengths, which can be attributed to either an erroneous scattering assumption or the melanin not being set to zero.



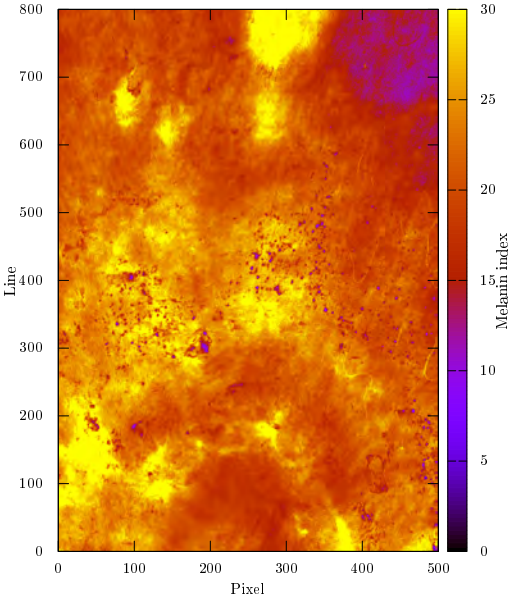
(a) The reconstructed RGB image of a hyperspectral image of a wound.



(b) The melanin type in 4.48a. Red is pheomelanin, white is Svaasand's melanin, blue is eumelanin and black is none.



(c) The melanin absorption in 4.48a.



(d) The melanin index for figure 4.48a.

Figure 4.48: Comparison between the results of the chosen melanin method and the melanin index.

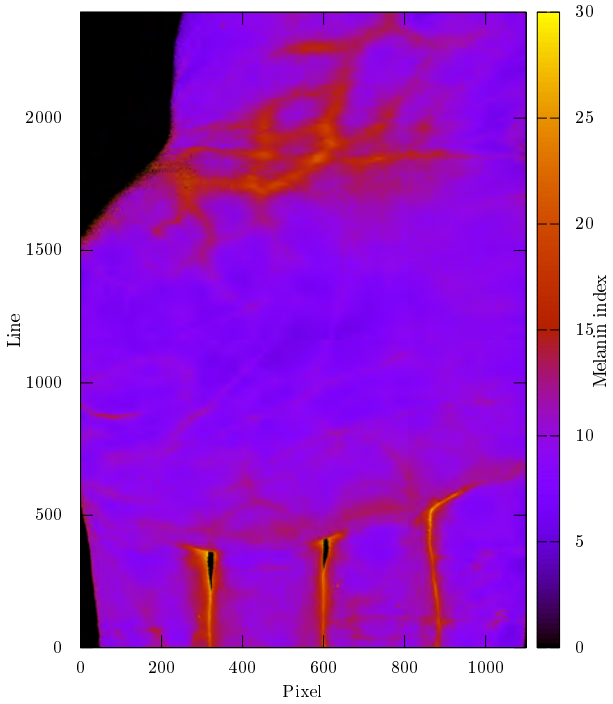


Figure 4.49: The melanin index for figure 4.45a.

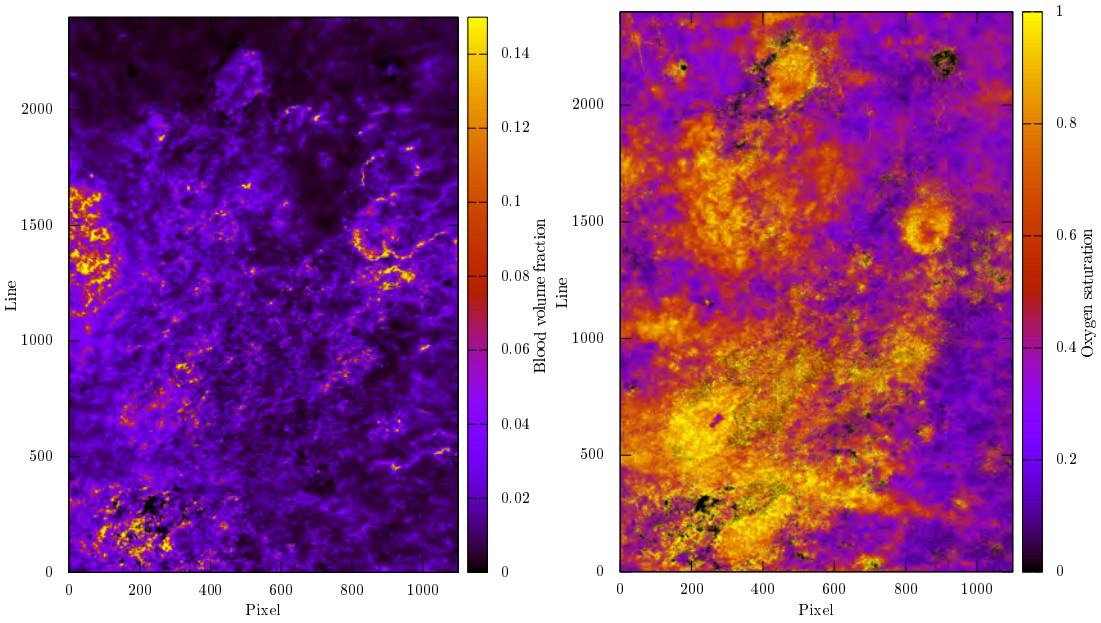
4.4.3 Choice of wavelength interval

Referring back to the parameters displayed in figure 4.50. The blood parameters for a longer wavelength interval, 690-820 nm, is shown in figure 4.56, in contrast to the earlier chosen interval. It has earlier been mentioned that the original wavelength interval for the longer wavelengths might not be optimal. This will now be investigated.

Two wavelength ranges have been tested for the longer wavelengths. For the upper interval are the arteries more distinct than the lower interval. This will be because smaller details are washed away by the meaning process throughout the penetration depth. The gross features like larger blood vessels become more apparent. The oxygenation values between the two wavelength intervals match to a certain extent, although some of the blood vessels for the 620-730 nm in 4.50 have a very low oxygenation as compared to the surrounding tissue (see line 2500). This can be attributed to the melanin being set too low, but at the same time is a corresponding high blood volume fraction not seen in the bvf map. Also the upper wavelength range has a lower oxygenation in the blood vessels than the surrounding tissue in figure 4.56, though not as extreme. The question is whether this is warranted.

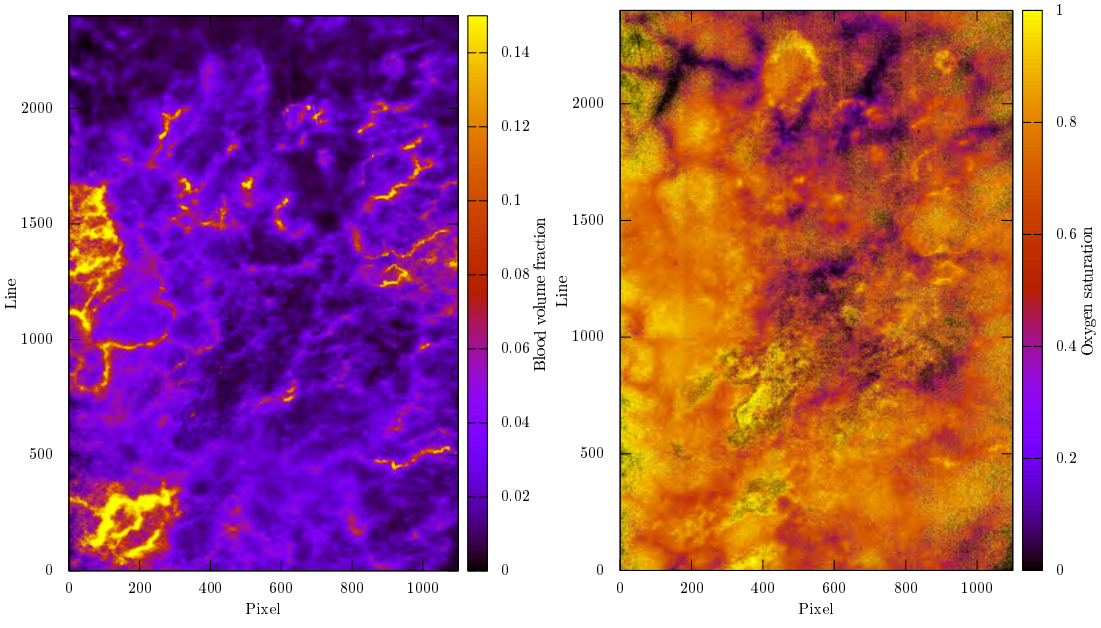
The deoxy dip is far more obvious in the spectrum warranting a low oxygenation (see figure 4.57b) than in the spectrum warranting a higher oxygenation (see figure 4.57a), even though they have a different melanin absorption. There are also no obvious melanin underestimation signs, the model is able to fit the spectrum with no overcompensation of other chromophores. Overall does the fit from the longer wavelengths in figure 4.57c seem to be able to fit more wavelengths outside its scope than the counterpart fitting slightly shorter wavelengths in 4.57b. This can be a sign that the penetration depth is varying too much. It is seen that the oxygenation at the even shorter wavelengths is very low as almost no spectral characteristics from oxy hemoglobin is seen. This might be able to influence the fit at the intermediate wavelengths more than at the longer wavelengths as the penetration depth is lower. Therefore will the fit at the longer wavelengths represent a better estimate of the "second layer" since the other estimate is too influenced by the upper layers to be of much use. In any case does a lower oxygenation in this blood vessel seem to be warranted judging from the prominent deoxy absorption maximum.

Blood vessels are prominent in the blood volume fraction map for the longer wavelengths. They do however vary in intensity. Especially can it be seen that the intensity is lowered as the blood vessel



(a) The blood volume fraction derived from 500-590 nm.

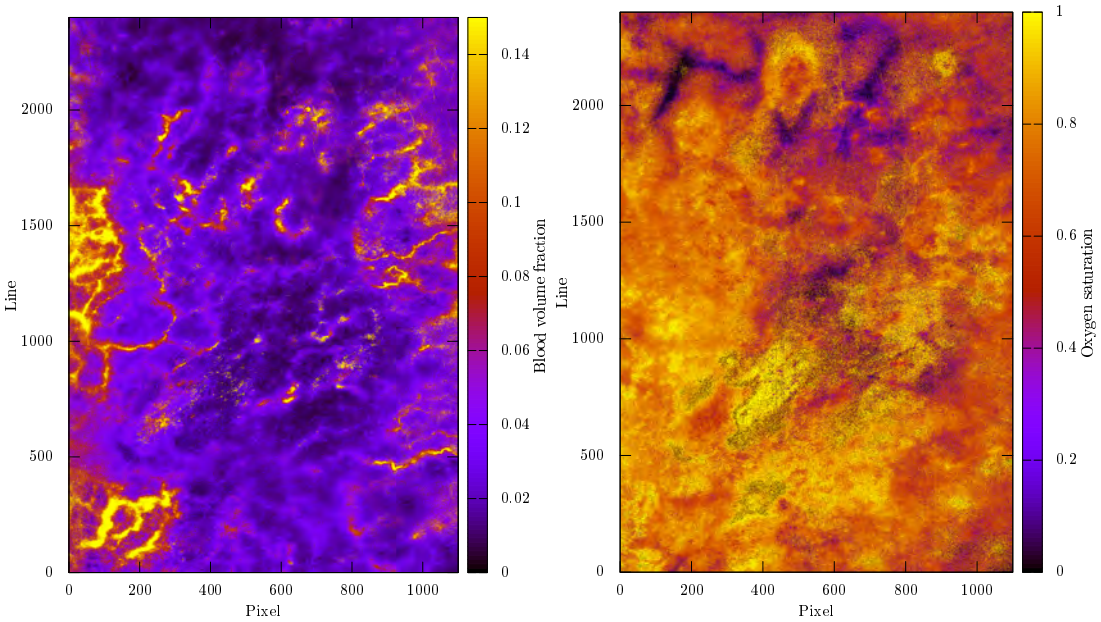
(b) The oxygen saturation derived from 500-590 nm.



(c) The blood volume fraction derived from 620-730 nm.

(d) The oxygen saturation derived from 620-730 nm.

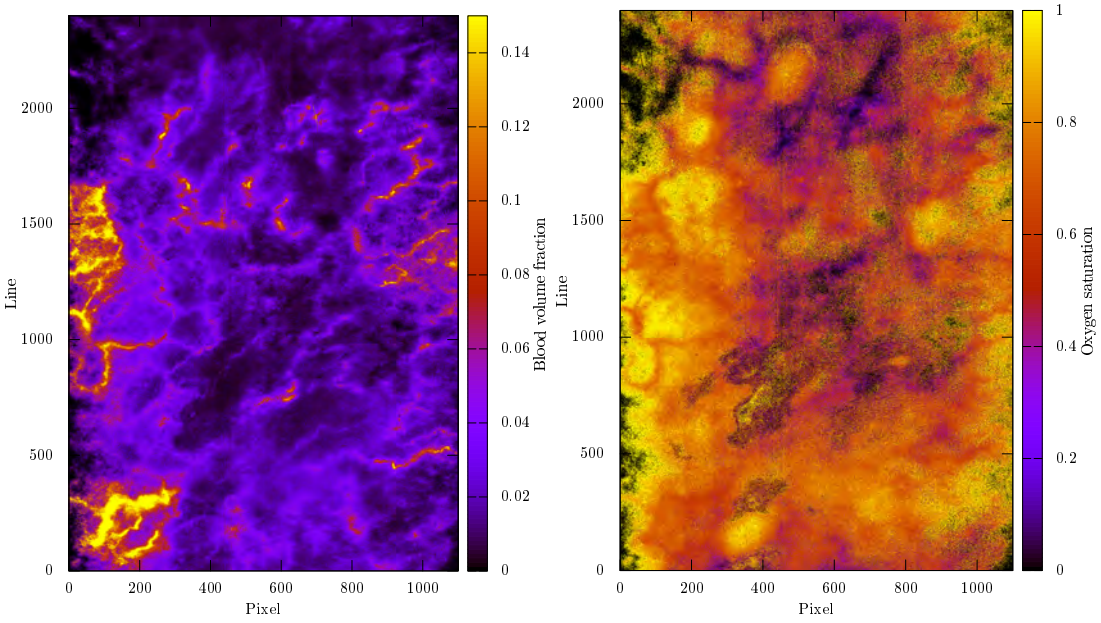
Figure 4.50: Blood parameters for the hyperspectral image represented by figure 4.19b. Svaasand's melanin model is used as the melanin model.



(a) The blood volume fraction derived from 620-730 nm.

(b) The oxygen saturation derived from 620-730 nm.

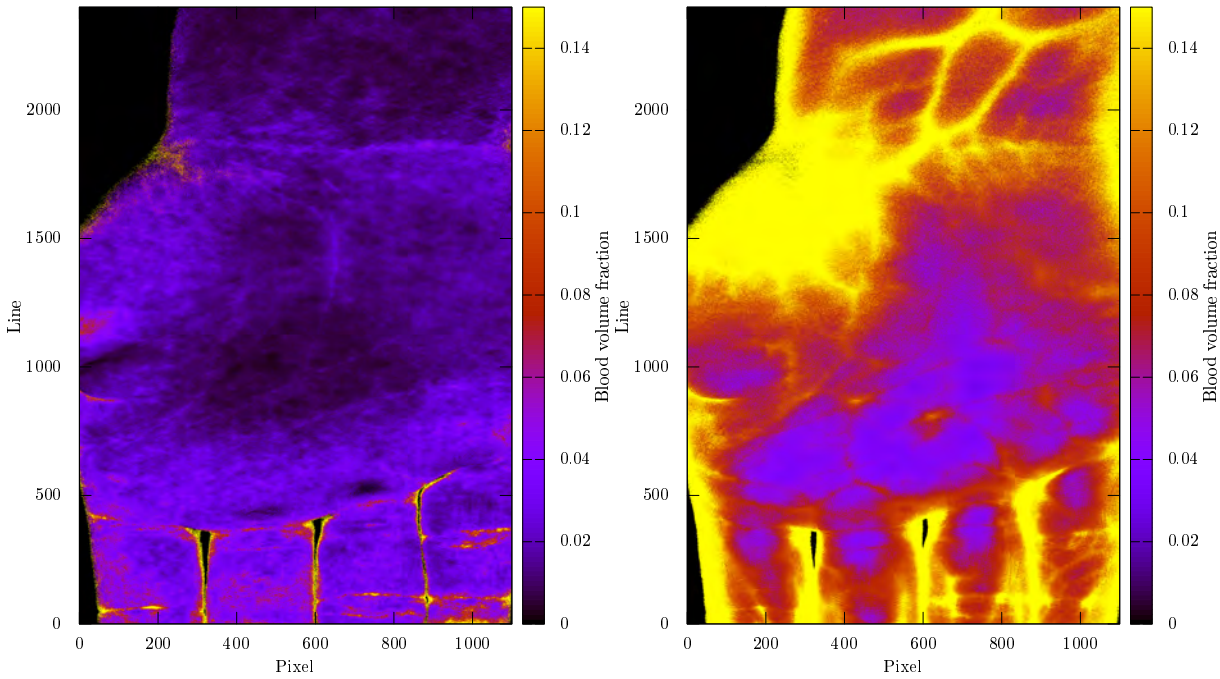
Figure 4.51: Blood parameters for the hyperspectral image represented by figure 4.19b when the melanin type is free to vary spatially.



(a) The blood volume fraction.

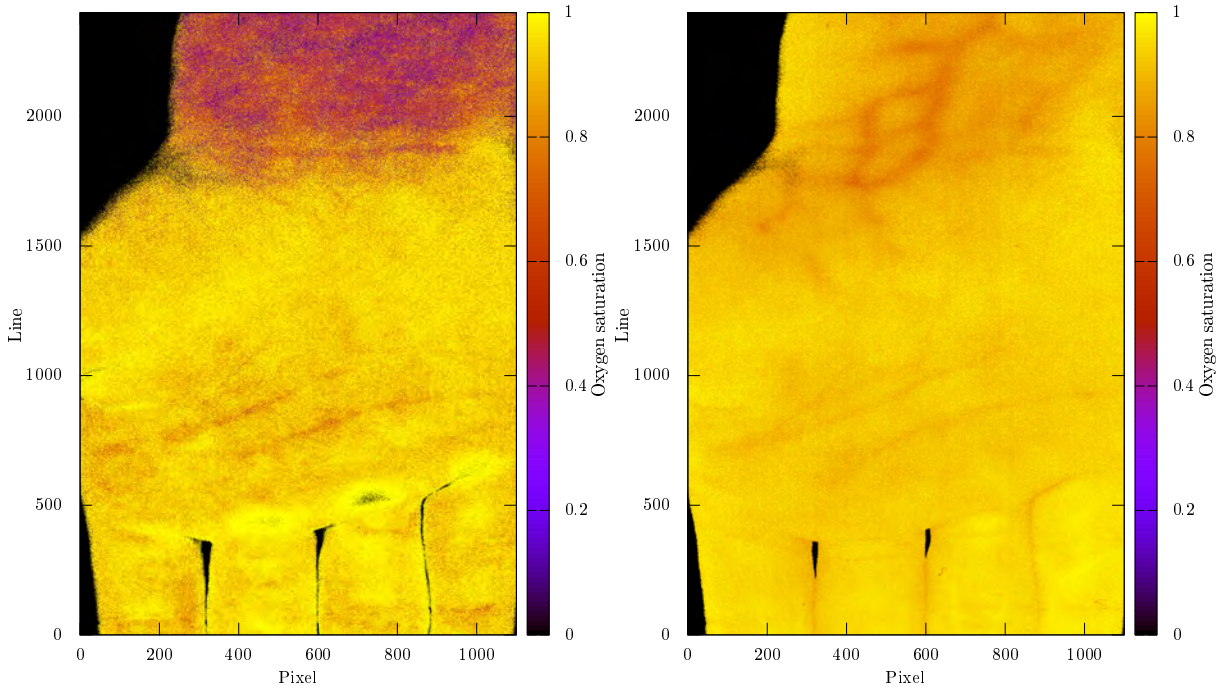
(b) The oxygen saturation.

Figure 4.52: Blood parameters for the hyperspectral image represented by figure 4.19b. Svaasand's melanin model is used, and Lawson-Hanson is being used for the unmixing. The wavelength interval is 620-730 nm.



(a) Blood volume fraction, 500-590 nm.

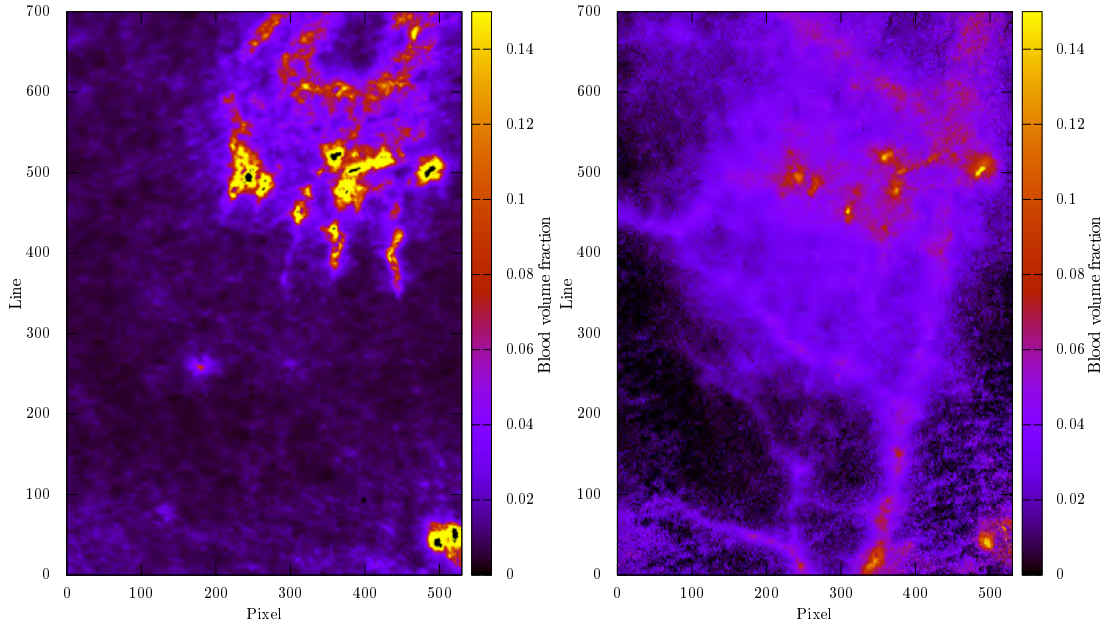
(b) Blood volume fraction, 690-830 nm.



(c) Oxygen saturation, 500-590 nm.

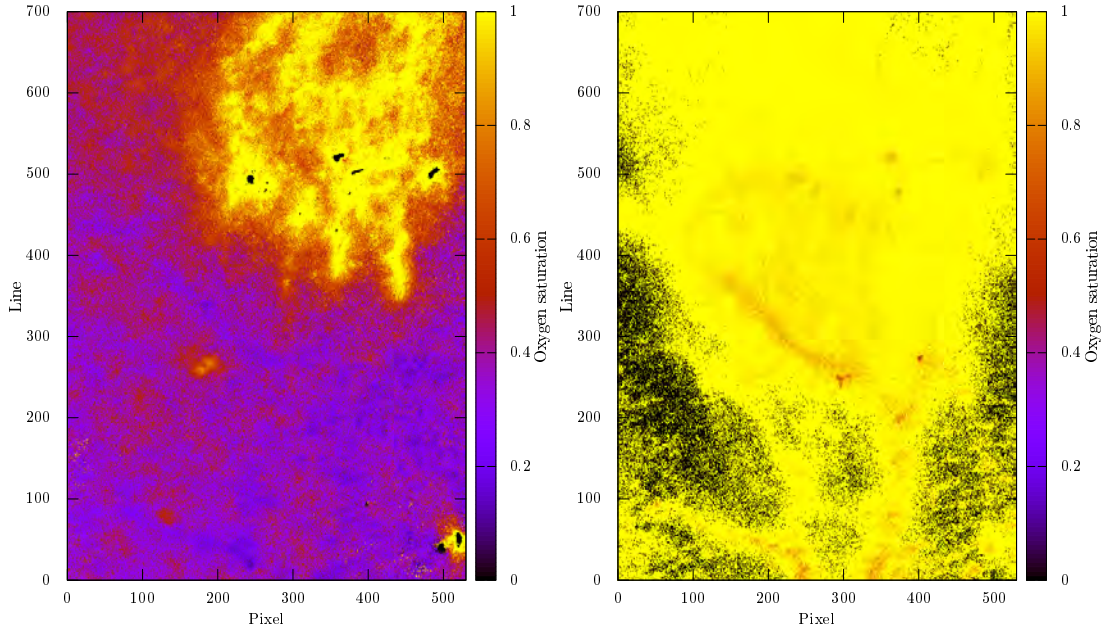
(d) Oxygen saturation, 690-830 nm.

Figure 4.53: Blood parameters for figure 4.45a. The melanin type is free to vary.



(a) Blood volume fraction, 500-590 nm.

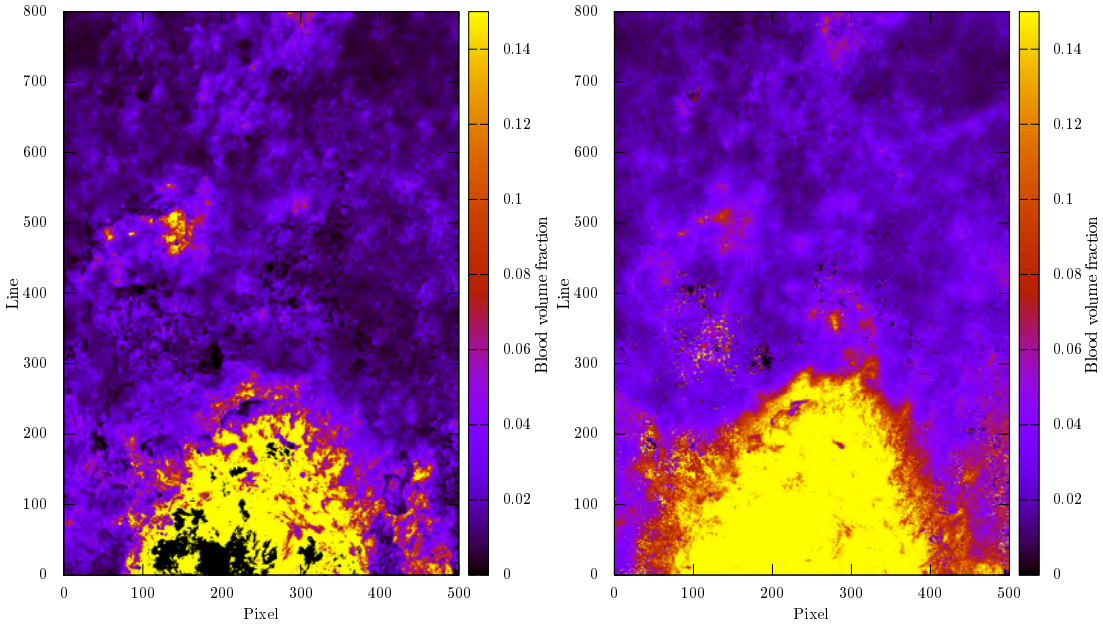
(b) Blood volume fraction, 620-730 nm.



(c) Oxygen saturation, 500-590 nm.

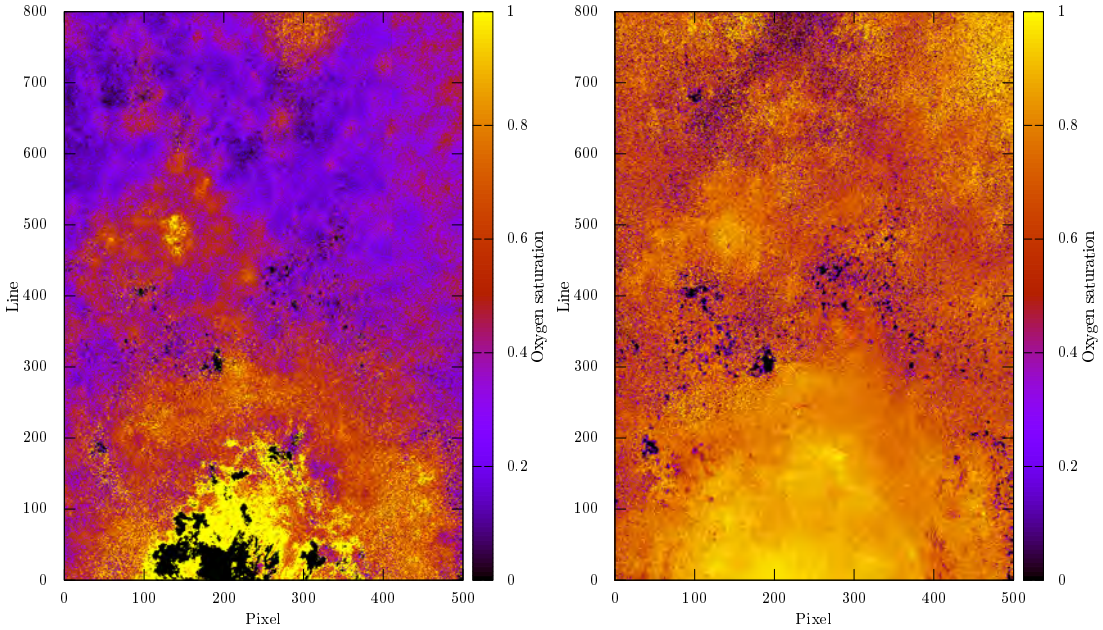
(d) Oxygen saturation, 620-730 nm.

Figure 4.54: Blood parameters for figure 4.46a. The melanin type was free to vary.



(a) Blood volume fraction, 500-590 nm.

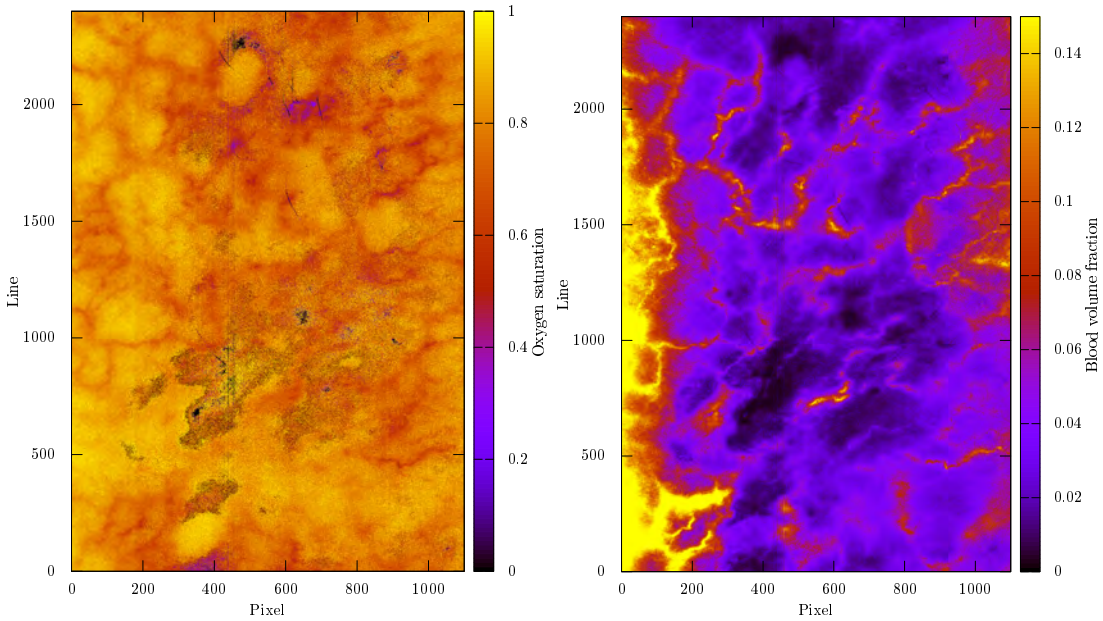
(b) Blood volume fraction, 620-730 nm.



(c) Oxygen saturation, 500-590 nm.

(d) Oxygen saturation, 620-730 nm.

Figure 4.55: Parameters for figure 4.48a.



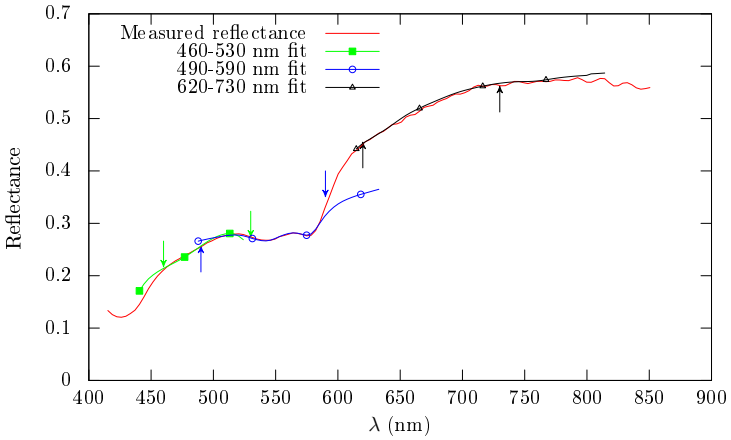
(a) Oxygen saturation.

(b) Blood volume fraction.

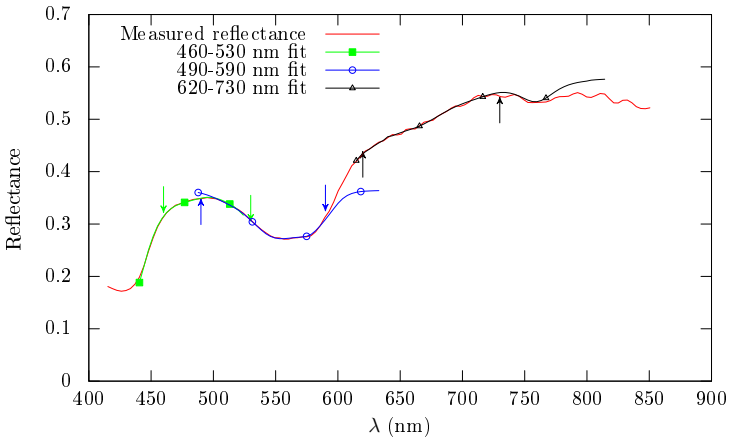
Figure 4.56: Parameters for figure 4.19b when a fitting range using slightly longer wavelengths is used, 690-820 nm instead of the usual 620-730 nm.

enters a hyperpigmented zone. This is no accident, when the melanin is high is the light blocked from reaching down into the lower layers of the skin. The model is a two-layered model designed to "scan" the skin down to two penetration depths, and the information gained will be reduced with an increased, blocking melanin absorption. A map of the penetration depths is shown in 4.58.

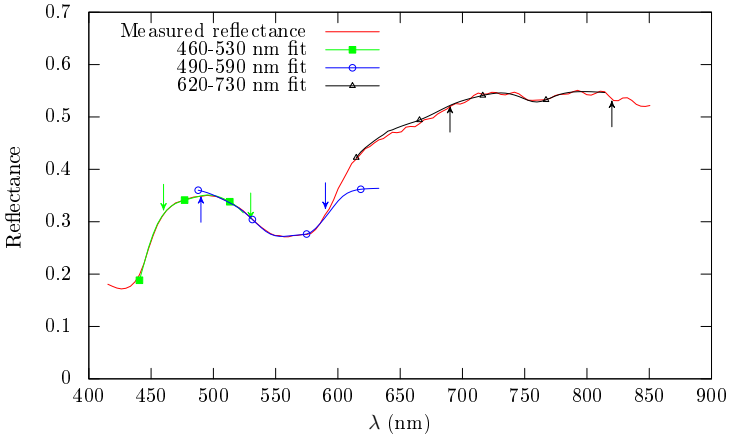
As expected are the penetration depths higher for the higher wavelength interval than for the slightly lower. The difference in the maximum and minimum penetration depths when melanin is not taken into account is shown in the same figure. This will show how well the chromophores may be fit to the dermal absorption. It is seen that the dermal penetration depth will vary the least over the 690-820 nm interval, in accordance with previous findings.



(a) The fit of pixel 228, line 2117 using 620-730 nm for the upper wavelength range. Apparent high oxygenation, $oxy_3 = 0.83$, $bvf_3 = 0.013$.

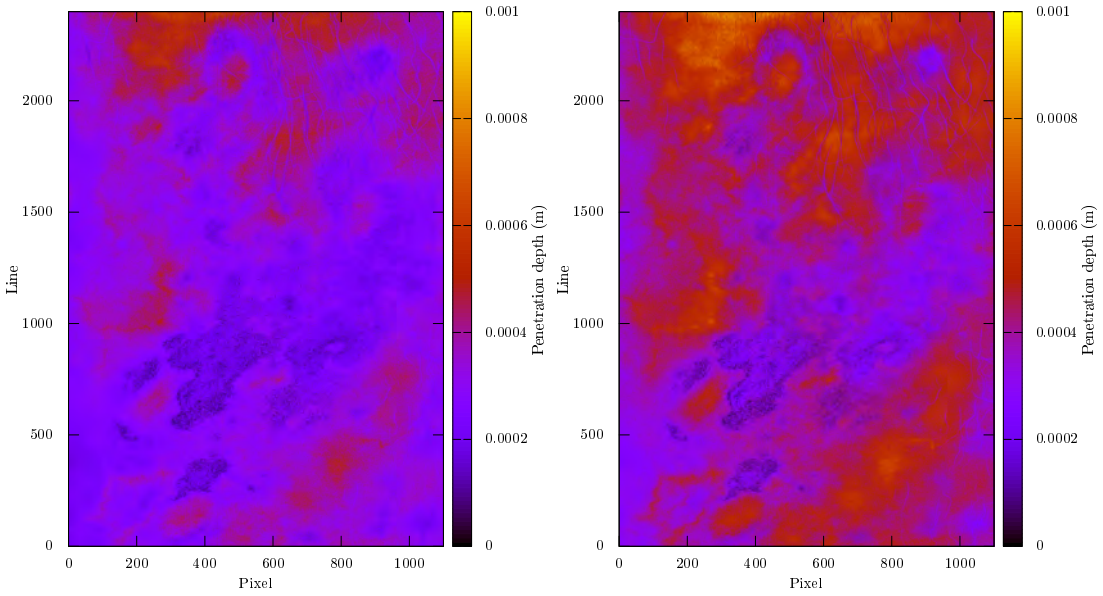


(b) The fit of pixel 274, line 2172 using 620-730 nm for the upper wavelength range. Apparent low oxygenation, $oxy_2 = 0.13$, $oxy_3 = 0.25$.



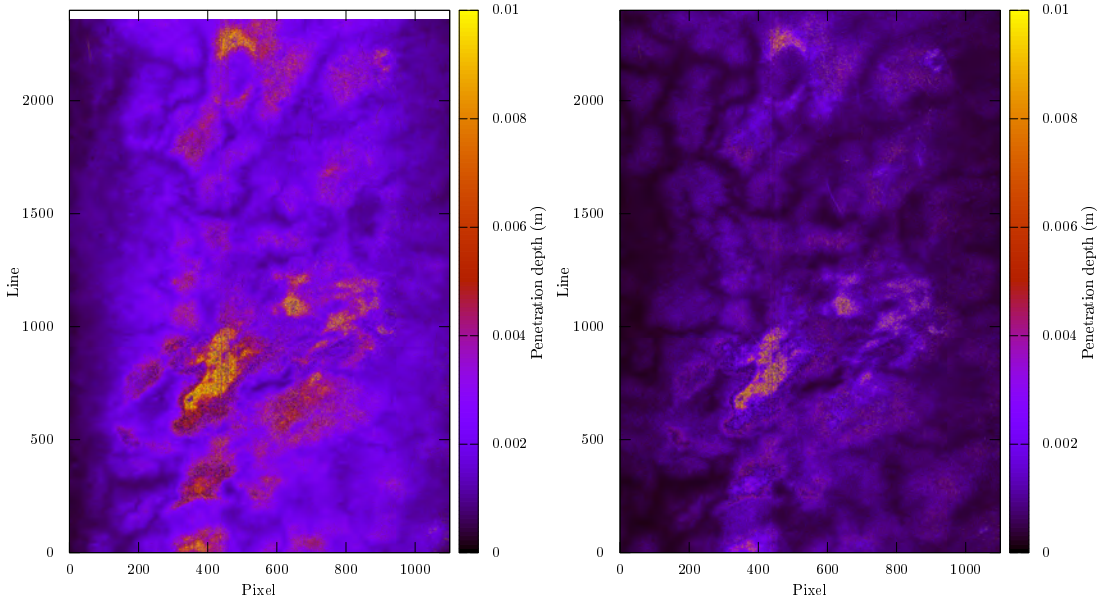
(c) The fit of pixel 274, line 2172 using 690-820 nm as the upper wavelength range. $Bvf_3 = 0.066$, $oxy_3 = 0.62$.

Figure 4.57: Comparison of fitting ranges in one spectrum, and comparison of one spectrum from an apparent high oxygen saturation region and one from an apparent low oxygen saturation region. Figure 4.19b is fitted. Fitting parameters found in table A.1.



(a) Mean penetration depth, 620-730 nm.

(b) Mean penetration depth, 690-820 nm.



(c) Difference between minimum and maximum dermal penetration depth, 620-730 nm.

(d) Difference between minimum and maximum dermal penetration depth, 690-820 nm.

Figure 4.58: Comparison of penetration depths.

4.5 Verification on simulations

In this section will the inverse model be verified on simulations. The simulations will be simple situations of the three- or two-layered skin model. The success on real data cannot be evaluated by the simulations, but it is still of some concern since an inverse method not giving back the same parameters as input into the forward model will not be very useful.

In addition will the diffusion model inverse approach be verified on Monte Carlo simulations, and the two-layered approach as applied on a three-layered reality will be investigated. Some comparison between the isotropic source function and Delta-Eddington will also be done.

4.5.1 Verification using the two-layered diffusion model

Here is the inverse, two-layered diffusion approach verified on spectra forward-modelled by the two-layered diffusion model.

There are three different ways of fitting the melanin to the epidermal absorption coefficient. It has been seen that if the melanin curve is used in the fitting can this lead to overestimation problems. In figure 4.60 is the melanin curve used throughout the fitting, while it in figure 4.61 is used only in the last iteration. In figure 4.59 is straight line fitting used throughout all iterations. For all of these were all parameters first fitted to dermis, and all fitted parameters except for melanin set in dermis before deriving the required epidermal absorption to bridge the missing dermal melanin gap. A different approach to the dermal fitting is to fit all parameters but not set the dermal absorption to the fitted parameters, only remove the apparent dermal melanin component from dermis and then derive the epidermal absorption. The result of using this approach on a third iteration of the method is seen in figure 4.62.

The method was expected to and has been verified on real data to underestimate the melanin in the presence of deoxy hemoglobin. This is seen also here, far stronger for higher input melanin coefficients. Straight line fitting will consistently underestimate the results also for higher oxygenations. The results are closer to the input values when the melanin curve is used either in the last or all iterations, though there still is underestimation for low oxygenations and blood volume fractions above 0.05. For the best case will the melanin at worst be underestimated by 150 m^{-1} at an input melanin coefficient of 1500 m^{-1} and extreme input parameters. This will give an error of 10%. For the less extreme parameters is the error far less. Force moving the melanin from dermis to epidermis in the model does apparently lessen the underestimation, but overestimation also becomes more severe. The error here is, though, less than 10%.

There is also some small overestimation even when a straight line is used in the epidermal fitting. There are three reasons for this, none of which are deoxy crosstalk. The error is most easily spotted when the input melanin absorption is 100 m^{-1}, as this is the initial value for the melanin in the inverse model and for which the parameters should be the most correct. The output melanin absorption is plotted as a function of input oxygenation and input blood volume fraction in figures 4.64 and 4.63, respectively. The reasons for overestimation are

- A constant of $\mu_{a,o} = 25$ m^{-1} is set in both dermis and epidermis, multiplied by $1 - B_d$ or $1 - B_e$. This will lead to a slight melanin overestimation when either the blood volume fraction is not absolutely correctly estimated or the constant is set freely in the fitting. The melanin must bridge the remaining absorption gap, and an absorption in the order of 25 m^{-1} smeared across a semi-infinite layer will be far larger when confined to a thin slice at the top of the skin model.
- When the melanin is fitted to epidermis are no special measures taken to correct for the blood and constant absorption set in epidermis in the simulation. These will therefore be fitted by the melanin. Figure 4.64 shows a slight difference in the melanin with respect to the oxygenation set in epidermis.
- Scattering errors. For these simulations were g_{ery} set to zero to be able to compare MCML directly against the diffusion model. This will lead to a far higher blood scattering, and absorption misfitting when the scattering is not correctly estimated from the beginning on. Overestimation

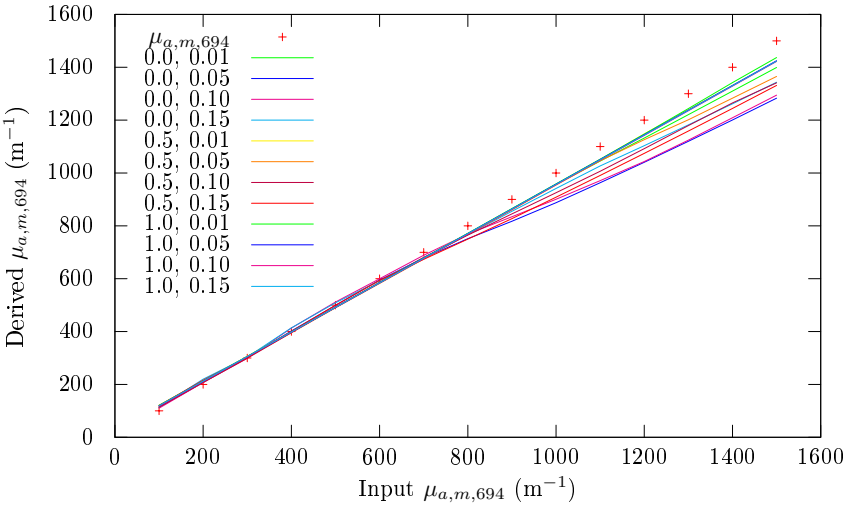


Figure 4.59: Derived melanin, using the isotropic diffusion inverse model on the isotropic diffusion forward model, melanin method using the 730-830 nm interval and a straight line in the epidermal fitting. Lines are titled by "oxy, bvf".

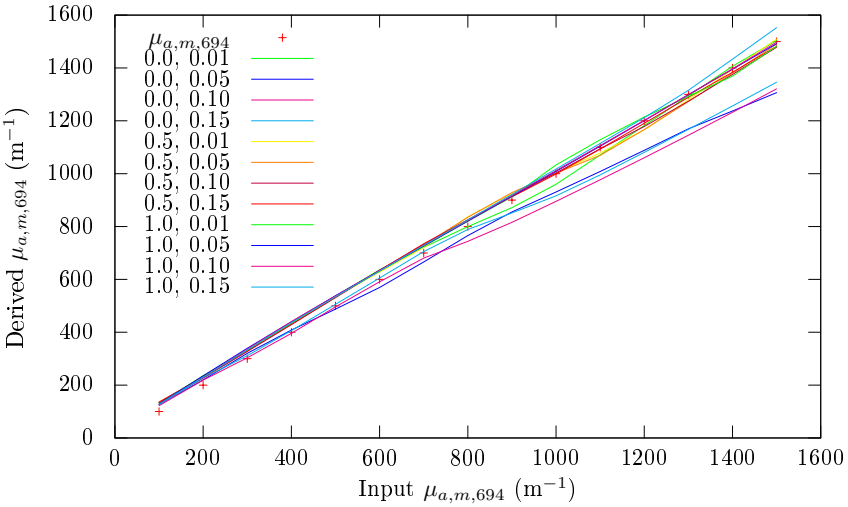


Figure 4.60: Derived melanin, using the isotropic diffusion inverse model on the isotropic diffusion forward model, melanin method using the 730-830 nm interval and the melanin curve in the epidermal fitting. Lines are titled by "oxy, bvf".

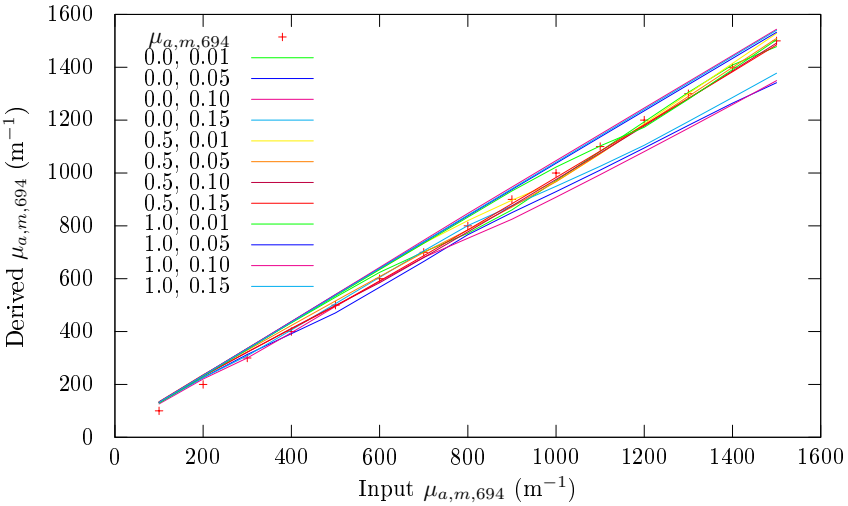


Figure 4.61: Derived melanin using the isotropic diffusion inverse model on the isotropic diffusion forward model, melanin method using the 730-830 nm interval. Linear fitting was used at the first iteration, the melanin curve in the second. Lines are titled by "oxy, bvf".

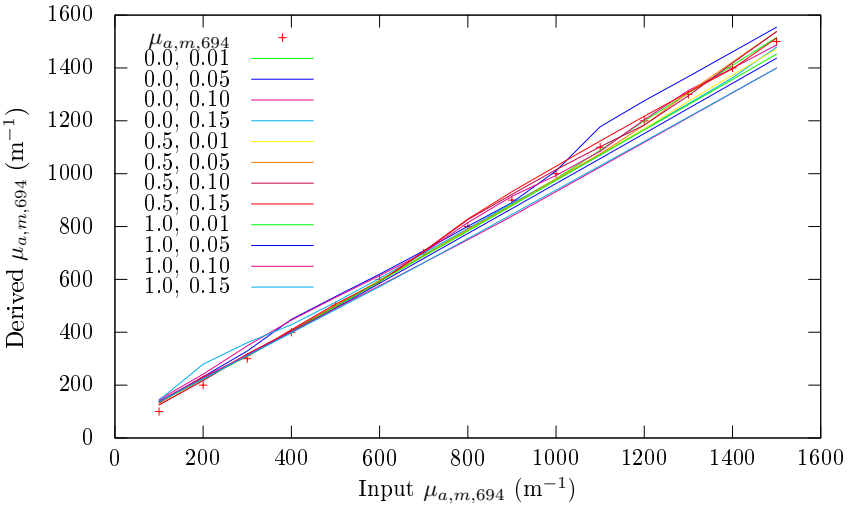


Figure 4.62: Derived melanin, using the isotropic diffusion inverse model on the isotropic diffusion forward model, output melanin as function of the input melanin. Three iterations of the melanin method, the fitted melanin in dermis is forced to epidermis. Lines are titled by "oxy, bvf".

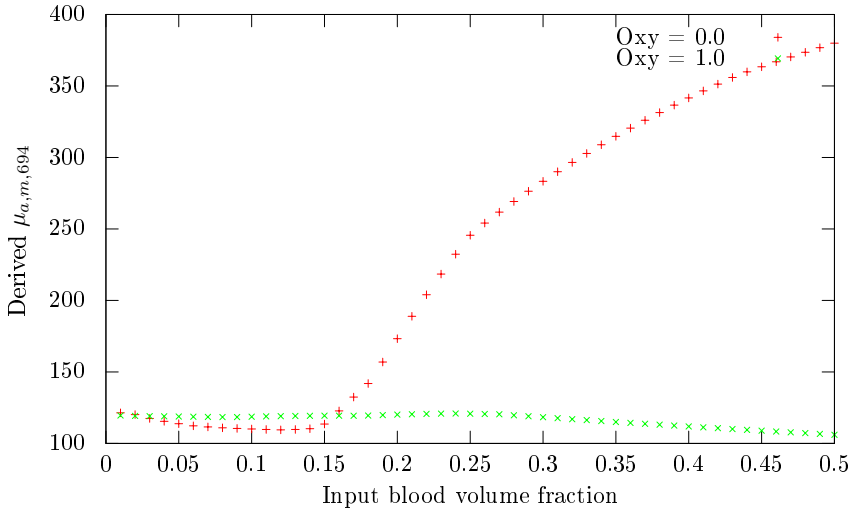


Figure 4.63: The inverted $\mu_{a,m,694}$ as a function of input blood volume fraction for input $\mu_{a,m,694} = 100$ m^{-1} for different blood oxygen saturations using the isotropic diffusion model.

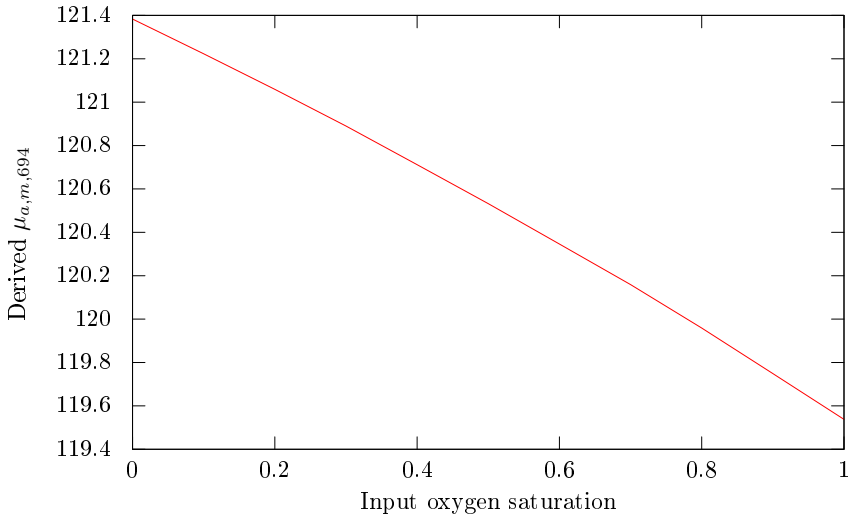


Figure 4.64: The inverted $\mu_{a,m,694}$ as a function of input oxygen saturation for input $\mu_{a,m,694} = 100$ m^{-1}, using the isotropic diffusion model.

will therefore become worse with increasing input blood volume fraction, as seen in figure 4.63. In all of these simulations were the fitted dermal melanin correctly set to 0. Due to the scattering misfitting is deoxy hemoglobin underestimated and carried over to epidermis, where the melanin is overestimated. This happens even though the initial melanin is set to the correct melanin.

Even if the method seemingly fails for the simplest case, this is acceptable. The scattering is set far lower for real data. Even if this should happen for real data will it be recognizable in the intermediate epidermal fitting as it will display the deoxy hemoglobin maximum at 760 nm.

4.5.2 Verification using the two-layered Monte Carlo model

Here is the diffusion inverse approach verified in a similar way against simulated Monte Carlo spectra.

A comparison between the forward simulation of the isotropic diffusion model, Delta-Eddington diffusion model and Monte Carlo model for a set of input parameters is shown in figure 4.65. It has been seen

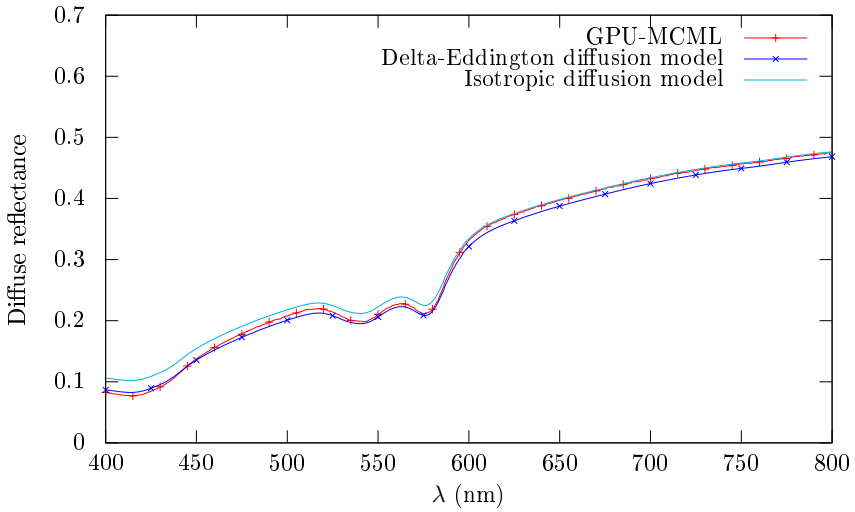


Figure 4.65: Comparison of GPU-MCML and diffusion model for oxy = 0.8, bvf = 0.01, $\mu_{a,m,694} = 700$ m^{-1}. The data used in this figure is taken from [10].

earlier [77, 10, 92] that Monte Carlo and Delta-Eddington will correspond better than the isotropic diffusion model for high absorption values, but this is a relation that will sometimes tend to break down. The former figure is an example of this. For low melanin coefficients should Delta-Eddington and the isotropic source function output the same melanin coefficients. For higher melanin coefficients is Delta-Eddington expected to underestimate, since Delta-Eddington already lies below MCML for the same set of input parameters.

It is seen in figure 4.66 that the method will consistently underestimate the results when straight line fitting is used, as it did earlier, though even worse. Now will the offshoot be in the order of 300 m^{-1} for an input melanin coefficient of 1500 m^{-1}. The melanin curve is used instead of straight line fitting in the last iteration in figure 4.67, which helps on the underestimation. With much deoxy hemoglobin present is the melanin still underestimated, however, but only with an undershoot of about 150 m^{-1} for an input melanin coefficient of 1500 m^{-1}.

When Delta-Eddington is used in the inverse method in figure 4.68, melanin curve used in the last fitting, is the underestimation worse not only for low oxygenation, and in the same order of magnitude as for the straight line fitting. This is expected, as was argued above.

When the dermal melanin is forced into epidermis as previously described will the result seen in figure 4.69 be seen. This is evidently far more unstable, with high overestimation of the melanin parameter. The overestimation is in the order of 300 m^{-1} even for low melanin absorptions at 300 m^{-1}. Technically can the full range of unpredictability be warranted since parts of the isotropic diffusion spectrum will

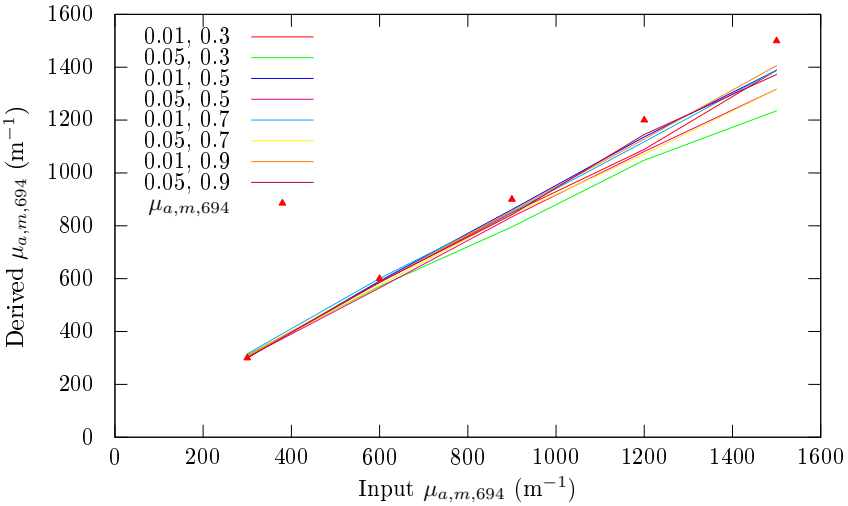


Figure 4.66: The diffusion model melanin output of the Monte Carlo forward model. Straight line fitting was used in the epidermal melanin extraction. Lines are titled by "bvf, oxy".

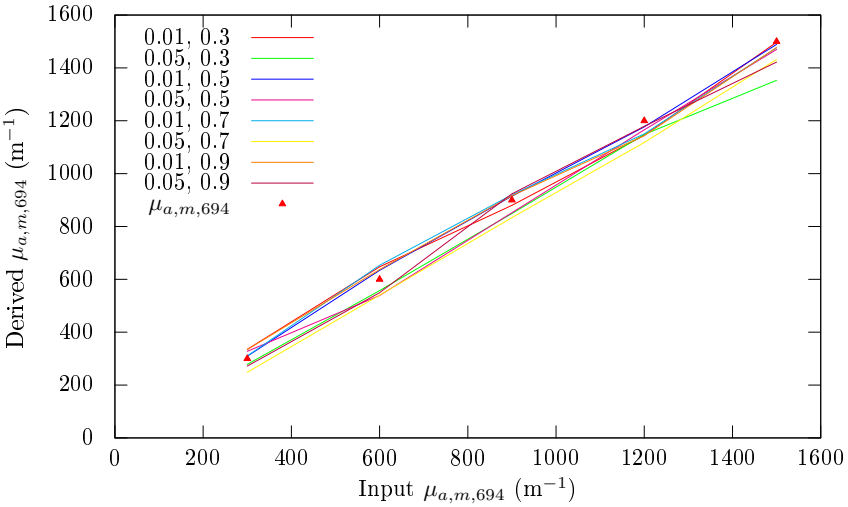


Figure 4.67: The diffusion model melanin output of the Monte Carlo forward model. Fitting using the melanin curve was used in the last epidermal iteration. Lines are titled by "bvf, oxy".

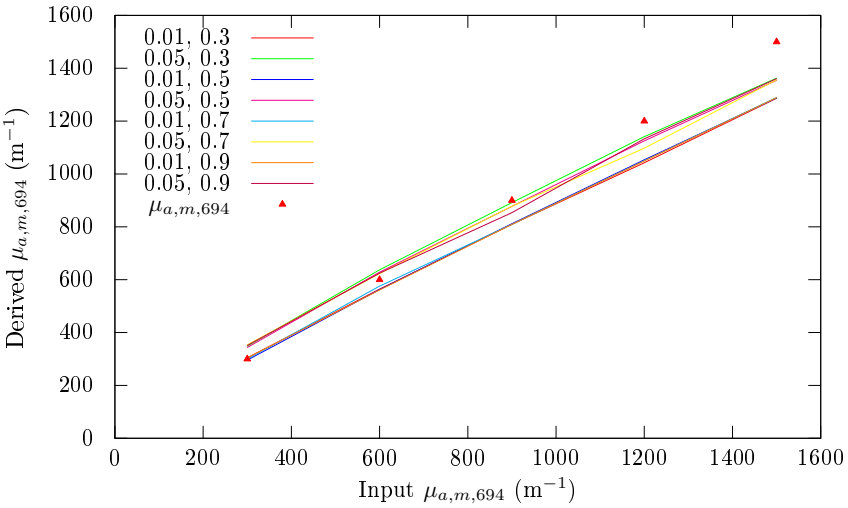


Figure 4.68: The Delta-Eddington diffusion model melanin output of the Monte Carlo forward model. Fitting using the melanin curve was used in the last epidermal iteration. Lines are titled by "bvf, oxy".

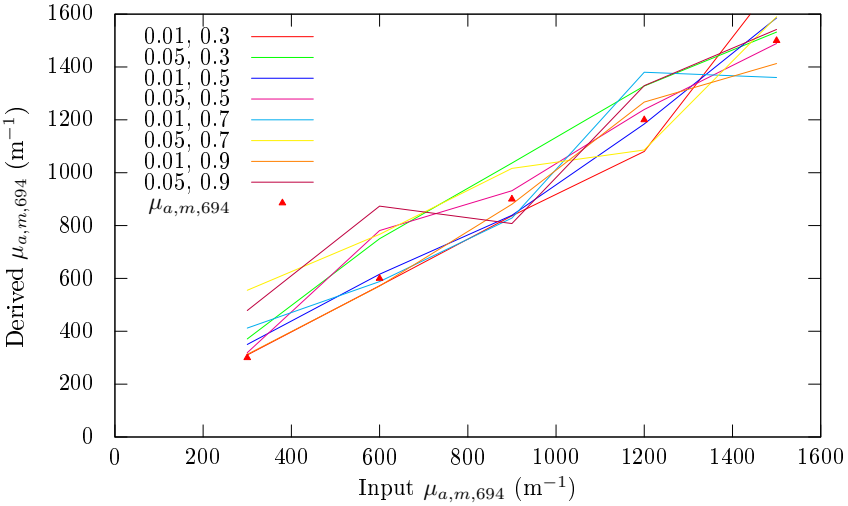


Figure 4.69: The inversion of the two-layered MCML model using the isotropic diffusion model, output melanin as function of the input melanin. Three iterations of the melanin method, the fitted melanin in dermis is forced to epidermis. Lines are titled by "bvf, oxy".

require a higher melanin absorption than the input melanin absorption in the forward model, as seen in figure 4.70. Still, for the wavelength range from which the melanin is extracted is the diffusion model and MCML equal for these melanin absorption coefficients, as seen in figures 4.65 and 4.78, and the behavior should not be seen.

Throughout this thesis has it been assumed that two iterations of the method is enough. In figures 4.71, 4.72 and 4.73 is the convergence as a function of iterations of the full method shown. Using straight line fitting in the first iteration before using the melanin curve in figure 4.71 will be stable and converge itself to an underestimated result about 30 m^{-1} under the desired result. The results are far more unpredictable if the derived dermal melanin absorption is forced to epidermis. In figure 4.73 is it seen that the first iteration will yield an overestimated result, though the subsequent straight line fittings will move the result downwards. This will be due to the straight line fitting's nature of underestimating the results. The results diverge if the melanin curve is used, as in 4.73.

From this will it be apparent that the strategy of force moving the melanin content to epidermis will give unpredictable results. Two iterations of the method is enough, and fitting using straight line fitting in the first iteration to avoid overestimation and fitting using the melanin curve in the next iterations to avoid too much underestimation will give stable convergence.

4.5.3 Verification using the three-layered Monte Carlo model

Here will the other parameters be investigated, when the two-layered diffusion model is applied on a multi-layered reality.

The stability of the derived parameters for increasing melanin absorption coefficients is shown in figure 4.74 for the isotropic diffusion model and 4.75 for Delta-Eddington.

The parameters derived from 500-590 nm are more stable than the parameters derived from the longer wavelength intervals. This is not surprising, since the longer wavelength intervals will use these parameters to compensate for the lack of melanin when this is underestimated. The oxygenation at 530-590 nm lies slightly higher than the input oxygenation in the Monte Carlo model, which is to be expected as it will represent a mean of 0.5 at the upper layer and 0.9 at the lower layer along some penetration depth. The same behavior is seen for the blood volume fraction.

The oxygenation at 500-590 increases with increasing melanin absorption. This is not physical. Increasing melanin content should have shielded the two-layered model from seeing the oxygenation in the lower layers, and made the derived oxygenation converge to 0.5. The same behavior was seen in the estimation of blood parameters for high melanin contents in figure 4.50. If Delta-Eddington instead is used will the oxygenation be more stable, as seen in figure 4.75, though the proper behavior still is not seen. The discrepancy can therefore be explained to be more of a fault lying with the diffusion model.

Derived parameters as a function of the input parameters for the same melanin content is shown in figure 4.76 for isotropic source functions and in figure 4.76 for Delta-Eddington. The derived parameters are seen with respect to the corresponding input parameter (the derived oxygenation at 500-590 nm against the oxygenation of the upper layer), but the other parameters (blood volume fraction, oxygenation at other layers) are varied to yield a standard deviation for the derived parameter. In addition is a mean of the same parameter derived for a different wavelength range (700 nm and up) calculated to show how much this will be influenced.

The range 500-590 nm will be able to characterize the relative differences of the parameters set in the uppermost dermal layer. Extracting the parameters from 690-800 nm or 620-730 nm will similarly characterize the relative differences, though with a larger standard deviation especially for low input oxygenation. This will be due to melanin underestimation. The range 620-730 outputs a larger standard deviation than the range 690-800 nm. This will be due to its shorter penetration depth, being more influenced by the parameters in the upper layer. Especially is it seen that the blood volume fraction here is almost varying as much as the blood volume fraction derived from 530-590 nm with increasing blood volume fraction in the upper layer, leading to large standard deviations in the derived blood volume fraction. Judging from this is 690-800 nm the better choice.

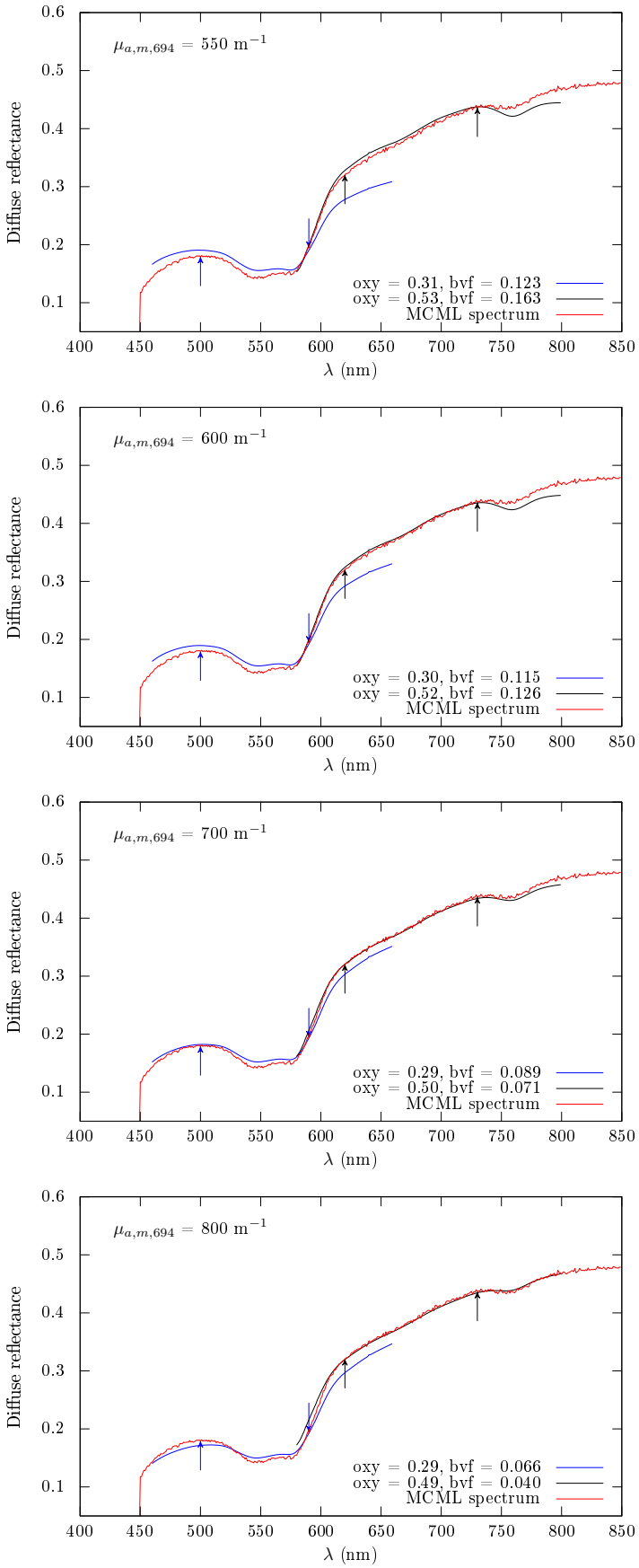


Figure 4.70: Fitting an MCML spectrum using the isotropic diffusion model with different input melanin contents. Input $\mu_{a,m,694}$= 600 m^{-1}, oxy = 0.3 and bvf = 0.05.

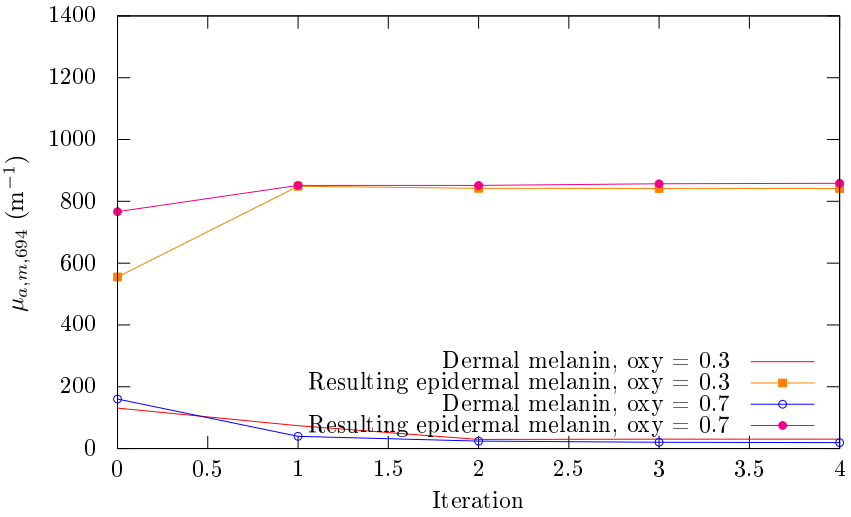


Figure 4.71: Convergence of the melanin, fitting using straight line fitting in the first iteration and the melanin curve in all subsequent iterations. The isotropic diffusion model applied on two-layered MCML spectra. Input melanin was 900 m^{-1}, input blood volume fraction 0.05.

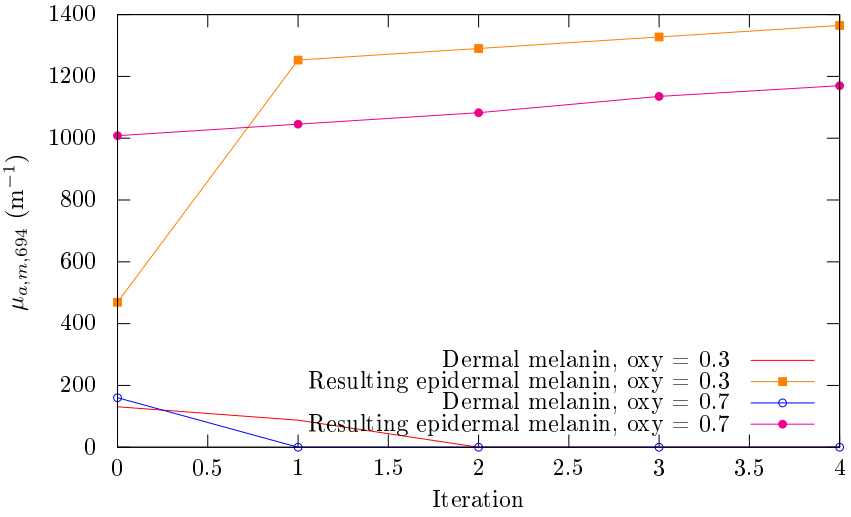


Figure 4.72: Divergence of the melanin. Melanin is forced from dermis to epidermis. Straight line fitting in the first iteration, melanin curve fitting in all subsequent iterations. The isotropic diffusion model is applied on two-layered MCML spectra. Input melanin was 900 m^{-1}, input blood volume fraction 0.05.

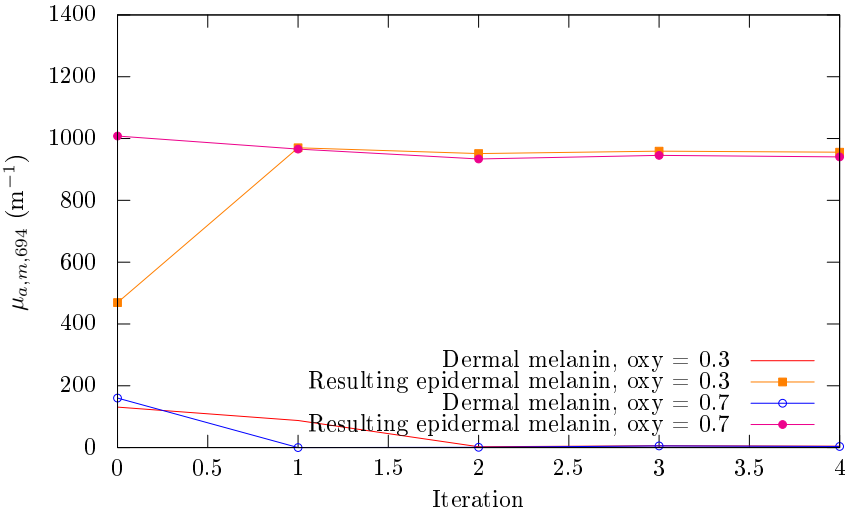


Figure 4.73: Convergence of the melanin. Melanin is forced from dermis to epidermis. Straight line fitting in all iterations. The isotropic diffusion model is applied on two-layered MCML spectra. Input melanin was 900 m^{-1}, input blood volume fraction 0.05.

For Delta-Eddington are the behaviors more or less the same. It is seen, though, that the oxygenation at 690-800 nm has a far larger standard deviation than for the isotropic case. Delta-Eddington had some misfitting with oxygenation values equal to zero, attributed to melanin underestimation.

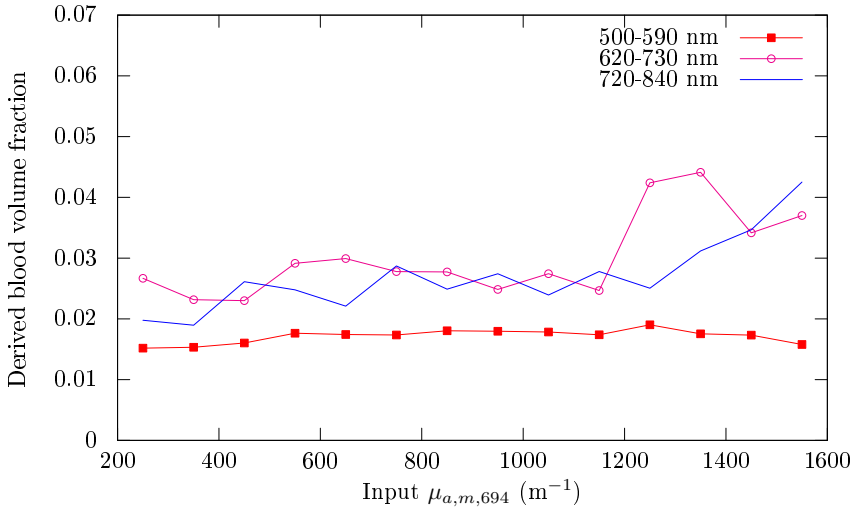
In figure 4.77 are the number of parameters varied reduced. The oxygenation derived from 690-730 nm and 690-800 nm are plotted against the oxygenation at the upper and lower layer and blood volume fraction at the lower layer with the other parameters set constant. Both will characterize the differences in oxygenation in the lower layer well enough, though there is not much difference in the stability with respect to the oxygenation in the upper layer. It is however seen that they both will be influenced by the blood volume fraction in the lower layer, but 620-730 nm more than 690-800 nm.

4.5.4 The effect of blood vessels

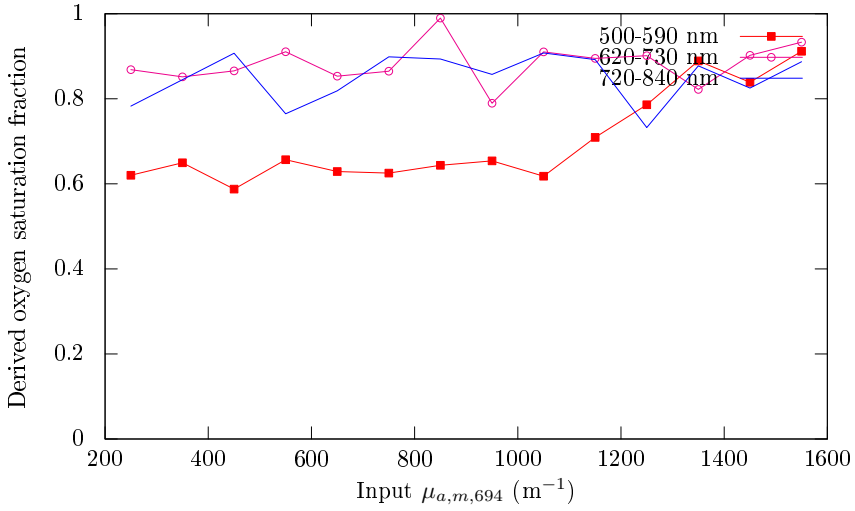
A reflectance spectrum from blood vessels have been approximated in figure 4.78. Blood vessels were simulated assuming a high blood volume fraction at the third layer and assuming a different refraction index of $n = 1.35$ for this layer (value from [114]). This is a crude approximation, as not only will the refraction index be different but also the scattering due to the packing of the red blood cells and scattering from the blood vessel walls, but the change of refraction index should illustrate some of the problems related to wrong boundary conditions.

As long as there is not much blood, the approximation using the diffusion model is expected to be good. When there is much more blood, using the diffusion model to approximate the Monte Carlo model will yield either a lower amount of blood or a lower amount of melanin depending on the spectral features, since the diffusion model spectrum for the same parameters lie lower than the Monte Carlo spectrum. When the boundary conditions are changed, using a refraction index more reminiscent of blood vessels while ignoring the blood vessel walls, the Monte Carlo spectrum lies higher yet and the melanin approximation will be expected to be worse.

Melanin underestimation in the presence of blood vessels is therefore to be expected.

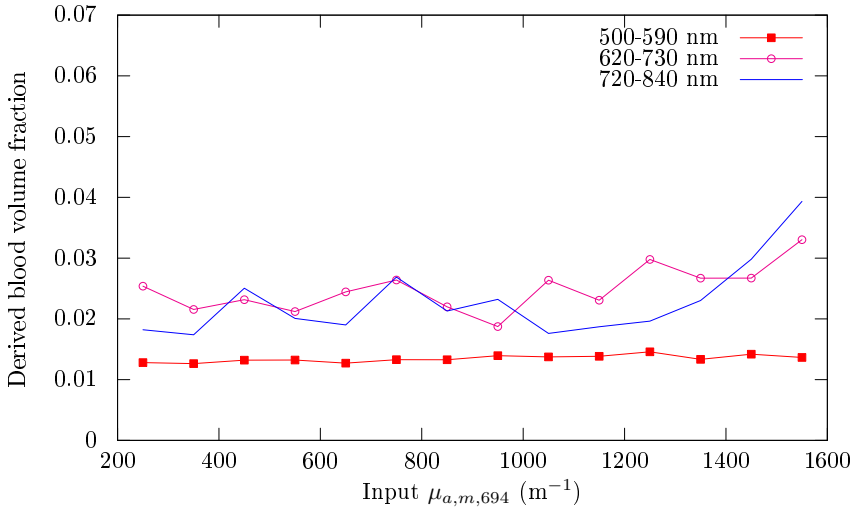


(a)

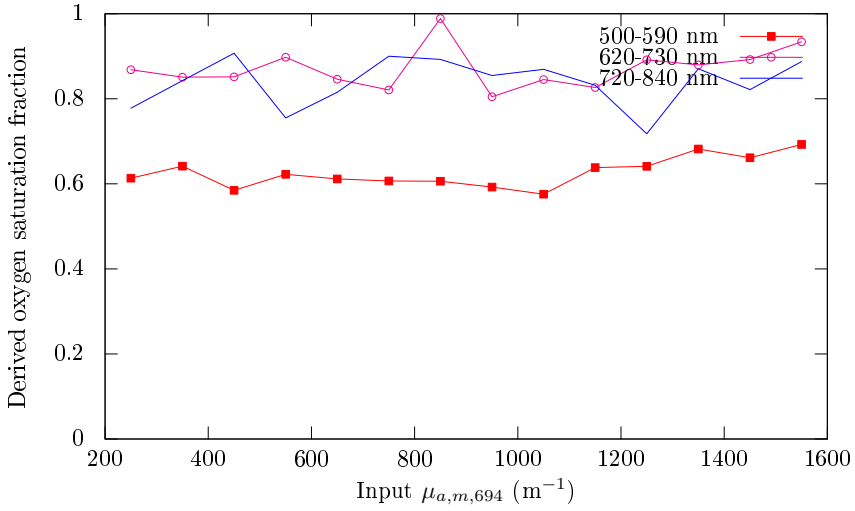


(b)

Figure 4.74: The derived oxygen saturation and blood volume fraction using the inverse isotropic diffusion model on MCML spectra, as a function of the melanin absorption coefficient input to MCML. MCML spectra generated using oxygen saturation 0.5 and 0.9 at the first and second dermal layer respectively, and 0.01 and 0.03 as the blood volume fraction at the same layers. The melanin curve is being used in the epidermal fitting in the last iteration.

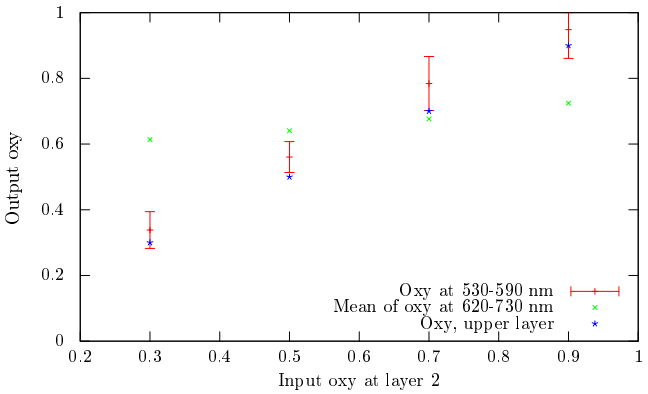


(a)

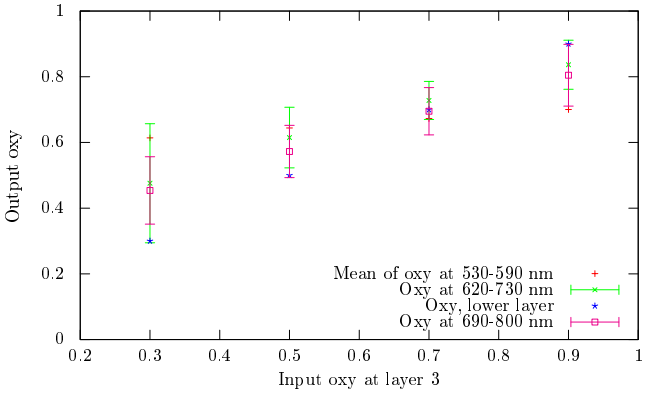


(b)

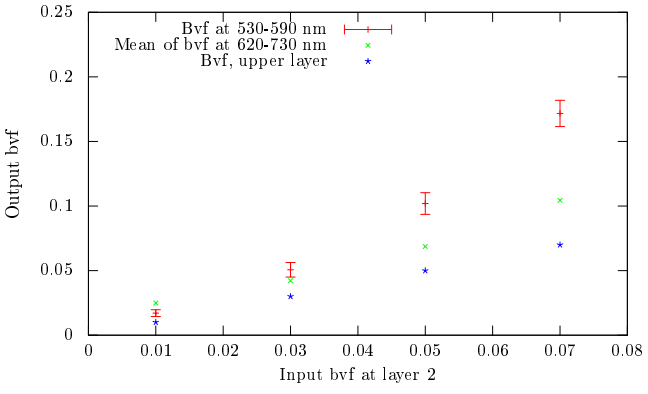
Figure 4.75: The derived oxygen saturation and blood volume fraction using the inverse Delta-Eddington diffusion model on MCML spectra, as a function of the melanin absorption coefficient input to MCML. MCML spectra generated using oxygen saturation 0.5 and 0.9 at the first and second dermal layer respectively, and 0.01 and 0.03 as the blood volume fraction at the same layers. The melanin curve is used in the last iteration of the epidermal fitting.



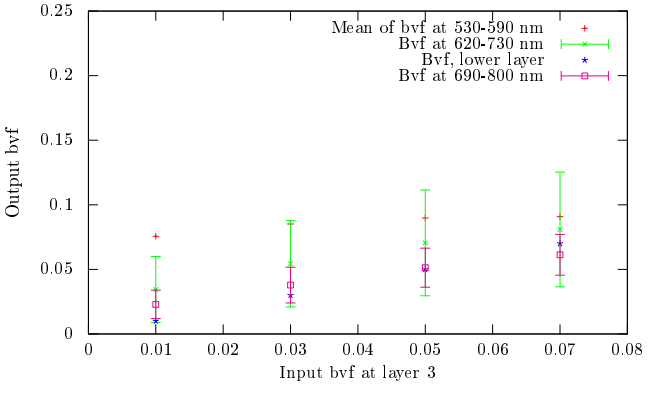
(a)



(b)

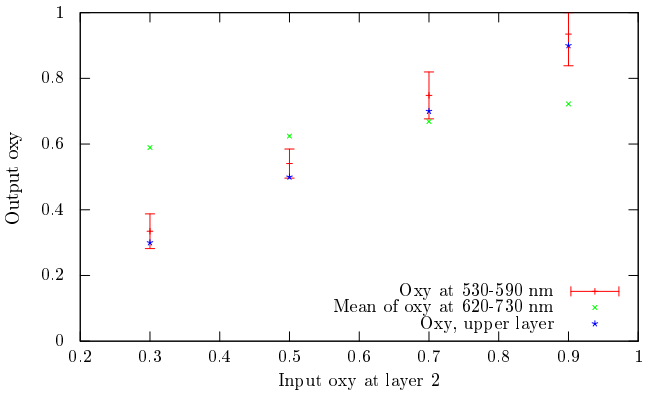


(c)

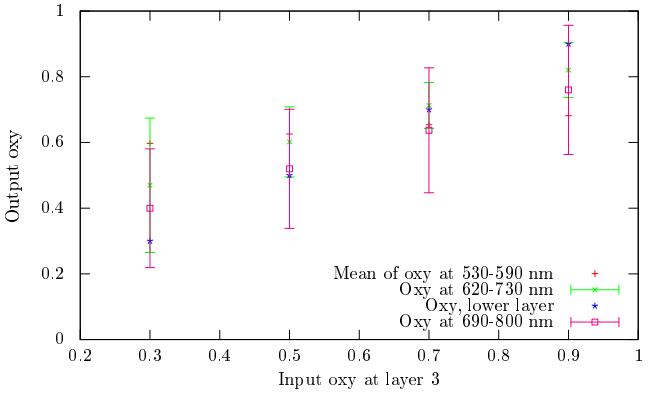


(d)

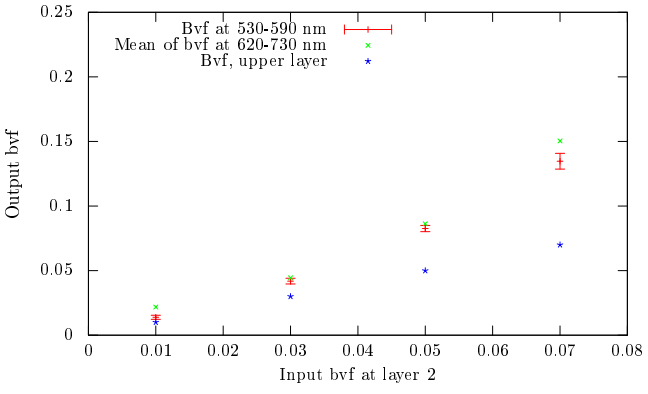
Figure 4.76: The stability of the derived parameters using the inverse isotropic diffusion model. Input reflectance was simulated using MCML. $\mu_{a,m,694} = 500$ m^{-1}, $d2 = 350$ μm. The bvfs on the respective layers were varied between 0.01, 0.03, 0.05 and 0.07, the respective oxy were varied between 0.3, 0.5, 0.7 and 0.9.



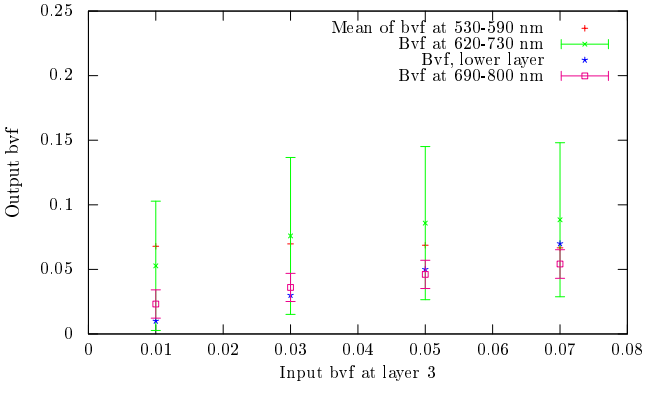
(a)



(b)

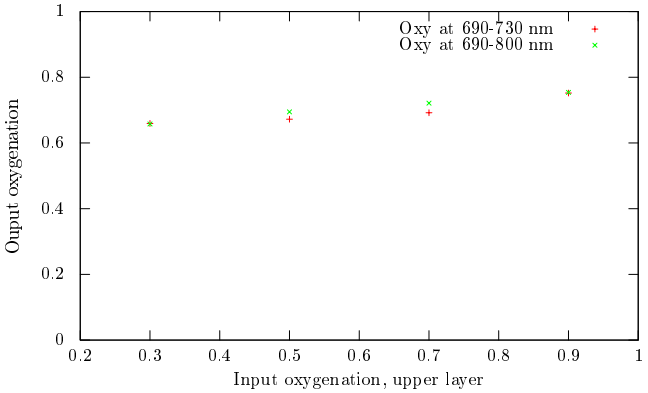


(a)

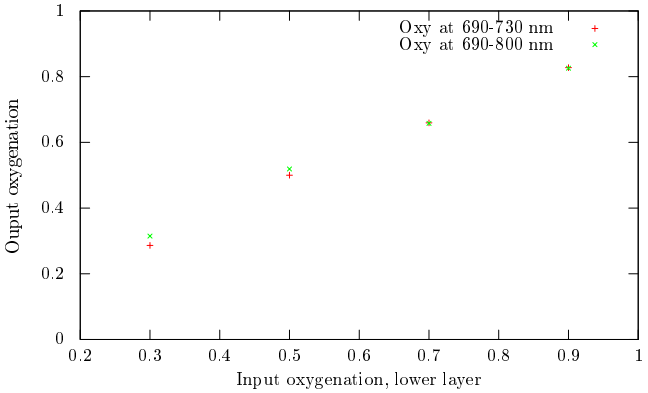


(b)

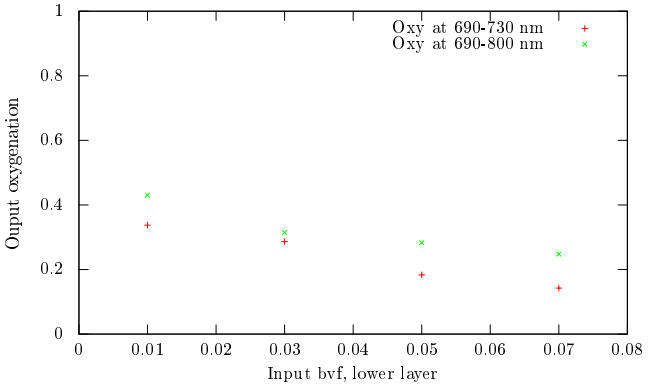
Figure 4.76: The stability of the derived parameters using the inverse Delta-Eddington diffusion model. Input reflectance was simulated using MCML. $\mu_{a,m,694} = 500$ m^{-1}, $d2 = 350$ μm. The bvfs on the respective layers were varied between 0.01, 0.03, 0.05 and 0.07, the respective oxy were varied between 0.3, 0.5, 0.7 and 0.9.



(a) $\mathrm{Bvf}_1 = 0.01$, $\mathrm{bvf}_2 = 0.03$, $\mathrm{oxy}_2 = 0.7$.



(b) $\mathrm{Bvf}_1 = 0.01$, $\mathrm{bvf}_2 = 0.03$, $\mathrm{oxy}_1 = 0.3$.



(c) $\mathrm{Bvf}_1 = 0.01$, $\mathrm{oxy}_1 = 0.3$, $\mathrm{oxy}_2 = 0.3$.

Figure 4.77: A comparison between fitting ranges for a stable set of input parameters. $\mu_{a,m,694} = 500$ m^{-1}, $d2 = 350$ μm.

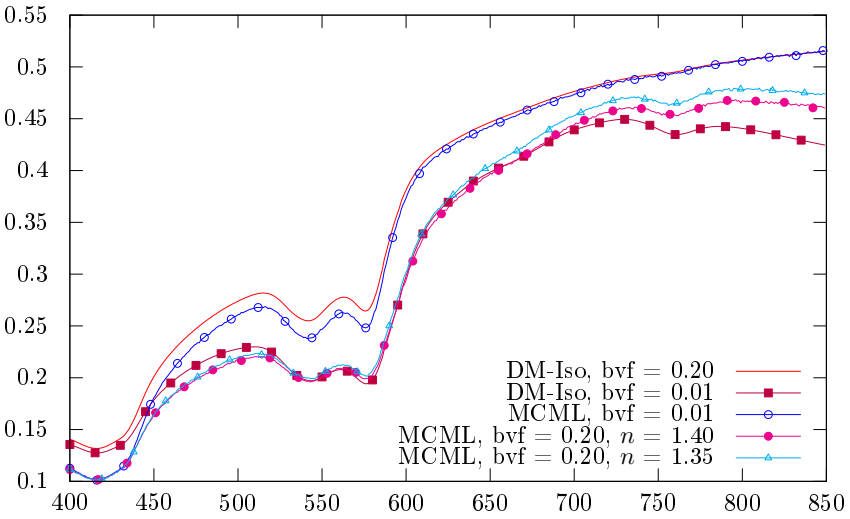


Figure 4.78: The isotropic diffusion model versus MCML for "normal" skin and skin containing a blood vessel approximation and skin just containing much blood. All were simulated using bvf = 0.01 in the upper layer, oxy = 0.6 in the upper dermis layer and oxy = 0.7 in the lower dermis layer.

4.6 Feasibility

The results depend heavily on the correct calibration of the data. The correct focusing is in part enforced by an autocalibration system, but the body samples will vary spatially across the field of view and go in and out of focus. Some of the parameters become unphysical only with the slightest unfocusedness, and with shadows and light source variation not being corrected by correcting the light source variation in the flat field calibration.

It will be difficult to get absolutely correctly calibrated data, which will make the model assumptions rather misaligned with reality, even if the model already assumes assumptions at odds with reality. The model should be made more robust to such discrepancies, either through preprocessing or in the treatment of the dermal absorption coefficient itself, after inversion. In the unmixing stage are fixed endmembers assumed. Some slack should be given and some probability density functions assigned to the individual levels, or the endmembers should be extracted directly from the image. But this will not necessarily solve the problems, as the problems will vary spatially across the field of view. The problems will also influence the melanin determination and lock the subsequent fitting into infeasibility, as both blurring and shadowing seem to influence the melanin level to be higher. If anything should the unmixing methods used in the melanin fitting be modified.

It is not immediately clear how this should be done. The non-linear least squares methods are "classical" methods [46]. The assumption is that the parameters to estimate are deterministic, just clouded by convolutions inherent in the model. The other approach will be to assume the parameters random, following a probability distribution, and leading to the Bayesian approach for estimation [46]. However, this will neither be correct, as the parameters to estimate (blood volume fraction, oxygenation, etc.) are in reality deterministic, but potentially can scene variability be implemented as more uncertainty in the variables. Bayesian estimation theory's strength is the potential for incorporating prior knowledge into the parameters [46], which is not more known when scene variability is taken into account. It can however be used for incorporating spatial information [99]. Partial least squares is often used in chemical spectroscopy community for unmixing absorption spectra [37]. This is a statistical approach, but its main strength is not feasibility variability but to correct for co-linearity, given that multiple chromophores are codependent. Zhang et al. [113] gave a method for allowing scene variability in ordinary, constrained unmixing, but this was mainly to alleviate the sum-to-one-constraint, which is not a constraint present in this problem. There does exist literature covering how to correct for scene variability and shadows [78, 89], but these will normally assume that the terrain levels are known a priori.

Still can it be assumed that the region of interest will be correctly focused, though the outlying parts are not.

In addition will there be other problems. Specular reflection is removed by applying crossed polarization filters, but the removal will hinge on the assumption that the imaged surface is flat. This will not be the case. Light from the surroundings will be diffusively reflected from the skin, which can in part cause some spectral characteristics not reminiscent of skin. The model will assume boundary conditions to never vary, which is especially not true in the blood vessels. The scattering is assumed to never vary, and this is known to not be true, and will give errors accumulating in the melanin determination. In the derivation of the model is the situation assumed to be one-dimensional, i.e. that light scattered and absorbed can be accounted for along one axis, inside one, large slab of skin. This is an assumption which can be assumed correct when dealing with spectrometers, but in the hyperspectral inversion will this assumption be assumed correct in each pixel. In other words will light be assumed to not scatter in-between pixels. Given the large penetration depths will this not be the case. If the pixels are integrated can this be more correct, however, as the situation will be more reminiscent of when measuring using a spectrometer. In this work has pixels been unmixed separately, but to make the independence assumption more correct can the results be meaned over some neighbourhood.

Noise-free images have been assumed by taking MNF-de-noised images as input. In a real-time situation can the noise removal of the images be less optimal.

4.7 Calibration and visualization

Some smaller parts of the task was to develop visualization for the results and real-time automatic calibration of the hyperspectral data.

4.7.1 Visualization

An example of the visualization is shown in figure 4.79. It is possible to see the relative changes, but the visualization of the different blood volume fractions is not good enough as it is not possible to discern the exact oxygenation value. This will mean that the visualization class must be subclassed into different oxygenation and blood volume fraction visualization classes so that the blood volume fraction and oxygenation can correctly be calculated, or the GPU-DM stage must output the oxygenation and blood volume fraction instead of the deoxy and oxy absorption coefficients. As all other data are output as absorption coefficients, would doing this go slightly against output expectations.

4.7.2 Calibration

Data containing the reflectance standard that is to be used in the future with other objects present in the field of view were not available, and much work was therefore not put into the development of the calibration stage. Still, some general notes will be discussed.

The scene common in the present images is a leg lying on top of a pillow. At the start of the image is there a paper strip to correct for light source variation and a circular reflectance standard to convert radiance data into reflectance data.

The light source variation cannot be corrected using a sheet of paper. The paper will diffusively reflect different intensities at the different wavelengths due to the presence of chromophores in the paper, as is shown in figure 4.80. This is not good enough. The band having the best signal to noise ratio was chosen as the band to correctify the image against, and this resulted in non-fittable spectra. No images only containing the paper strip as the light variation detection were therefore corrected for the variation of the light source across the field of view. This is likely one major source of error. Only the image in 4.45a was corrected against the variation of the light source as it had the calibration slab across the entire field of view present.

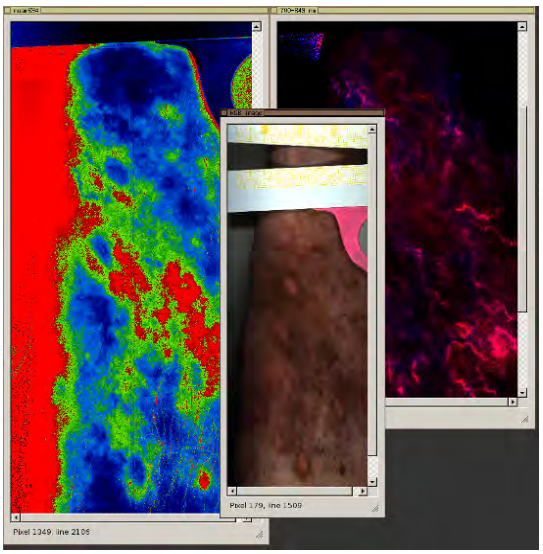


Figure 4.79: Visualization in GPU-DM

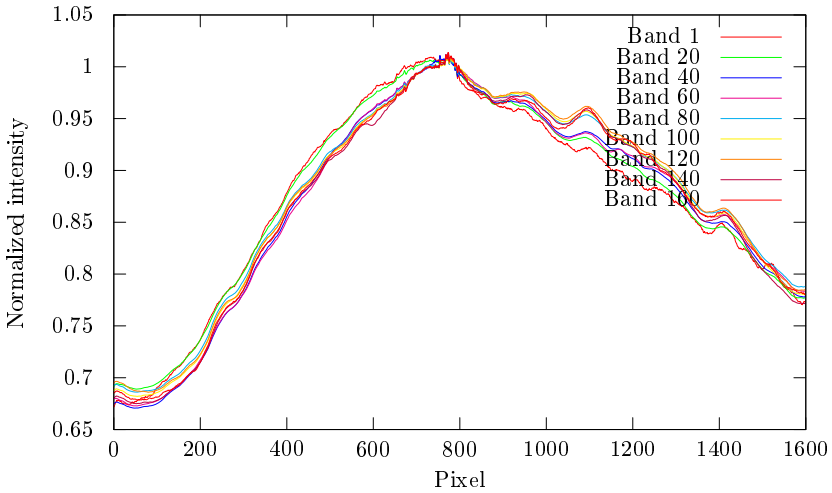


Figure 4.80: The light source variation across the field of view for different bands, as obtained from the paper strip.

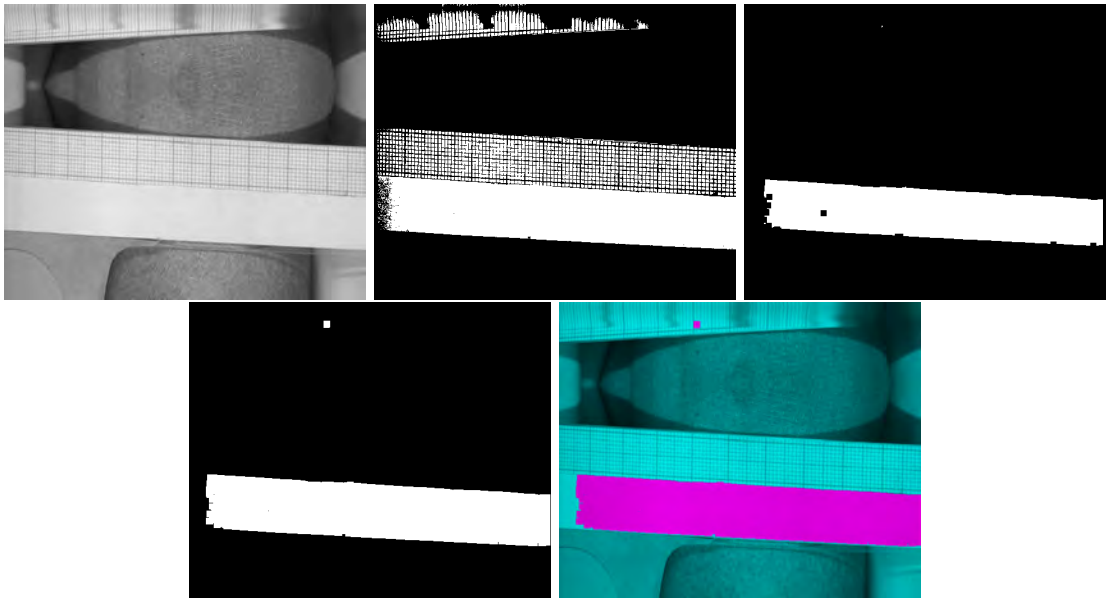


Figure 4.81: Reflectance standard extraction using simple thresholding. Band 20 is first thresholded based on some arbitrary value, eroded by a square structure element of size 10 and finally dilated by the same structuring element. The image is successfully segmented.

Some preliminary results were found regarding the detection of the calibration slab using simple thresholding operations. Even if the paper sheet will not be a correct reflectance standard will it still have the same intensity characteristics as the reflectance standard to be used. Thresholding line by line was found to be too unstable, and the stage will have to wait for a certain amount of lines before attempting detection to achieve good and stable results.

If only one band is segmented based on thresholding may the paper strip be segmented from the rest, as is seen in figure 4.81. This method was not very stable, and for these cases was the circular disc in the lower left corner desired, as this was a true reflectance standard.

In figure 4.82 are the uncalibrated spectra present in the scene shown. Based on this may optimal bands be chosen for thresholding, as the different materials to be removed from the scene will have different responses at different bands. Skin exhibits, as is known due to the high absorption of blood and melanin at the lower wavelengths, a lower image intensity than the rest of the materials at the first bands. The intensity is low also for the other bands due to the low signal to noise ratio due to the Gaussian light source, but still higher than the intensity for skin. In figure 4.83 is the first band in the hyperspectral image shown. The tissue is darker than the rest of the image. In the same figure is the image histogram shown, and in figure 4.84 the image histogram after the image has been smoothed using a Gaussian filter. Using the minima present in the image histogram can the image be thresholded to yield each of the binary images shown in figure 4.85. This is not 100 percent successful in segmenting the different, desired parts of the image, and other bands must be used.

Looking back at figure 4.82, all of the curves exhibit similar behavior due to light source contamination, especially for wavelength bands higher than band 80. The materials are mainly differentiated based on the first 80 bands. The pillow and the stripe are difficult to separate as they display the same behavior, and the pillow has a lower intensity only due to being partially in shadow. The disc also displays similar behavior, but has a lower intensity that is guaranteed to be lower than the rest because of its specification of reflecting only a fraction of 0.4 of the incoming light. The material surrounding the calibration slab has, however, a pinkish color which is easy to see the effects of in the spectrum. Skin is also characteristic up till the 80th band.

The important thing is to segment the middle parts of the stripe from the outlying pillow parts. Thus, band 50, where the difference seemingly is the largest, can be used to segment pillow from stripe. Band

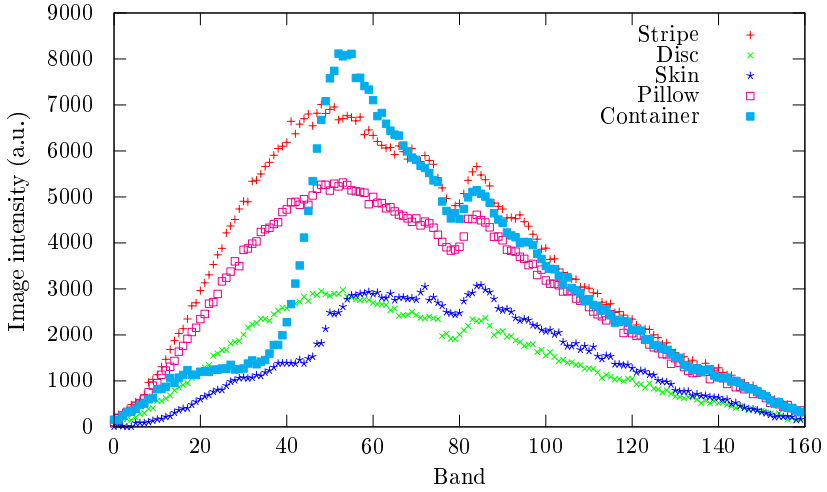


Figure 4.82: Uncalibrated spectra of different materials present in the calibration scene. These are plucked from one individual pixel present in each of the materials.

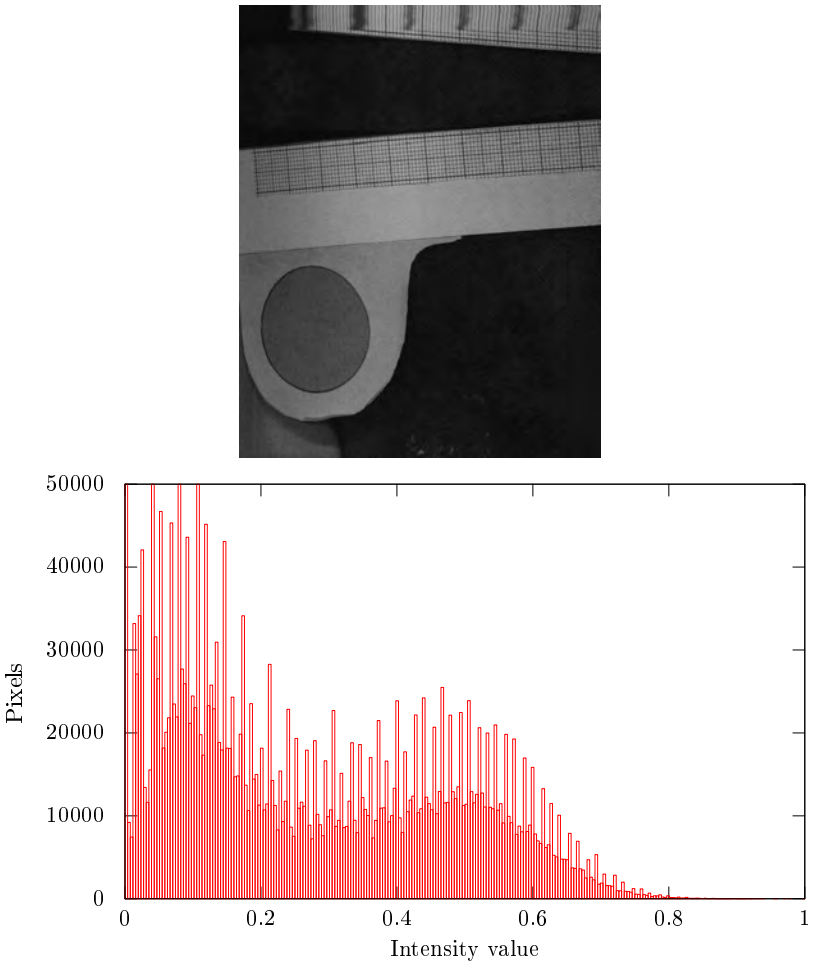


Figure 4.83: Band 0 of an image along with its image histogram

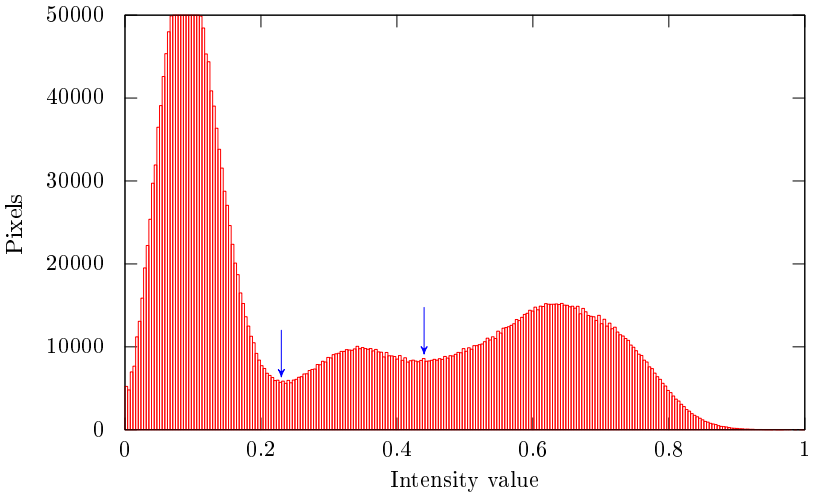


Figure 4.84: Image histogram after smoothing using a Gaussian filter of size 10. Arrows indicate the positions for later segmentation in figure 4.85.

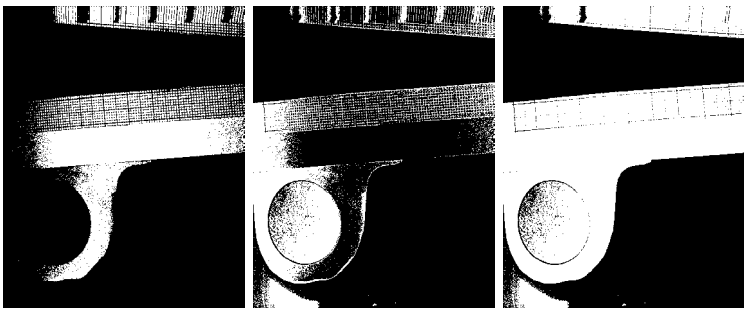
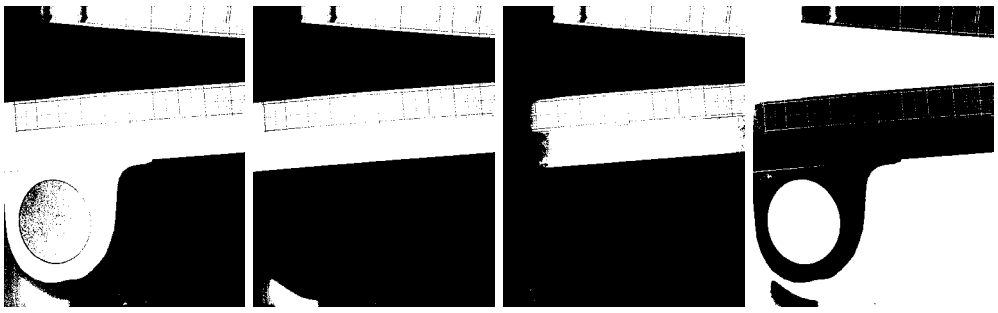


Figure 4.85: Segmentation of band 0 using the image histogram in figure 4.84.



(a) Image thresholded based on band 0

(b) Band 30 thresholded after multiplication by binary image from band 0.

(c) Band 50 thresholded after multiplication by binary image from band 30.

(d) Band 60 thresholded after multiplication by binary image from band 0.

Figure 4.86: The reflectance standard routine used in the final version.

30 can be used to segment the disc-surrounding material from the stripe as it has a low image intensity here due to the pinkish color, and band 60 can be used to enhance its contrast to skin and pillow alike. While this does not neccessarily yield a good segmentation of the calibration disc, it will yield its edges, which can be used in a Hough circle transform and segmented using some basic region growing technique. The result is shown in figure 4.86. In **CalibrationStage** was the disc detected by flood filling areas and dilating and eroding away the remaining, small structures.

The main obstacle for thresholding operations is the pillow, as it has a similar spectral behavior as a high intensity reflectance standard. The thresholding operations are therefore not enough for much else than what this calibration stage has been used for so far, namely the calibration of single hyperspectral images before saving to disc and MNF-denoising. These techniques cannot be used in the future, real-time setup, but then the calibration slab will also be square and easier to detect using i.e. the Hough transform [33]. The OpenCV documentation shows through examples that this may be made easy [65].

There are still some general problems present either way.

- Dynamicity. The method waits for a certain amount of lines before attempting to extract the calibration slab. If the calibration slab always is placed in approximately the same area, this poses no problems, but variations can.
- Memory. The whole image up till the limit must be saved for possible calibration standard extraction.
- Time. The number of lines becomes large in order to account for the possible variation of the slab placement. The integration space and time taken to properly integrate the calibration slabs becomes huge and processing of subsequent lines is delayed.
- Information. The whole segment of the image in which the calibration slab might have been is discarded, and information is lost, if no delay in processing is desired.

These procedures can be done every 100 lines and quit when the calibration slab surely has been detected if the square detection may be done fast. An objective measure of when the reflectance standard has been detected will not be easier to develop, however.

Some methods have been proposed for real-time detection of the calibration objects in the scene. In the future, a square reflectance standard which follows the contour of the leg will be used instead of a circular reflectance standard together with paper. This makes the above methods obsolete, and the methods were therefore not tested much in detail or enhanced. The methods were only developed to the extent that it would make analysis for this master thesis slightly easier than manually hard coding the pixel coordinates of the reflectance standards.

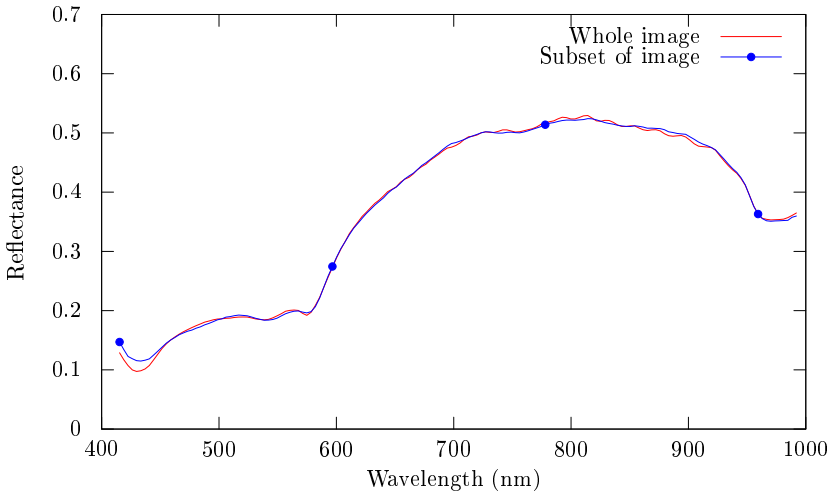


Figure 4.87: The spectra from the same pixel as obtained when using the entire image and a subset of the image containing only skin in the forward and inverse MNF transform. An estimate of the noise was obtained using the same subset of the image in both cases.

4.7.3 Skin masking

A concern which is a concern regardless of the reflectance standard in use in the future is masking of skin for MNF noise removal. MNF was used to segregate noise from the data. At first, the MNF transform was calculated from the entire hyperspectral image, but this resulted in non-fittable spectra. Foreign objects with a high signal to noise ratio, like the pillow, were present in the scene. Each individual spectrum is calculated as a linear combination of different noise-free spectra when the inverse MNF transform is applied. The spectra from the pillow and the like will in part be included in the spectra supposed to represent skin if the MNF transform is applied on the entire scene. The noise is removed, but wrong spectral properties are assigned to the wrong pixel to compensate. The difference between the spectra from the same pixel when the whole image is used to calculate the MNF transform and when a subset only containing skin is used to calculate the MNF transform is shown in figure 4.87.

The difference is not large, but there are small spectral artifacts present in the spectrum obtained using the whole image as input in the MNF transform. These are not present when a subset is used.

Segmenting based on thresholding was not found to be very successful, particularly of the presence of pillows partially being in shadow. Instead was spectral matching used: The mean of the middle parts of the image was compared against all the hyperspectral pixels in the image using equation (2.46). The result is shown in figure 4.88. Only the parts of the image being skin is distinct enough. The skin can after this transform be segmented using simple thresholding. There will be holes in the image due to parts of the skin not being similar to the central parts, but the boundaries may still be found by walking across the thresholded image starting at the edges.

Figure 4.88: Spectral matching of a hyperspectral image using the spectra in the middle parts of the image and (2.46).

Chapter 5

Conclusion and further work

A fast inverse light transport model using GPU parallelization has been developed and presented. The method itself, from input reflectance data to melanin coefficients and unmixed results to three separate penetration depths has been found to satisfy the deadline requirement of 30 ms by delivering its final unmixing results within 3.5 ms across a hyperspectral data line of 160 bands $\times$ 1600 pixels.

An obstacle has been to segregate the melanin absorption in epidermis from the properties residing in the separate dermis. By comparing different possible approaches has this thesis landed on a method which will avoid the problems present in other, commonly used methods. The approach used has been to first ignore the epidermal melanin absorption, fit all chromophores to dermis using dermal inversion and spectral unmixing, and then derive the required epidermal absorption by removing melanin from dermis. This is run for two iterations to remove underestimation problems. The method will theoretically and has been shown on real data to underestimate the melanin results, particularly for high melanin absorptions and an extreme blood volume fraction and low oxygenation, by 10%. This is however seen as less of a problem than overestimation. Underestimation of melanin can be compensated using more absorption, and rectified. Overestimation is more difficult to escape. A good estimate, close to the real value although underestimated, will in any case be useful.

A two-layered approach has been used to estimate the properties down to some penetration depth using different, carefully chosen wavelength ranges. This approach has been verified to be able to characterize relative differences in the parameters both on real data and simulations. As an absolute measure is the approach more uncertain. For pure visualization purposes will this be enough, but further calculations using the parameters can prove to be difficult as they will be too unreliable. Still, the melanin absorption will not be far from the actual value. Both the melanin absorption and the parameter estimates, corrected assuming a simple meaning process, can be used as a starting point for a far more arithmetically complicated three-layered fit.

An approach to detecting the required melanin type has been tested using the scaling of absorption values, but found inconclusive. It is certain that the presence of different melanin types will influence the spectrum dramatically as there are large differences in the absorption of the different melanin types. Testing different melanin types on the available data would however produce results that could be explained by calibration errors, light source variation errors, focusing errors, depth variations of the imaged body samples or an erroneous scattering assumption.

The abundance estimation algorithm in use, SCA, has been found to be unreliable as it has no convergence guarantees. Proper convergence was not seen in the presence of many chromophores, though it was successful enough for a low number of chromophores. In addition will the unmixing after inversion, and the unmixing used in the melanin determination stage, assume perfect data. This was seen to not be the case. SCA should be exchanged for a different algorithm, but the question is whether this should be some statistical method to include spatial information and leniency due to data miscalibration, or it just should be exhanged for a different NNLS algorithm with proper convergence guarantees. Likely is the last item not enough.

Visualization and automatic calibration has been developed. Proper data with a proper reflectance standard present was not available. The calibration was developed to the extent that it must be tweaked for each image, and is not automatic enough for proper real-time use. More advanced image processing methods are required, and should be tested once better data are available. The visualization will characterize the relative differences well enough. Single pixel spectra can be investigated by clicking pixels in the image.

GPGPU had its golden era in 2006-2010, with cheap GPUs being available for high performance computing. With the introduction of Tesla GPUs are NVIDIA aiming to tailor specific GPUs for computing. The amount of hyperspectral data and the resolution of hyperspectral data will scale with increasing resources available, as will the real-time requirements. Using commodity graphics cards might no longer be viable in the future, requiring the far more expensive high performance computing cards or a different strategy altogether. In any case can the parallelization strategies and inverse methods developed here be used in the future. There is also some possibility for optimizing the running times by tweaking things like the threads per block size and inverting multiple lines in parallel.

Bibliography

[1] *CUDA C Best Practices Guide*, October 2012. http://docs.nvidia.com/cuda.

[2] *CUDA C Programming Guide*, October 2012. http://docs.nvidia.com/cuda.

[3] S. Alaluf, U. Heinrich, W. Stahl, H. Tronnier, and S. Wiseman. Dietary carotenoids contribute to normal skin color and uv photosensitivity. *J Nutr*, 132(3):399–403, 2001.

[4] E. Alerstam, W. C. Y. Lo, T. D. Han, J. Rose, S. Andersson-Engels, and L. Lilge. Next-generation acceleration and code optimization for light transport in turbid media using gpus. *Biomed Opt Express*, 1(2):658–675, 2010.

[5] E. Alerstam, T. Svensson, and S. Andersson-Engels. Parallel computing with graphics processing units for high-speed monte carlo simulation of photon migration. *J Biomed Opt*, 13:060504, 2008.

[6] AMD. http://developer.amd.com/resources/heterogeneous-computing/opencl-zone/. Visited 2013-06-01.

[7] A. N. Bashkatov, E. A. Genina, V. I. Kochubey, M. M. Stolnitz, T. A. Bashkatova, O. V. Novikova, A. Y. Peshkova, and V. V. Tuchin. Optical properties of melanin in the skin and skin-like phantoms. In *Proc. SPIE 4162, Controlling Tissue Optical Properties: Applications in Clinical Study, 219*, volume 4162, pages 219–226, 2000.

[8] A. N. Bashkatov, E. A. Genina, V. I. Kochubey, and V. V. Tuchin. Optical properties of human skin, subcutaneous and mucous tissues in the wavelength range from 400 to 2000 nm. *J Phys D: Appl Phys*, 38:2543–2555, 2005.

[9] H. K. Biesalski and U. C. Obermueller-Jevic. Uv light, beta-carotene and human skin - beneficial and potentially harmful effects. *Arch Biochem Biophys*, 389:1–6, 2001.

[10] A. Bjorgan. Real time inverse modelling of hyperspectral images. Technical report, NTNU, 2012.

[11] R. Bro and S. D. Jong. A fast non-negativity-constrained least squares algorithm. *J Chemometr*, 11:393–401, 1997.

[12] G. Broughton II, J. E. Janis, and C. E. Attinger. Wound healing: An overview. *Plast Reconstr Surg*, 117:1e–S–32e–S, 2006.

[13] H. Buiteveld, J. M. H. Hakvoort, and M. Donze. The optical properties of pure water. In *Proc. SPIE 2258, Ocean Optics XII, 174*, volume 2258, pages 174–183, 1994.

[14] A. Burns and A. Wellings. *Real-Time systems and programming languages*. Addison-Wesley, 2009.

[15] C. Chang and D. C. Heinz. Constrained subpixel target detection for remotely sensed imagery. *IEEE T Geosci Remote*, 38(3):1144–1159, 2000.

[16] D. Chen and R. J. Plemmons. Nonnegativity constraints in numerical analysis. In *The Birth of Numerical Analysis*. World Scientific Publishing Company, 2009.

[17] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2009.

[18] J. B. Dawson, D. J. Barker, D. J. Ellis, J. A. Cotterill, E. Grassam, G. W. Fisher, and J. W. Feather. A theoretical and experimental study of light absorption and scattering by in vivo skin. *Phys Med Biol*, 25(4):695, 1980.

[19] A. R. De Pierro. On the convergence of the iterative image space reconstruction algorithm for volume ect. *IEEE T Med Imaging*, MI-6(2):174–175, 1987.

[20] M. Denstedt, B. S. Pukstad, L. A. Paluchowski, J. E. Hernandez-Palacios, and L. L. Randeberg. Hyperspectral imaging as a diagnostic tool for chronic skin ulcers. In *Proc. SPIE, Photonic Therapeutics and Diagnostics IX*, volume 8565, pages 85650N–85650N–14, 2013.

[21] L. F. A. Douven and G. W. Lucassen. Retrieval of optical properties of skin from measurement and modeling the diffuse reflectance. In *Proc. SPIE 3914, Laser-Tissue Interaction XI: Photochemical, Photothermal, and Photomechanical, 312*, pages 312–323, 2000.

[22] H. Du, R. A. Fuh, J. Li, A. Corkan, and J. S. Lindsey. Photochemcad: A computer-aided design and research tool in photochemistry. *Photochem Photobiol*, 68:141–142, 1998.

[23] G. A. Elliott, B. C. Ward, and J. H. Anderson. Gpusync: A framework for real-time gpu management. In submission, available on https://wiki.litmus-rt.org/litmus/Publications (visited 2013-06-01).

[24] ENVI. http://www.exelisvis.com/productsservices/envi/envi.aspx. Visited 2013-06-01.

[25] S. I. Fox. *Human Physiology*. McGraw-Hill Higher Education, 2010.

[26] V. Franc, V. Hlavac, and M. Navara. Sequential coordinate-wise algorithm for the non-negative least squares problem. Technical report, Center for Machine Perception, Czech Technical University, Prague, Chezch Republic, 2005.

[27] J. Franke, D. A. Roberts, K. Halligan, and G. Menz. Hierarchical multiple endmember spectral mixture analysis (mesma) of hyperspectral imagery for urban environments. *Remote Sens Environ*, 113(8):1712 – 1723, 2009.

[28] V. Fresse, D. Houzet, and C. Gravier. Gpu architecture evaluation for multispectral and hyperspectral image analysis. In *Design and Architectures for Signal and Image Processing (DASIP), 2010 Conference on*, pages 121 –127, oct. 2010.

[29] V. Fresse, D. Houzet, and C. Gravier. Evaluation of cpu and gpu architectures for spectral image analysis algorithms. In *Proc. SPIE, Parallel Processing for Imaging Applications*, pages 78720M–78720M–11, 2011.

[30] J. Galeano, R. Jolivot, F. Marzani, and Y. Benezeth. Unmixing of human skin optical reflectance maps by non-negative matrix factorization algorithm. *Biomed Signal Proces*, 8(2):169–175, 2012.

[31] Gnuplot. http://www.gnuplot.info. Visited 2013-06-08.

[32] C. Gonzalez, S. Sanchez, A. Paz, J. Resano, D. Mozos, and A. Plaza. Use of fpga or gpu-based architectures for remotely sensed hyperspectral image processing. *Integration*, 46:89–103, 2013.

[33] R. C. Gonzalez, R. E. Woods, and S. L. Eddins. *Digital Image Processing using MATLAB*. Gatesmark Publishing, 2009.

[34] C. González, J. Resano, A. Plaza, and D. Mozos. Fpga implementation of abundance estimation for spectral unmixing of hyperspectral data using the image space reconstruction algorithm. *IEEE J Sel Top Appl*, 5:248–261, 2012.

[35] D. González, C. Sánchez, R. Veguilla, N. G. Santiago, S. Rosario-Torres, and M. Vélez-Reyes. Abundance estimation algorithms using nvidia (r) cuda (tm) technology. In *Proc. of SPIE Vol. 6966*, 2008.

[36] A. A. Green, M. Berman, P. Switzer, and M. D. Craig. A transformation for ordering multispectral data in terms of image quality with implications for noise removal. *IEEE T Geosci Remote*, 26:65–74, 1988.

[37] D. M. Haaland and E. V. Thomas. Partial least-squares methods for spectral analyses. 1. relation to other quantitative calibration methods and the extraction of qualitative information. *Anal Chem*, 60:1193–1202, 1988.

[38] R. C. Haskell, L. O. Svaasand, T. Tsay, T. Feng, M. S. McAdams, and B. J. Tromberg. Boundary conditions for the diffusion equation in radiative transfer. *J Opt Soc Am A*, 11(10):2727–2741, Oct 1994.

[39] D. C. Heinz and C. Chang. Fully constrained least squares linear spectral mixture analysis method for material quantification in hyperspectral imagery. *IEEE T Geosci Remote*, 39(3):529–545, 2001.

[40] L.G. Henyey and J.L Greenstein. Diffuse radiation in the galaxy. *Astrophys J*, 93:70–83, 1941.

[41] HySpex. http://www.hyspex.no. Visited 2013-06-03.

[42] A. Ishimaru. *Wave Propagation and Scattering in Random Media*. Wiley-IEEE Press, 1997.

[43] S. L. Jacques, C. A. Alter, and S. A. Prahl. Angular dependence of hene laser light scattering by human dermis. *Lasers Life Sci*, 1:309–333, 1987.

[44] S. L. Jacques, R. Samatham, and N. Choudhury. Rapid spectral analysis for spectral imaging. *Biomed Opt Express*, 1(1):157–164, 2010.

[45] S. Kato, K. Lakshmanan, A. Kumar, M. Kelkar, Y. Ishikawa, and R. Rajkumar. Rgem: A responsive gpgpu execution model for runtime engines. In *Real-Time Systems Symposium (RTSS), 2011 IEEE 32nd*, pages 57–66, 2011.

[46] S. M. Kay. *Fundamentals of Statistical Signal Processing: Estimation theory*. Prentice Hall PTR, 1993.

[47] N. Keshava and J.F. Mustard. Spectral unmixing. *IEEE Signal Proc Mag*, 19(1):44 –57, jan 2002.

[48] A. Kienle, M. S. Patterson, L. Ott, and R. Steiner. Determination of the scattering coefficient and the anisotropy factor from laser doppler spectra of liquids including blood. *Appl Opt*, 35(19):3404–3412, 1996.

[49] D. Kim, S. Sra, and I. S. Dhillon. A new projected quasi-newton approach for the nonnegative least squares problem. Technical report, The University of Texas at Austin, 2006.

[50] J. Kim and H. Park. Toward faster nonnegative matrix factorization: A new algorithm and comparisons. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, 2008.

[51] N. Kollias and A. Baqer. Spectroscopic characteristics of human melanin in vivo. *J Invest Dermatol*, 85:38–42, 1985.

[52] N. Kollias and A. Baqer. On the assessment of melanin in human skin in vivo. *Photochem Photobiol*, 43:49–54, 1986.

[53] L. L. Randeberg, E. B. Roll, L. T. Norvang Nilsen, T. Christensen, and L. O. Svaasand. In vivo spectroscopy of jaundiced newborn skin reveals more than a bilirubin index. *Acta Paediatr*, 94(1):65–71, 2005.

[54] C. L. Lawson and R. J. Hanson. *Solving Least Squares problems*. Prentice-Hall, 1974.

[55] libgsl. http://www.gnu.org/software/gsl/. Visited 2013-06-01.

[56] Y. Liu, L. Hong, K. Wakamatsu, S. Ito, B. Adhyaru, C-Y. Cheng, C. R. Bowers, and J. D. Simon. Comparison of structural and chemical properties of black and red human hair melanosomes. *Photochem Photobiol*, 81(1):135–144, 2007.

[57] Y. Luo and R. Duraiswami. Efficient parallel nonnegative least squares on multicore architectures. *SIAM J Sci Comput*, 33:2848–2863, 2011.

[58] S. J. Matcher, M. Cope, and D. T. Delpy. In vivo measurements of the wavelength dependence of tissue-scattering coefficients between 760 and 900 nm measured with time-resolved spectroscopy. *Appl Optics*, 36(1):386–396, 1997.

[59] MCML. http://omlc.ogi.edu/software/mc/. Visited 2013-06-03.

[60] I. V. Meglinski and S. J. Matcher. Quantitative assessment of skin layers absorption and skin reflectance spectra simulation in the visible and near-infrared spectral regions. *Physiol Meas*, 23(4):741, 2002.

[61] L. Norvang, T. Milner, J. Nelson, M. Berns, and L. Svaasand. Skin pigmentation characterized by visible reflectance measurements. *Laser Med Sci*, 12:99–112, 1997.

[62] L. T. Norvang, E. J. Fiskerstrand, J. S. Nelson, M. W. Berns, and L. O. Svaasand. Epidermal melanin absorption in human skin. In *Proc. SPIE 2624, Laser-Tissue Interaction and Tissue Optics, 143*, volume 2624, pages 143–154, 1996.

[63] NVIDIA. https://developer.nvidia.com/opencl. Visited 2013-06-01.

[64] OpenCL. http://www.khronos.org/opencl/. Visited 2013-06-01.

[65] OpenCV. http://opencv.org/. Visited 2013-06-01.

[66] OpenMP. http://openmp.org. Visited 2013-06-09.

[67] Ocean Optics. http://www.oceanoptics.com/products/ispref.asp. Visited 2013-05-30.

[68] Ocean Optics. http://www.oceanoptics.com/products/usb4000.asp. Visited 2013-05-30.

[69] D. A. Patterson and J. L. Hennessy. *Computer Organization and Design*. Morgan Kaufmann, 2012.

[70] A. Plaza, P. Martinez, R. Perez, and J. Plaza. A quantitative and comparative analysis of endmember extraction algorithms from hyperspectral data. *IEEE T Geosci Remote*, 42(3):650 – 663, march 2004.

[71] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes*. Cambridge University Press, 2007.

[72] Qt. http://qt-project.org/. Visited 2013-06-01.

[73] M.S. Ramsey and P. R. Christensen. Mineral abundance determination: Quantitative deconvolution of thermal emission spectra. *J Geophys Res*, 103(B1):577–596, 1998.

[74] L. L. Randeberg, J. H. Bonesronning, M. Dalaker, J. S. Nelson, and L. O. Svaasand. Methemoglobin formation during laser induced photothermolysis of vascular skin lesions. *Laser Surg Med*, 34(5):414–419, 2004.

[75] L. L. Randeberg, O. A. Haugen, R. Haaverstad, and L. O. Svaasand. A novel approach to age determination of traumatic injuries by reflectance spectroscopy. *Laser Surg Med*, 38(4):277–289, 2006.

[76] L. L. Randeberg, E. L. P. Larsen, and L. O. Svaasand. Characterization of vascular structures and skin bruises using hyperspectral imaging, image analysis and diffusion theory. *J Biophotonics*, 3(1-2):53–65, 2010.

[77] L. L. Randeberg, A. Winnem, R. Haaverstad, and L. O.. Svaasand. Performance of diffusion theory vs monte carlo methods. In *Diagnostic Optical Spectroscopy in Biomedicine III*, page ThB3. Optical Society of America, 2005.

[78] R. Richter, T. Kellenberger, and H. Kaufmann. Comparison of topographic correction methods. *Remote Sens*, 1(3):184–196, 2009.

[79] J. Riesz. *The spectroscopic properties of melanin*. PhD thesis, University of Queensland, 2007.

[80] D. A. Roberts, M. Gardner, R. Church, S. Ustin, G. Scheer, and R. O. Green. Mapping chapparal in the santa monica mountains using multiple endmember spectral mixture models. *Remote Sens Environ*, 65:267–279, 1998.

[81] C. J. Rossbach, J. Currey, M. Silberstein, B. Ray, and E. Witchel. Ptask: Operating system abstraction to manage gpus as compute devices. In *Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles*, number 233-248, 2011.

[82] I. S. Saidi, S. L. Jacques, and F. K. Tittel. Mie and rayleigh modeling of visible-light scattering in neonatal skin. *Appl Opt*, 34(31):7410–7418, Nov 1995.

[83] E. Salomatina, B. Jian, J. Novak, and A. N. Yaroslavsky. Optical properties of normal and cancerous human skin in the visible and near-infrared spectral range. *J Biomed Opt*, 11(6):064026–1 – 064026–9, 2006.

[84] S. Sanchez and A. Plaza. Real-time implementation of a full hyperspectral unmixing chain on graphics processing units. In *Proc. SPIE, Satellite Data Compression, Communications, and Processing VII*, pages 81570F–81570F–9, 2011.

[85] R. R. Seeley, T. D. Stephens, and P. Tate. *Essentials of Anatomy and Physiology*. McGraw-Hill Publishing Co., 2006.

[86] S. T. Seljebotn. Continuous autofocus for line scanning hyperspectral camera. Master's thesis, NTNU, 2012.

[87] J. Setoain, M. Prieto, C. Tenllado, and F. Tirado. Gpu for parallel on-board hyperspectral image processing. *Int J High Perform Comput Appl*, 22(4):424–437, November 2008.

[88] J. Setoain, C. Tenllado, M. Prieto, D. Valencia, A. Plaza, and J. Plaza. Parallel hyperspectral image processing on commodity graphics hardware. In *Parallel Processing Workshops, 2006. ICPP 2006 Workshops. 2006 International Conference on*, pages 8 pp. –472, 0-0 2006.

[89] J. D. Shepherd and J. R. Dymond. Correcting satellite imagery for the variance of reflectance and illumination with topography. *Int J Remote Sens*, 24:3503–3514, 2002.

[90] T. Skauli, T. V. Haavardsholm, Kaasen I., G. Arisholm, A. Kavara, T. O. Opsahl, and Skaugen A. An airborne real-time hyperspectral target detection system. In *Proc. SPIE 7695, Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XVI*, volume 7695, 2010.

[91] T. Spott. *Characterization of layered tissue structures with diffusively propagating photon-density waves*. PhD thesis, NTNU, 1999.

[92] T. Spott and L. O. Svaasand. Collimated light sources in the diffusion approximation. *Appl Opt*, 39(34):6453–6465, Dec 2000.

[93] T. Spott, L. O. Svaasand, R. E. Anderson, and P. F. Schmedling. Application of optical diffusion theory to transcutaneous bilirubinometry. In *Proc. SPIE 3195, Laser-Tissue Interaction, Tissue Optics, and Laser Welding III, 234*, volume 3195, pages 234–245, 1998.

[94] W. Stallings. *Operating systems - internals and design principles*. Pearson Education Limited, 2012.

[95] G. N. Stamatas and N. Kollias. Blood stasis contributions to the perception of skin pigmentation. *J Biomed Opt*, 9(2):315–322, 2004.

[96] G. N. Stamatas, B. Z. Zmudzka, N. Kollias, and J. Z. Beer. In vivo measurement of skin erythema and pigmentation: new means of implementation of diffuse reflectance spectroscopy with a commercial instrument. *Br J Dermatol*, 159(3):683–690, 2008.

[97] J. M. Steinke and A. P. Shepherd. Comparison of mie theory and the light scattering of blood cells. *Appl Opt*, 27:4027–4033, 1988.

[98] J. M. Steinke and A. P. Shepherd. Diffusion model of the optical absorbance of whole blood. *J Opt Soc Am A*, 5:813–822, 1988.

[99] M.R. Stites, T.K. Moon, J.H. Gunther, and G.P. Williams. A bayesian framework for abundance estimation in hyperspectral data using markov random fields. In *Signals, Systems and Computers, 2007. ACSSC 2007. Conference Record of the Forty-First Asilomar Conference on*, pages 725–729, 2007.

[100] L. Svaasand, L. Norvang, E. Fiskerstrand, E. Stopps, M. Berns, and J. Nelson. Tissue parameters determining the visual appearance of normal skin and port-wine stains. *Laser Med Sci*, 10:55–65, 1995.

[101] T. Theoharis, G. Papaioannou, N. Platis, and N. M. Patrikalakis. *Graphics and Visualization: Principles & Algorithms*. A. K. Peters, 2007.

[102] A. J. Thody, E. M. Higgins, K. Wakamatsu, S. Ito, S. A. Burchill, and J. M. Marks. Pheomelanin as well as eumelanin is present in human epidermis. *J. Invest. Dermatol.*, 97:340–344, 1991.

[103] S-H. Tseng, A. Grant, and A. J. Durkin. In vivo determination of skin near-infrared optical properties using diffuse optical spectroscopy. *J Biomed Opt*, 13(1):014016–1 – 014016–7, 2008.

[104] T. Tseng, C. Chen, Y. Li, and K. Sung. Quantification of the optical properties of two-layered turbid media by simultaneously analyzing the spectral and spatial information of steady-state diffuse reflectance spectroscopy. *Biomed Opt Express*, 2(4):901–914, 2011.

[105] M. H. Van Benthem and M. R. Keenan. Fast algorithm for the solution of large-scale non-negativity-constrained least squares problems. *J Chemometrics*, 18:441–450, 2005.

[106] M. J. C. van Gemert, S. L. Jacques, H. J. C. M. Sterenborg, and W. M. Star. Skin optics. *IEEE Trans Biomed Eng*, 36:1146–1154, 1989.

[107] M. Vélez-Reyes, A. Puetz, P. Hoke, R. B. Lockwood, and S. Rosario. Iterative algorithms for unmixing of hyperspectral imagery. In *Proc. SPIE 5093, Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery IX, 418*, volume 5093, pages 418–429, 2003.

[108] R. E. Walpole, R. H. Myers, S. L. Myers, and K. E. Ye. *Probability and Statistics*. Prentice Hall, 2011.

[109] L. Wang, S. L. Jacques, and L. Zheng. Mcml monte carlo modeling of light transport in multi-layered tissues. *Comput Meth Prog Bio*, 47(2):131 – 146, 1995.

[110] L. V. Wang and H. Wu. *Biomedical Optics, Principles and Imaging*. John Wiley & Sons, 2007.

[111] M. L. Wolbarsht, A. W. Walsh, and G. George. Melanin, a unique biological absorber. *Appl Opt*, 20:2184–2186, 1981.

[112] R. Zhang, W. Verkruysse, B. Choi, J. A. Viator, B. Jung, L. O. Svaasand, G. Aguilar, and J. S. Nelson. Determination of human skin optical properties from spectrophotometric measurements based on optimization by genetic algorithms. *J Biomed Opt*, 10(2):024030–024030–11, 2005.

[113] Y. Zhang, X. Fan, and R. Wei. Linear spectral unmixing with generalized constraints for hyperspectral imagery. In *Geoscience and Remote Sensing Symposium, 2012 IEEE International*, pages 4106–4109, 2012.

[114] O. Zhernovaya, Sydorukm O., V. Tuchin, and A. Douplik. The refractive index of human hemoglobin in the visible range. *Phys Med Biol*, 56:4013–4021, 2011.

[115] W. G. Zijlstra, A. Buursma, and O. W. van Assendelft. *Visible and near infrared absorption spectra of human and animal haemoglobin*. VSP, Utrecht, 2000.

[116] G. Zonios and A. Dimou. Light scattering spectroscopy of human skin in vivo. *Opt Express*, 17(3):1256–1267, Feb 2009.

[117] G. Zonios, A. Dimou, I. Bassukas, D. Galaris, A. Tsolakidis, and E. Kaxiras. Melanin absorption spectroscopy: new method for noninvasive skin investigation and melanoma detection. *J Biomed Opt*, 13:014017, 2008.

Appendix A

Inverse parameters

Parameters for the inverse simulations seen in assorted reflectance figures are here presented.

Table A.1: Fitting parameters for inverse simulations.

Figure	bvf_1	oxy_1	$methb_1$ (μM)	bil_1 (μM)	bet_1 (μM)	mel_1 (m^{-1})	$const_1$ (m^{-1})	bvf_2	oxy_2	$methb_2$ (μM)	mel_2 (m^{-1})	$const_2$ (m^{-1})	bvf_3	oxy_3	$methb_3$ (μM)	mel_3 (m^{-1})	$const_3$ (m^{-1})	$\mu_{a,m,694}$ (m^{-1})
4.20	0.010	0.00	0.80	1.20	4.63	90.3	0.00	0.016	0.50	0.00	0.00	59.7	0.016	0.64	29.5	0.00	0.00	680
4.23	0.006	0.00	0.00	0.00	0.00	0.00	0.00	0.012	0.73	0.00	0.00	0.00	0.017	0.73	0.00	0.56	0.39	1190
4.26	0.023	0.71	0.00	2.85	4.45	14.9	0.00	0.019	0.66	0.00	0.00	102	0.035	0.85	2.20	1.19	0.87	493
4.29	0.018	0.00	0.00	0.00	0.00	0.00	0.00	0.015	0.34	0.00	0.00	63.4	0.024	0.66	11.9	0.00	0.00	1030
4.33a	0.022	0.44	0.00	5.42	2.83	18.1	0.00	0.015	0.36	0.00	16.7	86.7	0.047	0.61	0.00	7.67	2.00	981
4.33b	0.005	0.00	0.00	0.00	0.00	0.00	0.00	0.012	0.25	0.00	0.00	0.00	0.018	0.53	0.00	12.4	14.1	921
4.34a	0.013	0.71	74.0	13.1	0.43	0.00	0.00	0.007	0.00	0.00	0.00	140	0.025	0.31	22.0	0.00	0.00	595
4.34b	0.007	0.00	0.00	0.00	0.00	0.00	7.43	0.010	0.05	0.00	0.00	0.00	0.016	0.07	10.6	3.01	2.10	561
4.35a	0.071	0.94	317	91.8	6.12	0.00	0.00	0.042	0.71	0.00	333	0.00	0.056	0.88	49.4	0.00	0.00	977
4.35b	0.027	0.00	0.00	0.00	0.00	0.00	210	0.050	0.53	0.00	0.00	0.00	0.058	0.82	12.8	0.00	0.00	912
4.36a	0.036	0.79	7.20	50.6	0.00	0.00	0.00	0.018	0.44	100	73.0	0.00	0.044	0.63	15.1	0.00	0.00	460
4.36b	0.019	0.54	0.00	7.64	0.00	0.00	0.00	0.019	0.40	0.00	0.00	7.04	0.035	0.64	0.00	3.39	5.06	397
4.40a	0.019	1.00	783	1900	37.3	0.00	0.00	0.011	1.00	996	0.00	0.00	0.027	0.78	40.8	0.00	0.00	809
4.40b	0.000	0.00	0.00	0.00	11.6	34.1	248	0.013	1.0	0.00	70.5	0.00	0.025	7.50	0.75	2.46	3.51	679
4.41a	0.005	0.00	0.00	0.00	0.00	0.00	0.00	0.012	0.61	0.00	0.00	0.00	0.006	0.38	0.00	2.82	8.11	677
4.41b	0.025	0.95	103	44.0	0.00	0.00	0.00	0.015	0.70	0.00	21.9	159	0.027	0.49	13.4	0.00	0.00	677
4.43	0.024	0.00	0.00	0.00	0.00	79.2	640	0.037	0.37	0.00	259	0.00	0.086	0.59	0.00	4.27	2.48	779
4.47	0.026	0.95	0.00	28.9	6.89	47.9	0.00	0.009	0.48	0.00	113	0.00	0.046	0.83	5.10	1.79	1.39	655
4.57a	0.008	0.82	0.00	16.9	1.77	10.2	0.00	0.003	0.52	0.00	25.7	12.6	0.013	0.83	3.00	0.00	0.00	635
4.57b	0.021	0.22	0.00	6.65	3.22	34.2	14.9	0.014	0.13	0.00	15.9	139	0.043	0.25	5.40	12.8	5.45	356
4.57c	0.021	0.22	0.00	66.5	3.22	34.2	14.9	0.014	0.13	0.00	15.9	139	0.066	0.62	0.00	20.2	0.00	356

Appendix B

Code

B.1 CUDA kernels

Headers for the CUDA kernels are shown here. For full implementation, refer to attached code files (cudakernels.cu, inv.cu, base.cu) if available.

Code B.1: appendix/GPU–DM/cudakernels.h

```
1  //remove muam694-component from input mua
2  __global__ void removeMuam(float *mua, float *endvalues, float *AT, int
       startblockInd, int numBands, int endmembers, int muamInd, size_t pitch);
3
4  //perform moving average of size samples on the input array
5  template<int BLOCK_DIM>
6  __global__ void movingAverage(float *arr, float *outarr, int samplesdiv2,
       size_t pitch);
7
8
9  //deinitializing muam
10 __global__ void muamDeinit(float *muam, size_t pitch);
11
12 __global__ void skindataDeinit(float *muam, float *bvf, float *oxy, float *
       melanin_type, size_t pitch);
13
14 //check scaling of muad at wavelengths 1 and 2 and move the melanin type up one
       step. PHEOMELANIN_GPU -> SVAASAND_MELANIN_GPU -> EUMELANIN_GPU -> ??? wrong
       scattering
15 __global__ void CheckReflectanceScaling(float *muad, float *melanin_type, int
       wlenind1, int wlenind2, size_t pitch, float scaling_thresh);
16
17 __global__ void SetMelaninType(float *melanin_type, int inputType, size_t pitch
       );
18
19 //penetration depth calculations
20 __device__ float calcPenDepth(float muae, float muad, float musd, float gcol);
21 __global__ void PenDepths(float *muae, float *muad, float *muse, float *musd,
       float *gcol, int startwlenind, int endwlenind, size_t pitch, float *
       pendepths, float *pendepthsstd);
22
23
24 //ordinary least squares. Both implementations are in reality unoptimized. Uses
       interpolation. Uppers and downers contain indices for interpolation.
25 __global__ void GPUlseunoptim(float *lsemat, float *input, float *res, float
       startWlen, float stepWlen, int antWlens, int *uppers, int *downers, float *
```

```
    upperwlens, float *downerwlens, size_t pitch, size_t intPitch);
__global__ void GPUlseOptim(float *lsemat, float *input, float *res, int
    startind, int endind, int *numWlens, float *wlens, float *startwlens, int
    antWlens, size_t pitch, size_t intPitch);

//calculate oxy and bvf based on absorption coefficients
__global__ void calcOxyBvf(float *inputRes, float *oxyOut, float *bvfOut,
    size_t pitch);

//NNLS using ISRA
template<int MAX_NUM_BANDS>
__global__ void ISRA(float *mua, float *endout, float *chromarr, size_t pitch,
    int startwlenInd, int numEndmembers, int numBands);

//implementation of SCA using shared memory
template<int MAX_ENDMEMBERS, int MAX_BLOCKDIM>
__global__ void SCA(float *AT, float *H, float *inputmua, float *x, float *mu,
    float *prevChrom, int prevMethbind, int currMethbind, int startind, int
    numBands, int numEndmembers, size_t pitch);

//implementation of SCA using registers
template<int MAX_ENDMEMBERS>
__global__ void SCAFast(float *AT, float *H, float *inputmua, float *x, float *
    mu, float *prevChrom, int prevMethbind, int currMethbind, int startind, int
    numBands, int numEndmembers, size_t pitch);
```

Code B.2: appendix/GPU–DM/inv.h

```
//invert muae from input reflectance
__global__ void ReflIsoL2InvertMuae(float *muae, float *muse, float *muad,
    float *musd, float *gcol, float *lineData, size_t pitch, int startblockInd);

//invert muad from input reflectance
__global__ void ReflIsoL2InvertMuad(float *muae, float *muse, float *muad,
    float *musd, float *gcol, float *lineData, size_t pitch, int startblockind);


//calculate optical parameters given input skin data
__global__ void calcSkinData(float *wlens, float *oxy_arr, float *Bd_arr, float
     *muam694_arr, float *melanintype_arr, float *muae, float *muse, float *muad
    , float *musd,
                float *muh_oxy, float *muh_deoxy, float *melanin_base, float *
                    musm, float *musr, float *musb_base, size_t pitch, int
                    startblockind);

//monochromatic spectral unmixing
__global__ void MultVector(float *multVec, float *mua, float *res, float factor
    , int startwlenind, int endwlenind, size_t pitch);
__global__ void StraightLine(float *wavelengths, float *mua, float wlen, float
    *res, int startwlenInd, int endwlenInd, size_t inputPitch);


//calculate isotropic reflectance, forward only
__global__ void ReflIsoL2(float *muae, float *muse, float *muad, float *musd,
    float *gcol, float *res, size_t pitch, int startblockind);
__device__ float ReflIsoL2Calc(float mua1, float musr1, float mua2, float musr2
    );

//calculate error in reflectance
__global__ void ReflIsoL2ErrorCheck(float *muae, float *muse, float *musd,
    float *gcol, float *AT, float *x, int endmembers, int numbands, float *
```

```
    inputres, float *outputres, size_t pitch, int startblockind, int diff);
```

Code B.3: appendix/GPU–DM/base.h

```
#ifndef BASE_H_DEFINED
#define BASE_H_DEFINED

#include <vector>


//base spectral dependencies
struct GPUSkinBase{
    float *muh_oxy;
    float *muh_deoxy;
    float *melanin_base; //contains (695/lambda)^3.46
    float *gcol; //anisotropy factor for collagen
    float *musm; //unreduced mie scattering
    float *musr; //unreduced rayleigh scattering coefficient
    float *musb_base; //unreduced blood scattering coefficient
    float *wavelengths; //wavelengths
    void allocate(int samples, int bands, size_t byteWidthPerBlock, int
        numberOfBlocks, const std::vector<float> &wavelengths);
};

//skin data
struct GPUSkinData{
    float *muam694;
    float *bvf;
    float *oxy;
    float *melanin_type; //SVAASAND_MELANIN_GPU, EUMELANIN_GPU or
        PHEOMELANIN_GPU
    void allocate(size_t byteWidthPerBlock, int numberOfBlocks, int thrPerBl);
    float *download(int samples); //download all arrays to an host array
    size_t pitch;
    int height;
    size_t byteWidth;
};

//optical properties
struct GPUOpticalProps{
    float *muae;
    float *muse;
    float *muad;
    float *musd;
    void allocate(size_t byteWidthPerBlock, int numberOfBlocks);
};

#endif
```

B.2 Wrapper code

An NDA has been signed with FFI regarding their framework, and some parts of the code were very framework-specific and disclosed too much of the inner workings of the hyperspectral framework. Therefore are only some parts of the code included in this appendix, primarly those calling CUDA kernels, and some other parts to demonstrate the system. With major parts missing will the code not be compilable. Here are only the headers shown. For code/stubs, refer to attached code files (gpudmblokk.cu, calibration.cpp, fitting.cpp, melanin.cpp, visualize.cpp) if available. Some functionality has been cut.

Code B.4: appendix/GPU–DM/gpudmblokk.h

```
//input: spectral image. Output: float image where each band corresponds to a chromophore
class GPUDM{ //omitted inherited classes
    public:
        GPUDM(int bands, int samples, const std::vector<float> &wlens);

        //
        //omitted framework functions
        //

        //inversion
        void invertReflectance(float *refl, bool printToFile, int pixel = -1);
            //takes in a host allocated array, uploads it to the GPU, inverts it using the isotropic diffusion model
        void invertMuam694(int iterations, bool printToFile, int pixel = -1);
            //clutterfunction for inverting muam694. Just to clean up execute() a bit
        void invertMuad(GPUSkinData *skinData, float *simrefl, float *deriv, float *currRefl, int startwlenInd, int endwlenInd, cudaStream_t *processingStream);
        void invertMuae(GPUSkinData *skinData, float *simrefl, float *deriv, float *currRefl, int startwlenInd, int endwlenInd, cudaStream_t *processingStream, bool shouldCalcSkinData);
        void detectMelaninType(float *refl, GPUSkinData *skinData); //detect correct melanin type based on the muad scaling at input wavelengths. Will only be able to determine when there is "pure" pheomelanin and "pure" eumelanin, but a mix is more likely.
        void checkReflectanceScaling(float *refl, GPUSkinData *skinData, int wlenind1, int wlenind2, dim3 gridDim, size_t pitch);

        //spectral unmixing
        void doOneChromSpectralUnmixing(GPUSkinData *skinData, float *gpuMua, int startwlenInd, int endwlenInd, cudaStream_t *processingStream, MelaninDetermination choice);

    private:
        int num_melanin_iterations;

        //inversion arrays, all allocated on the GPU
        GPUSkinData *_skinDataMuadInv;
        GPUSkinData *_skinDataMuaeInv;
        GPUOpticalProps *_optProps;
        GPUSkinBase *_skinBase;
        float *_calarray;
        float *_simrefl;
        float *_deriv;
        float *_currRefl;
        int _numIterations;

        //image properties
        int _bands;
        int _samples;
        std::vector<float> _wlens;
        bool _shouldCal;

        //GPU properties
        size_t _byteWidth;
        int _height;
        size_t _pitch;
```

```
45          int _threadsPerBlock;
46          dim3 _dimBlock;
47          dim3 _dimGrid;
48          cudaStream_t processingStream;
49          cudaStream_t transferStream;
50
51          //melanin fitting variables
52          float _factor;
53          float *_melcurve;
54          float *_gpumeltype; //melanin type. 0: svaasand, 1: pheomelanin, 2:
                eumelanin
55          float *_gpureflscaling; //reflectance scaling calculated from some
                wavelengths
56          int _scalewlenind1;
57          int _scalewlenind2;
58
59          //framework properties
60          //
61          // omitted
62          //
63
64          //wavelength range for melanin determination
65          int _startwlenInd;
66          int _endwlenInd;
67          float _startwlen;
68          float _endwlen;
69
70          SCAFitting *sca450;
71          SCAFitting *sca530;
72          SCAFitting *sca700;
73          SCAFitting *scabefore;
74
75          //to ensure methb inclusion in first SCA fitting
76          float *methbdummy;
77          int methbDummyInd;
78
79          //forward simulation
80          float *_gpumuad;
81          float *_gpures;
82  };
83
84
85  const float SCALE_WLEN_1 = 541;
86  const float SCALE_WLEN_2 = 576;
87
88
89  //fitting on the GPU using SCA
90  class SCAFitting : public LSEFitting{
91      public:
92          SCAFitting(Chromophores chrom, const std::vector<float> &wlens, float
                startwlen, float endwlen, int samples, int bands, int
                threadsPerBlock, size_t pitch);
93          void doFitting(GPUSkinData *skinData, float *gpuMua, bool shouldCopy,
                bool shouldUpdateSkindata, float *prevChrom, int prevMethbind);
94          float *getRes(){return _res;}; //return current chromophore fitting
                result
95          int getNumEndmembers(){return _endmembers;};
96          Chromophores getChrom(){return _chrom;};
97          int getMethbInd(){return _currMethbInd;};
98          float *reflerror(GPUOpticalProps *optProps, GPUSkinBase *skinBase,
                float *measrefl); //calculate the error from forward reflectance
```

```
99          int getFitStartInd(){return _startind;};
100         int getFitEndInd(){return _endind;};
101     protected:
102         void prepSCA(); //create arrays
103         float *_GPUATA; //H matrix on the GPU
104         float *_GPUA; //transpose of chromophore matrix on GPU
105         float *_mu; //den mu-greia i algoritmen anae rikke hva den heter
106         int _startind; //start index for fitting
107         int _endind;
108         int _currMethbInd;
109
110         float *_gpuerror; //for forward reflectance calculation, not really a
                reasonable part of this class but whatever
111 };
112
113 //fitting using ISRA
114 class ISRAFitting : public SCAFitting{
115     public:
116         ISRAFitting(Chromophores chrom, const std::vector<float> &wlens, float
                startwlen, float endwlen, int samples, int bands, int
                threadsPerBlock, size_t pitch);
117         void doFitting(GPUSkinData *skinData, float *gpuMua);
118 };
```

Code B.5: appendix/GPU–DM/calibration.h

```
1  class CalibrationStage{ //inherited classes are omitted
2      public:
3          CalibrationStage(int bands, int samples, float *calarr, float calfact,
               std::vector<float> wlens); //bands, samples, wlens are propertes of
               the incoming hyperspectral image, which should in reality be
               detected at the first iteration of the execution loop.
4
5          //
6          // omitted framework functions
7          //
8
9          cv::Mat segment(cv::Mat inputImage, cv::Mat inputBinary, int threshold,
                bool shouldMult); //segment image based on some threshold, and
               multiply it against some other image
10         float* detectan(float *inputImage, int height, int width); //full
               calibration detection, old images with paper strip and circular
               reflectance standard
11         float* easyDetect(float *inputImage, int height); //just something to
               collect data from pre-defined calibration slab
12         bool *skinThresh(float *inputImage); //returns a binary cv::Mat object
               containing 1 where there should be skin and 0 where there should be
               not
13     private:
14         int _samples; //samples in the output image
15         int _imageInputSamples; //samples in the input image
16         int _lines; //lines per block of data
17         int _bands;
18         std::vector<float> _wlens;
19
20         float* _calarray; //contains calibration arrays after successful
               detection
21         float _calfactor; //reflectance standard factor, usually 0.4, 0.5 or
               0.99
22         float *_buffer;
23
```

```
        //omitted framework variables
};
```

Code B.6: appendix/GPU–DM/fitting.h

```
#ifndef FITTING_H_DEFINED
#define FITTING_H_DEFINED

//general fitting class, for generating interpolation tables and such
class Fitting{
    public:
        Fitting(Chromophores chrom, const std::vector<float> &wlens, float
            startwlen, float endwlen, int samples, int bands);
        void generateInterpolTables(); //creates interpolation tables for later
             interpolation
        static float** generateChromophoreMatrix(Chromophores chrom, float
            startwlen, int antWlens, int endmembers); //generates chromophore
            matrix for the specified wavelengths

    protected:
        //image properties
        int _endmembers;
        int _samples;
        int _bands;

        //interpolation properties
        int _startwlen;
        int _antWlens;

        //interpolation arrays
        float *_upperWlens;
        float *_downerWlens;

        //interpolation indices. Uppers are upper indices corresponding to
            wavelength number i from _minWlen, downers are lower indices
        float _minWlen;
        float _maxWlen;
        float _stepWlen;

        std::vector<float> _wlens;

        Chromophores _chrom;
        void lseprep();
        int *_uppers;
        int *_downers;
};

#endif
```

Code B.7: appendix/GPU–DM/hansonfitting.h

```
#ifndef HANSONFITTING_H_DEFINED
#define HANSONFITTING_H_DEFINED

#include "fitting.h"

//stage for unmixing muad output using the Lawson-Hanson algorithm
class HansonFitting { //inheritance classes omitted
    public:
        HansonFitting(Chromophores chrom, const std::vector<float> &wlens,
            float startwlen, float endwlen, int samples, int bands);
```

```
10          void generateHansonMatrix(); //create gsl matrix containing the
                chromophore matrix
11          float * doHansonFitting(float *mua); //do the actual fitting
12
13          //framework stuff
14          //
15          // omitted
16          //
17
18      private:
19          //chromophore matrix to be pseudoinverted
20          gsl_matrix *_chromMat;
21
22          //framework properties
23          //
24          // omitted
25          //
26  };
27
28  #endif
```

Code B.8: appendix/GPU–DM/melanin.h

```
1   #ifndef MELANIN_H_DEFINED
2   #define MELANIN_H_DEFINED
3   float melanin(float w);
4   enum MelaninType{PHEOMELANIN, EUMELANIN, SVAASAND};
5
6   const float MELANIN_REFERENCE_WAVELENGTH = 694.0f;
7
8   enum MelaninDetermination{USE_STRAIGHT_LINE, USE_MELANIN_CURVE};
9
10  #define PHEOMELANIN_GPU 0
11  #define SVAASAND_MELANIN_GPU 1
12  #define EUMELANIN_GPU 2
13  #define NONE_GPU 3
14
15  #define kPheo 4.780f
16  #define kEu 2.429f
17
18  const float SCALING_THRESHOLD_MELANINTYE_DETECTION = 0.90;
19
20  #endif
```

Code B.9: appendix/GPU–DM/chromophores.h

```
1   #ifndef CHROMOPHORES_H_DEFINED
2   #define CHROMOPHORES_H_DEFINED
3
4
5   class Chromophores{
6       public:
7           Chromophores(); //blood is set as standard
8           void checkAndSet(bool *val);
9           void checkAndUnset(bool *val);
10
11          void setWat(){checkAndSet(&haswat);};
12          void setKonst(){checkAndSet(&haskonst);};
13          void setMel(){checkAndSet(&hasmel);};
14          void setMethb(){checkAndSet(&hasmethb);};
15          void setBil(){checkAndSet(&hasbil);};
```

```
16          void setBet(){checkAndSet(&hasbet);};
17          void setDeoxy(){checkAndSet(&hasdeoxy);};
18          void setOxy(){checkAndSet(&hasoxy);};
19          void setCOHb(){checkAndSet(&hascohb);};
20
21          void unsetWat(){checkAndUnset(&haswat);};
22          void unsetKonst(){checkAndUnset(&haskonst);};
23          void unsetMel(){checkAndUnset(&hasmel);};
24          void unsetMethb(){checkAndUnset(&hasmethb);};
25          void unsetBil(){checkAndUnset(&hasbil);};
26          void unsetBet(){checkAndUnset(&hasbet);};
27          void unsetDeoxy(){checkAndUnset(&hasdeoxy);};
28          void unsetOxy(){checkAndUnset(&hasoxy);};
29          void unsetCOHb(){checkAndUnset(&hascohb);};
30
31          int getNumEndmembers(){return numEnd;};
32          float *getAbsArray(float w); //returns absorption array across all
                chromophores
33          float getAbs(float w, int end); //returns absorption for specified
                chromophore
34          int getMelInd(); //returns the index corresponding to melanin
35          int getMethbInd(); //returns the index corresponding to methemoglobin
36      private:
37          bool hasoxy;
38          bool hasdeoxy;
39          bool haswat;
40          bool haskonst;
41          bool hasmel;
42          bool hasmethb;
43          bool hasbil;
44          bool hasbet;
45          bool hascohb;
46
47
48
49          int numEnd;
50
51  };
52
53  #endif
```

Code B.10: appendix/GPU–DM/visualize.h

```
1   enum ShouldSave{DO_SAVE_SPECTRUM, DO_NOT_SAVE_SPECTRUM};
2
3   //image viewer. Parts of this is adapted from norsk elektro optikk
4   //for displaying physical images on input
5   class ImageViewer : public QWidget{
6       Q_OBJECT
7       public:
8           ImageViewer(QWidget *parent = NULL) : QWidget(parent){
9               QGridLayout *lay = new QGridLayout(this);
10              area = new QScrollArea;
11              lay->addWidget(area);
12              image = new QLabel;
13              image->setBackgroundRole(QPalette::Base);
14                  image->setSizePolicy(QSizePolicy::Ignored, QSizePolicy::Ignored
                        );
15              area->setWidget(image);
16              image->setScaledContents(true);
17              bar = new QStatusBar;
```

```
18              lay->addWidget(bar);
19              bar->showMessage("Ready");
20              image->setMouseTracking(true);
21              image->installEventFilter(this);
22              hyData = NULL;
23
24          };
25          void newImageArray(uchar *imgData, int numPixels, int k); //set QImage
                to incoming image array
26          void setHyData(float *hyData){this->hyData = hyData;}; //save full
                hyperspectral data in the image viewer it is set as the "main" image
                 viewer to catch all mouse click events and send data to
                invertReflectance
27      protected:
28          void paintEvent(QPaintEvent *evt);
29          bool eventFilter(QObject *object, QEvent *event); //event filter for
                catching mouse hover events and displaying pixel placements at
                bottom
30      private:
31          QStatusBar *bar;
32          QScrollArea *area;
33          QLabel *image;
34          uchar *imgData;
35          int numPixels, k;
36          float heightScale;
37          float widthScale;
38          float *hyData;
39      signals:
40          void mouseClicked(int, int, ShouldSave);
41      public slots:
42          void forwardSimulation(int, int, ShouldSave);
43
44  };
45
46  //for inputting pixel number and line number and inverting the specific pixel
47  class PixelChooser : public QWidget{
48      Q_OBJECT
49      public:
50          PixelChooser(QWidget *parent = NULL);
51      private:
52          QLineEdit *line;
53          QLineEdit *pixel;
54      private slots:
55          void buttonTriggered();
56      signals:
57          void pretendMouseClicked(int pixel, int line, ShouldSave);
58  };
59
60
61  //adapted from norsk elektro optikk
62  //stage for viewing chromophore images
63  //images are only sent to one or several imageviewer classes
64  class VisualizeStage{ //inheritance classes are omitted
65      public:
66          VisualizeStage(int numBands, int numPixels, int numLine, std::vector<
                int> bands, std::vector<float> maxVals, std::vector<float> minVals,
                bool RGBimage, QStringList winTitles, bool canShowSpectra = false,
                QWidget *parent = NULL); //canshowspectra: for saving all
                hyperspectral data to the image viewer. RGBImage: if set, will mix
                the first bands to an RGB image. Otherwise display monochromatic
                images
```

```
67          void setWindowTitle(QString title){imView->setWindowTitle(title);};
68
69          //framework
70          //
71          // omitted
72          //
73      protected:
74           bool canShowSpectra;
75
76          //indices for navigating the image
77          int k;
78          int c;
79          int t;
80
81          //image properties
82          int numBands, numPixels, interleave;
83
84          //received data
85          quint16* blockData;
86
87          //current line
88          int numLine;
89
90          //saved image data
91          uchar* imgDataRGB; //RGB
92          float* floatDataRGB; //grayscale
93
94          std::vector<uchar*> imgDataNonRGB; //the data that is to be displayed
                as RGB images
95          std::vector<float*> floatDataNonRGB; //data to be displayed as
                grayscale
96
97          //indices navigating the received images
98          std::vector<int> cs;
99
100         //framework
101         //
102         // omitted
103         //
104
105         bool RGBImage; //whether it is supposed to be an RGB image or not
106
107         std::vector<int> bands; //list of bands to be displayed (or
                chromophores or whatever)
108         std::vector<ImageViewer*> imViews; //grayscale image views
109         std::vector<float> maxVals; //list of maximum values
110         std::vector<float> minVals; //list of minimum values
111
112         bool firstTime; //whether the line is the first line received
113
114         virtual void imageStream(float* blockData, int blockLine); //inputting
                data to image
115         ImageViewer *imView; //RGB image view
116
117         float *hyData; //hyperspectral data, incoming data is saved if
                canShowSpectra is set
118 };
119
120 //subclass of VisualizeStage for converting uncalibrated hyperspectral images
        to RGB images
121 class RGBVisualize : public VisualizeStage{
```

```
122     public:
123         RGBVisualize(int numBands, int numPixels, int numLine, std::vector<int>
                bands, QWidget *parent = NULL);
124     protected:
125         virtual void imageStream(float *blockData, int blockLine);
126 };
127
128 //set 0, 1, 2, 3 in an incoming array to different colors.
129 class FourTypeVisualize : public VisualizeStage{
130     public:
131         FourTypeVisualize(int numPixels, int numLine, QWidget *parent = NULL);
132     protected:
133         virtual void imageStream(float *blockData, int blockLine);
134 };
```

B.3 CPU-DM

Parts of the CPU implementation of the skin models is shown here, with three-layered and two-layered reflectance calculations. Omitted parts are the GUI and the functions doing inversion using the above reflectance calculations. Only headers are shown, for full code refer to attached code (cpudm.cpp, getabs.cpp, nnls.cpp) if available.

Code B.11: appendix/CPU–DM/cpudm.h

```
1  #ifndef GPUDM_H_DEFINED
2  #define GPUDM_H_DEFINED
3
4  //delta-eddington, tree-layered and two-layered
5  void RflE2L3(float mua1, float mus1, float mua2, float mus2, float mua3, float
       mus3, float gcol, float *res, float d1, float d2);
6  void RflE2L2(float mua1, float mus1, float mua2, float mus2, float gcol, float
       *res, float *derivmuad, float *derivmuae);
7
8  //isotropic source function, two-layered and three-layered and one-layered
9  void RflIsoL2(float muae, float muse, float muad, float musd, float gcol,
       float *res, float *derivmuad, float *derivmuae);
10 void RflIsoL2(float muae, float muse, float muad, float musd, float gcol,
       float *res, float *deriv);
11 void RflIsoL3(float mua1, float mus1, float mua2, float mus2, float mua3,
       float mus3, float gcol, float d1, float d2, float *res);
12 void RflIsoL1(float mua, float mus, float gcol, float *res, float *deriv);
13
14 //for finding the next muad/muae given derivatives and reflectance value
15 void nextMuad(float lineData, float refl, float deriv, float *muad, float
       safeinit);
16
17 //skin data calculation
18 void calcSkinData(float lambda, float oxy1, float Bd1, float oxy2, float Bd2,
       float muam694, float bil, float car, float met, float wat, float *mua1,
       float *mus1, float *mua2, float *mus2, float *mua3, float *mus3, float *gcol
       ); //setter ekstrakromoforene i lag 3
19 void calcSkinData(float lambda, float oxy1, float Bd1, float oxy2, float Bd2,
       float muam694, float bil2, float bil3, float car1, float car3, float met2,
       float met3, float wat1, float wat2, float wat3, float *mua1, float *mus1,
       float *mua2, float *mus2, float *mua3, float *mus3, float *gcol);
20
21 #endif
```

Code B.12: appendix/CPU–DM/getabs.h

```
1
2 float getAbs(float refl, float w, float oxy, float bvf, float muam694);
3 float getAbs(int lag, float refl, float w, float oxy, float bvf, float muam694)
    ;
```

Code B.13: appendix/CPU–DM/nnls.h

```
1 //GNU scientific libraries
2 #include <gsl/gsl_vector.h>
3 #include <gsl/gsl_matrix.h>
4
5 //solve chromophores * x = b in least squares sense with x > 0
6 bool nnls_hanson(gsl_matrix *chromophores, gsl_vector *absspec, int param, int
    obs, gsl_vector *x, double *std);
```