FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Spectral object reflectance estimation using smartphones

Miguel Cruz Fernandes

Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Luís Filipe Teixeira

March 27, 2018

Spectral object reflectance estimation using smartphones

Miguel Cruz Fernandes

Mestrado Integrado em Engenharia Informática e Computação

March 27, 2018

Abstract

During the past years, the increasing development of technology has been mesmerizing. It is normal that new things are discovered every day or developed even further, for greater achievements. One product that has suffered an exponential growth is the smartphone. More and more it is capable of doing tasks that were only assigned to specific devices. Nowadays, a smartphone is easily accessible to the vast majority of people and it is only normal that it will incorporate increasingly different functions throughout the years to come.

Two of the most noticeable attributes in a smartphone are its camera and its screen. As the new generations of smartphones keep coming, the gap between professional high-end equipment and smartphone attributes gets smaller. Even though the quality is still not the same, it is not bold to try to figure out how close it gets.

The study behind the thesis does not aim to replicate the exact same results as high-end devices do, when it comes to spectral reflectance estimation, but rather see how close it can get, utilising a smartphone. The smartphone camera and screen are two of the three components needed to estimate a spectral reflectance, being a surface the third component. If the camera and the screen were to be used in an object's spectral reflectance estimation, how close would the results get to a spectrophotometer measurement? Being able to utilise a hand-sized, inexpensive device instead of heavy, bulky and expensive equipment for the same end would prove to be very useful and practical. There have been claims that it is already possible to get results as good as high-end devices, using a smartphone. This thesis is focused on studying if this really seems to be possible.

An Android application was developed for the smartphone to be able to use the screen as an illuminant and allow for different colors of the visible spectrum to be used as output, at will. This way, it is possible to illuminate an object across distinct wavelength ranges and take a picture of said object using the front camera. Only the screen calibration was implemented, to measure the spectral distribution of the smartphone, with a spectroradiometer. The spectral object reconstruction was done using the measurements taken, as the illuminant, a standard RGB camera sensitivity and an objects reflectance data set (Munsell color data set). The methods used for estimation were the Wiener estimation, a linear estimation and principal eigenvectors estimation. In order to evaluate the spectral difference and colorimetric performance, the methods used were the root mean square error and the CIEDE2000 color difference method, respectively.

The results proved to be promising, although the introduction of noise in the main equation of the simulation still leaves open how well it would behave in a real conditions scenario.

Resumo

Durante os últimos anos o avanço do desenvolvimento de tecnologia tem sido fascinante. É normal que novos processos sejam descobertos todos os dias ou mais deselvolvidos, para maior consquistas. Um produto que sofreu um crescimento exponencial foi o *smartphone*. É capaz de realizar cada vez mais tarefas que eram apenas atribuídas a aparelhos específicos. Nos dias de hoje, um *smartphone* está acessível à grande maioria das pessoas e é apenas normal que incorpore mais, e uma variedade maior de funções nos anos que se aproximam.

Dois dos atributos com maior destaque num *smartphone* são a sua câmera e ecrã. À medida que as novas gerações de *smartphone* vão chegando, a separação entre equipamento profissional e os atributos de um *smartphone* vai diminuindo. Apesar da qualidade não ser a mesma, não é uma perda de tempo tentar descobrir o quão perto pode chegar.

O estudo desenvolvido nesta tese não tenta replicar os mesmos resultados que equipamento de alta tecnologia, relativamente à estimativa de reflectância espectral, mas sim ver o quão perto se consegue chegar, utilizando um *smartphone*. A câmera e ecrã de um *smartphone* são dois de três componentes utilizados para estimar uma reflectância espectral, sendo qualquer superfície o terceiro. Se a câmera e ecrã fossem utilizados para uma estimativa de reflectância espectral de um objecto, quão perto poderiam os resultados chegar, comparativamente às medições efectuadas por um espectrofotómetro? A possibilidade de usar um aparelho muito mais barato, que cabe na palma da mão, em vez de usar equipamento pesado, volumoso e não tão acessível financialmente, para o mesmo fim, seria extremamente útil e prático. Apesar de não confirmado, foi afirmado que já é possível obter resultados tão fidedignos com um *smartphone*, como com os aparelhos de alta tecnologia designados para o mesmo fim. Esta tese é focada em estudar se isto realmente é possível.

Uma aplicação Android foi desenvolvida para permitir que o ecrã seja usado como iluminante, podendo percorrer diferentes cores, à escolha. Desta forma, é possível iluminar um objecto em intervalos de comprimento de onda distintos e tirar fotos do mesmo, usando a câmera frontal. Apenas foi implementado o processo de calibração para medir a distribuição espectral proveniente do ecrã, usando um espectrorradiómetro. A reconstrução espectral foi aplicada usando: as medições registadas, como iluminante; a sensibilidade de uma câmera RGB comum; e uma *data set* de reflectância de diferentes objectos (*Munsell color data set*). Os métodos usados para a estimativa foram a estimativa Wiener, uma estimativa linear e a estimativa de autovectores. Para avaliar a diferença espectral e a performance colorimétrica, os métodos usados foram a raiz do valor quadrático médio (rms) e o método de diferença de cor CIEDE2000, respectivamente.

Os resultados foram promissores apesar da introdução de interferências no cenário principal ainda deixar em aberto o quão bem resultaria num cenário em condições reais.

Acknowledgements

I would like to thank Professor Dorit Merhof, the Institute of Imaging & Computer Vision and RWTH Aachen University, for giving me the possibility of doing this project under their supervision and for all the help in setting it up.

I would like to thank Tarek Stiebel, for his patience, help and availability in coordinating my thesis throughout these five months.

I would like to thank my home coordinator Professor Luís Teixeira and the Faculty of Engineering of the University of Porto for supporting my work abroad.

I would like to thank my mother Paula and my father Eduardo for the education given to me, for the constant attempts to motivate me and for giving me the possibility of studying and working abroad. I would also like to thank my sister Leonor, my grandparents (the ones I am still able to hug and the ones I am not), my godparents and the rest of my family, for being so present during my growth as a person.

I would like to thank all my friends, from University and high school, along with the people I've met from Aachen, Almería, Munich and various parts of the world, with a special mention to Gabriel Ribeiro, for keeping my (in)sanity at the right levels.

Miguel Fernandes

"Give a man a fire, and he's warm for a night.
Set him on fire, and he's warm for the rest on his life."

Terry Pratchett

Contents

List of Figures

LIST OF FIGURES

List of Tables

Abbreviations

ADB	Android Debug Bridge
JDK	JAVA SE Development Kit
nm	Nanometers
IDE	Integrated Development Environment
XML	Extensible Markup Language
rms	Root Mean Square
JND	Just Noticeable Difference

Chapter 1

Introduction

Nowadays, measuring the spectral reflectance functions of surfaces is an essential requirement in many tasks. It is possible to analyse the measurement taken and get information that you could not get in any other way. The most intuitive task is probably object classification, e.g. making a statement about the age of food. Another prominent example is the creation of high-fidelity color images, in fields such as medicine [MM99] and artwork [HK07, IB99].

The process to measure a spectral reflectance consists of three main components: a surface to be measured, an illuminant and a camera. The surface is what we want to analyse, which can be anything. Having a smartphone with a screen that can be used as an illuminant and a front camera, fulfills the missing components. It is a much more portable device, cheaper and more accessible than any high-end equipment designed for the same purpose.

To be able to take pictures of a surface, to then analyse its' spectral reflectance is a straightforward process. A radiation is incident onto an object, which then reflects or absorbs said radiation, according to the object's properties and considering the wavelength of the illuminant. A picture is taken, to be able to analyse the spectral reflectance and estimate it. This process is repeated over whichever wavelength of the spectrum the camera can detect. It is important to estimate the curve, since it will contain noise such as natural light, for instance, to try to filter it out as best as possible. With an RGB (frontal) camera and a screen to serve as illuminant in a smartphone, the process can be replicated, by sweeping through the visible spectrum and taking pictures of the illuminated object. This scenario is shown in Figure 1.1.

In this thesis, only the potential of using the screen as an illuminant is studied and tested. The application was still developed to its fullest capabilities, to represent the whole process described next. Thanks to The Institute of Imaging & Computer Vision, in RWTH Aachen University, who supported the development of this thesis, access to a spectroradiometer was provided, making it is possible to take accurate measurements, later to be used in the development of the results.

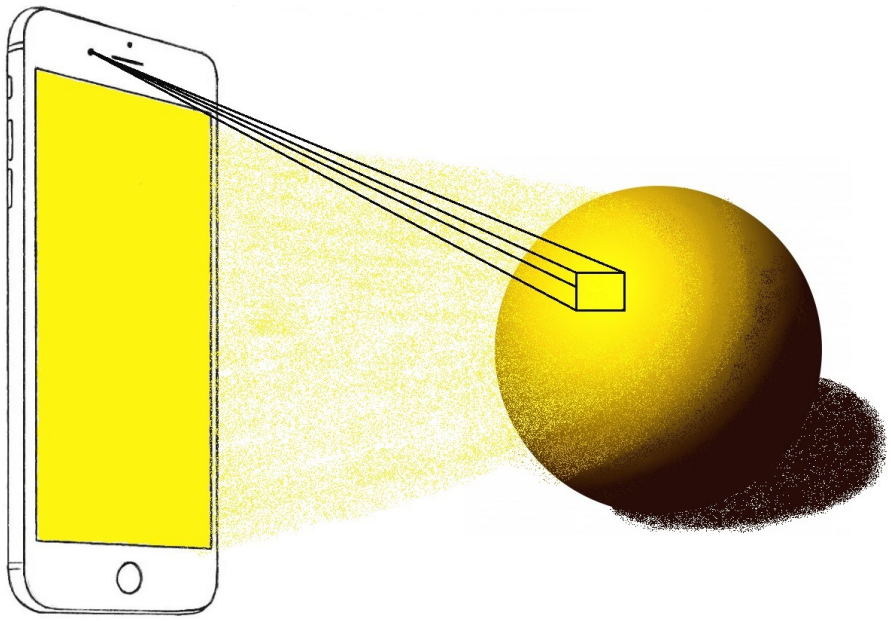

Figure 1.1: Picture being taken of an object illuminated by a smartphone's screen

1.1 Context

Spectral object reflectances are usually measured in controlled environments. The reason for this is that when viewing an object, the spectral stimulus entering the measurement device will actually consist of two parts: the spectral object reflectance and the spectral power distribution of the illuminant. In order to separate both of them, the illuminant needs to be known.

The precise spectrum of the light reflected by a surface is usually measured using costly, high-end devices, such as spectrophotometers. Such devices are capable of directly measuring the wavelength dependent intensity of the incoming light. Loosely speaking, those devices allow a narrowband sampling of the incoming light. As a typical illuminant, perfect white is usually aimed at. In other words, a broadband illumination.

A cheaper way to measure the spectral object reflectance is to use narrowband illumination and a broadband sensing device. As an example, consider the surface of interest viewed by a monochrome camera, while the surface is illuminated with red light. The illuminant basically defines the sampling point of the spectral object reflectance. Sweeping through the entire wavelength range equals to a complete sampling of the spectral object reflectance. Compared to the devices mentioned before, smartphones are able to provide an inexpensive, fast and practical solution.

The human eye, comparable to RGB cameras, uses three types of cone receptors to process spectral data over the visible spectrum, whose wavelength ranges from 380 to 780 nm, resulting in a three-channel color image. Being able to obtain this data allows for calculations to be performed in order to reconstruct spectral object reflectances.

1.2 Motivation and Goals

While the measurement process itself is well known and straightforward, the question arises on how well this task of spectral object reflectance measurement can be performed with standard consumer hardware as the one used in smartphones.

Considering that technology keeps leaping forward, its development allows for new possibilities everyday. The smartphone market has had an explosive development during the past years [K.07], so it is only normal that people try to use (and abuse) its capabilities. It is possible to use a smartphone as a spectrophotometer since it contains enough components to do so. The interesting part of this is to see how well it can do.

An application was created in order to control the smartphone to illuminate, using the device's screen. The main part of the application is that it has to be capable of going through different colors of the visible spectrum, at command, and take pictures in between. This way, the pictures can be saved and then transferred to a computer, for further analysis.

To better understand which screen output we are getting, it is crucial that the spectral power distribution of the light irradiated from the screen is precisely characterized, through a calibration process, with a spectroradiometer, since machinery and the human eye perceive these colors in different ways. Having the spectral distribution of each radiation, we can use this data as part of the simulated spectral estimation. There are different methods to reconstruct the spectral curve, which will be compared.

1.3 Structure of the thesis

Besides the introduction itself, this thesis contains 3 more chapters. On chapter 2 we have a description of the fundamentals associated with the thesis. Chapter 3 talks about the choice of tools for the study and implementation. Chapter 4 refers to the implementation. This includes all the android development, as well as the device's screen illumination calibration, the spectral object reflectance estimation process and the results associated with it. Chapter 5 refers to the conclusion and future work.

Chapter 2

Fundamentals

In this chapter the fundamental knowledge associated with the thesis will be approached. We will start by defining spectral object reflectance, followed by some concepts on how the human eye perceives color. Finally, the process for spectral object reflectance reconstruction will be described, along with the algorithms used to calculate it and the different error metrics.

2.1 Spectral Object Reflectance

Reflectance is the fraction of incident radiant flux that is reflected by a surface and is given by

$$R = \frac{\Phi_{e,r}}{\Phi_{e,I}}, \tag{2.1}$$

where $\Phi_{e,r}$ is the radiant flux reflected by that surface and $\Phi_{e,I}$ is the radiant flux received by that surface.

Spectral reflectance curve is the plot of the reflectance as a function of the radiant electromagnetic wavelength. Different surfaces reflect or absorb a radiation in different ways. The reflectance properties of a given object or surface depend on the material it is made of and the circumstances it is measured in. By analysing the spectral object reflectance, it is possible to identify different surface features or materials.

2.1.1 Electromagnetic Spectrum

Light is radiation in the form of electromagnetic waves that make vision possible to the human eye [OR06]. In 1666, Sir Isaac Newton (Figure 2.1) conducted an experiment that showed that white light, such as sunlight, can be divided into several types of coloured light, by placing a glass prism in front of a light beam (Figure 2.2). Seeing that objects appear to be the same colour as the beam illuminating them, and realising that a beam of coloured light will not change independently

Figure 2.1: Sir Isaac Newton

of how many times it is reflected or refracted, led Newton to conclude that color is not a property of the object, but of the light that is reflected by it[I.72].

The human eye is only sensitive to a certain part of the electromagnetic spectrum, the visible spectrum. In frequency, it is located above infrared and bellow ultraviolet radiations, with a spectral range between 380 to 780 nm. Electromagnetic radiation can be classified by its wavelength or frequency, as shown in Figure 2.3.

The electromagnetic spectrum covers electromagnetic waves with frequencies from below one hertz to above 10^{25} hertz. Starting at a low frequency, the types of electromagnetic radiation are divided into the following separate bands[Meh11]:

1. Radio waves
2. Microwave radiation
3. Infrared radiation
4. Visible radiation
5. Ultraviolet radiation
6. X-ray radiation
7. Gamma radiation

Exposure from high ultraviolet to bigger frequencies is considered to be a health hazard, because of the chemical reactions that can occur, causing DNA damage and cancer, for instance. The radiation from visible light or lower frequencies cannot cause such effects.

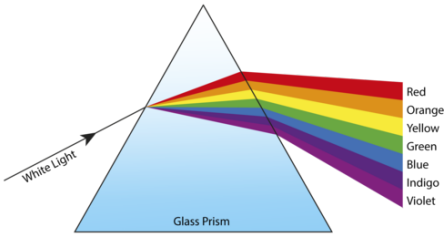

Figure 2.2: Prismatic dispersion of white light

2.1.2 Usability

The spectral reflectances have many different uses, namely in agriculture and medicine, for instance. Figure 2.4 shows typical reflectance curves of water, vegetation and soil. Healthy vegetation has a smaller reflectance in the visible spectrum, which then increases when it starts nearing the infrared. As a practical example, stressed vegetation or vegetation in different life lengths can also be identified due to its significantly lower spectral reflectance in the infrared region (Figure 2.5). This data can be used to differentiate said vegetation in order to improve farming activities (for commercial or personal use or simply for paisagistic development).

2.2 Human vision

The human eye, with normal vision, has two types of light sensitive cells, known as the rods and the cones. The rods perceive lightness or darkness in darker environments, which is known as scotopic vision, and the cones perceive color in brighter environments, which is known as photopic vision. In environments with intermediate brightness between scotopic and photopic vision, the cones and rods function in order to provide what is known as mesopic vision[RWGH11, S.00].

There are three kinds of cone cells that sense light in a long, middle and short wavelength, with the ratio of around 32:16:1, respectively. The normalised spectral sensitivity of the cone cells can be visualised in Figure 2.6, where S, M and L correspond to the short (420-440 nm), middle (530-540 nm) and long (560-580 nm) wavelengths, respectively[OR06]. These individual spectral sensitivities of the three kinds of cone cells render three stimulus values, also known as tristimulus values.

Having the spectral power distribution of incident light as the function $f(x)$, the linear model that contains the cone responses can be represented as:

$$c = \int_{\lambda} S_i(\lambda) f(\lambda) d\lambda, \tag{2.2}$$

where $S_i(\lambda)$ corresponds to the spectral sensitivity function of the ith cone and $i \in 1,2,3$.

Considering N uniform wavelengths samples over the visible region range, equation 2.2 is written as:

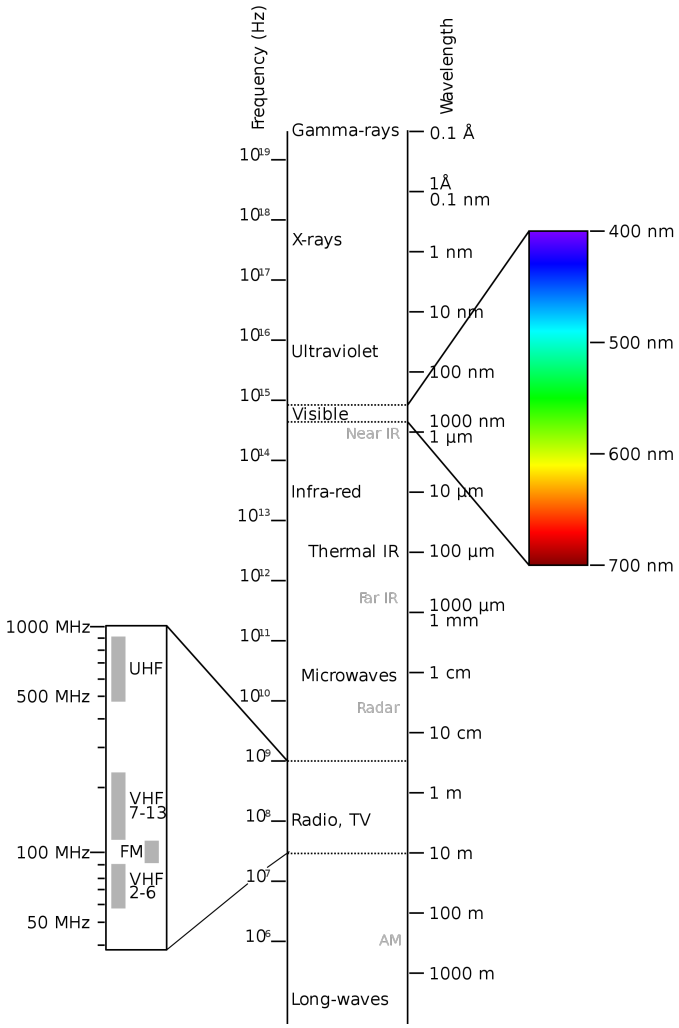

Figure 2.3: Wavelengths and frequencies of electromagnetic radiation and light

$$c = \int_{i=1}^{N} S_i(\lambda_i) f(\lambda_i) d\lambda, \tag{2.3}$$

where λ_i corresponds to the uniformly spaced wavelength and $i \in 1,2,3$.

2.3 Spectral Object Reflectance Reconstruction

A multi-channel camera system obtains a multi-channel image using a single-channel camera that filters the incoming color signal through a set of color filters. The objective of the reconstruction is to make estimations from a low-dimensional data into high-dimensional data. To be able to accurately recover the spectral reflectances of an object's surface, the system responses should be known[Y.99, IB99, SX04].

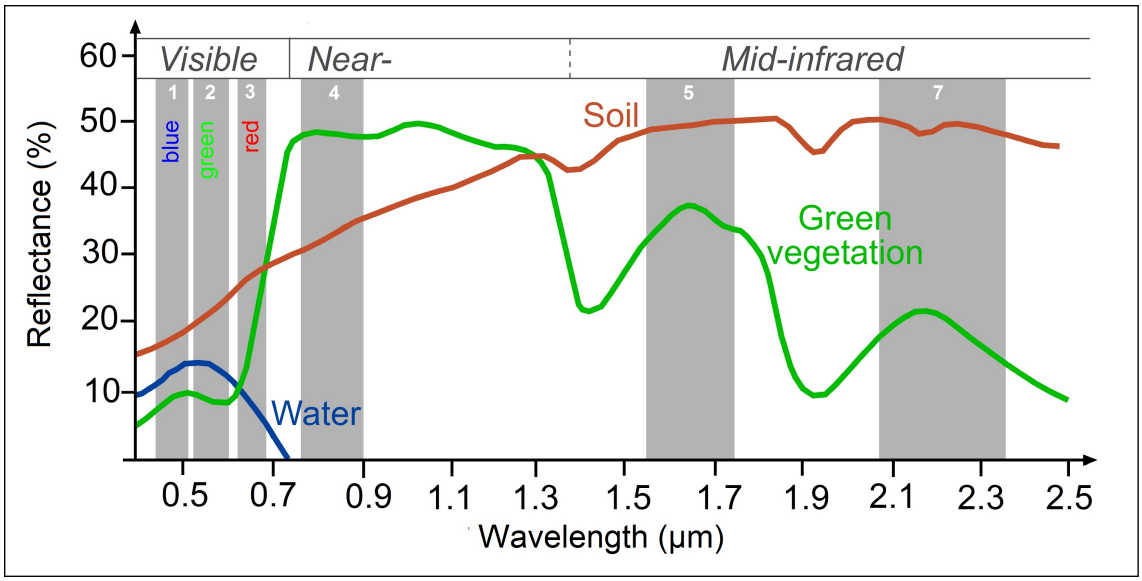

Figure 2.4: Spectral reflectances of water, vegetation and soil

Being **L** the spectral power distribution of an illuminant, **r** the spectral reflectance of an object, and **s** the spectral sensitivity of the camera, then the response **y** of the channel **k**, can be represented as

$$y_k = \int_\lambda L(\lambda) s_k(\lambda) r(\lambda) d\lambda, \tag{2.4}$$

where λ is the wavelength variable and k=1,...,m.

By equally sampling the wavelength range into t regions, we can present equation 2.4 as vectors and matrices. In this form, the equation is

$$y = FO, \tag{2.5}$$

where **y** is an i-dimensional column vector for the multi-channel image and **O** is an s-dimensional column vector which corresponds to the spectral reflectance of the object. **F** corresponds to a linear $(i \times s)$ matrix and is represented as:

$$F = UES, \tag{2.6}$$

where,

$$U = [k_1, ..., k_m]^t. \tag{2.7}$$

The column vector k_m corresponds to the response of the mth filter and $[]^t$ is a transpose. In equation 2.6, **E** stands for the $(s \times s)$ diagonal matrix for the spectral power distribution of the illuminant, and **S** stands for the $(s \times s)$ diagonal matrix for the spectral sensitivity of the camera.

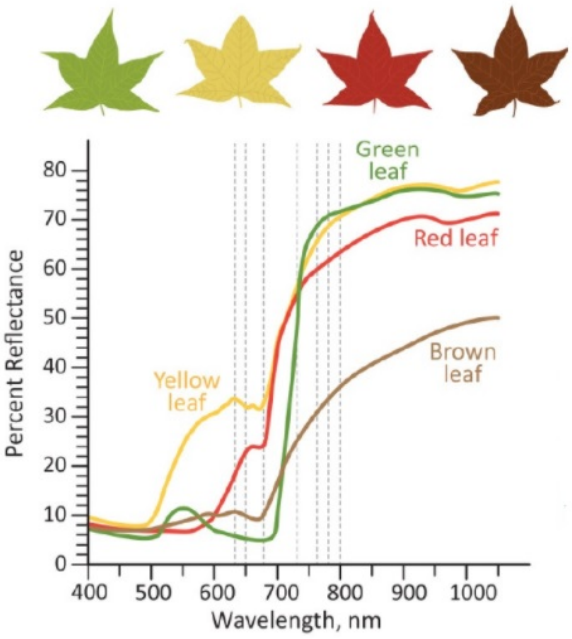

Figure 2.5: Spectral reflectances of different vegetation

There are several different methods to estimate an object's spectral reflectance. The Wiener estimation method minimizes the mean square error (MSE) between the original spectral reflectance and the estimated one. The linear estimation utilises a set of known spectral reflectances with already known camera responses[CD04] to minimize the least square error. The principal eigenvectors works as a linear transformation, but only changes by a scalar factor.

2.3.1 Wiener estimation

The Wiener estimation model uses three matrixes: an estimation matrix, the spectral sensitivity of the camera sensors and the system noise. These matrixes have been proved to be a crucial point, regarding the accuracy of the estimation[HHH+]. To reconstruct the spectral object reflectance, there needs to be prior information, such as sensors' information and the illumination. This results in the method being quite difficult to apply.

Adding camera system noise to equation 2.4, results in:

$$y_k = \int_\lambda L(\lambda)s_k(\lambda)r(\lambda)d\lambda + e_k, \tag{2.8}$$

where e_k corresponds to the noise associated to the channel k. Which means that equation 2.6 would be:

$$F = UES + e_k, \tag{2.9}$$

Using the Wiener estimation method, the reconstructed spectral reflectance can be found as

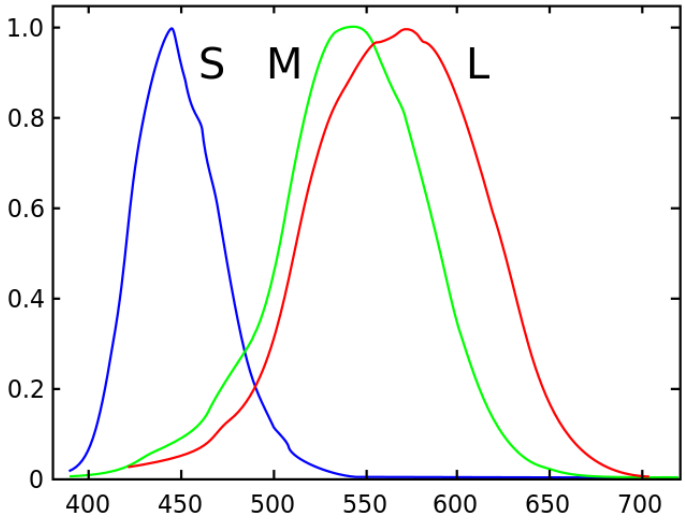

Figure 2.6: Normalised spectral sensitivity of human cone cells of short, middle and long wavelengths

$$\hat{r} = GF, \tag{2.10}$$

where F is the response vector (equation 2.9) and G is the estimation matrix. On the spectral object reflectance reconstruction, G is used to minimize the mean square error between the original spectral object reflectance r and the estimated one $\hat{r}$

$$e = \{|r - \hat{r}|\}, \tag{2.11}$$

where $\{.\}$ signifies that the error e is normalised. The estimation matrix G is defined as:

$$G = R_{rv} R_{vv}^{-1}, \tag{2.12}$$

where

$$R_{rv} = \{rv^t\}, R_{vv} = \{vv^t\}. \tag{2.13}$$

In equation 2.12 and equation 2.13, R_{rv} and R_{vv} are the correlation matrices of the spectral reflectance of the test and noise, respectively. If these two matrices are equal to the original matrices of spectral reflectance and noise, the error will be minimized.

2.3.2 Linear estimation

The main objective of the linear estimation is to minimize the least square error for a set of known spectral reflectances functions, with previous knowledge of the camera responses. This way, we can reconstruct the spectral reflectance simply.

Having a training set S where x_i and y_i correspond to the camera responses and the spectral reflectances, repectively, can be represented as:

$$S = (x_i, y_i), \tag{2.14}$$

where $i = 1...m$.

Considering a different set of camera responses x, the spectral reflectance at wavelength can be represented as:

$$y_i = f_i(x), \tag{2.15}$$

where $i = 1...n$ and $f_i(x)$ are the associated functions $f_1(x)$,...,$f_n(x)$[ZW08].

The objective function L is given by:

$$L(f, S) = |r - wX|^2, \tag{2.16}$$

where X is an $(m \times n)$ matrix, having each row n amount of camera responses and r corresponds to the spectral reflectance vector.

Given the input vector x, which contains the camera's channels responses, the following linear form is presented:

$$f(x) = w, x. \tag{2.17}$$

In order to minimize the mean square error of equation 2.16, the solution of w is searched.

Finally, the estimated spectral reflectance $\hat{r}$ can be found using the following equation:

$$\hat{r} = (X^t X)^{-1} X^t r \tag{2.18}$$

2.3.3 Principal eigenvectors

An eigenvector of a linear transformation is a vector different of zero that changes by a scalar factor, when that linear transformation is applied to it. Having equation 2.4, we have the matrix form:

$$p_r = y\lambda, \tag{2.19}$$

where p_r represents a vector for the image data and λ the illuminant weights. Working on a finite-dimensional vector space, the linear transformation p_r can be represented as:

$$p_r = xA, \tag{2.20}$$

where A corresponds to the weights for a spectral reflectance and x corresponds to y on equation 2.4.

Having equation 2.19 and equation 2.20, we have:

$$xA = y\lambda. \tag{2.21}$$

If x is nonsingular, A can be obtained by inverse matrix of x. Using A, the spectral object reflectance can be estimated.

2.4 Error Metrics

There are several ways to evaluate the accuracy of a spectral object reflectance's estimation. The root mean square (rms) is one way to find the spectral error and the CIELAB color space is commonly used to find the colorimetric error.

2.4.1 Root Mean Square (rms)

The spectral root mean square error is one of the methods used to compare the differences between an original reflectance $\mathbf{r}$ and its estimate $\hat{\mathbf{r}}$. It is calculated as

$$rms = \sqrt{\frac{\sum_\lambda [r(\lambda) - \hat{r}(\lambda)]^2}{N}}, \tag{2.22}$$

where N = 401, which is the spectrum wavelength of visible light (380 to 780).

2.4.2 CIE XYZ and CIELAB color spaces

A color space is used to link distributions of wavelengths in the electromagnetic visible spectrum and the physiological colors perceived by the human eye, typically in terms of tristimulus values. These tristimulus values associated with a color space can be conceptualised as amounts of three primary colors in a tri-chromatic color model.

Having the XYZ tristimulus values, we can calculate the colorimetric error after converting them to the CIE L*a*b* (CIELAB) color space, specified by the International Commission on Illumination (CIE), in 1976 [G.82], calculating as:

$$L^* = 116 f\left(\frac{Y}{Y_n}\right) - 16 \tag{2.23}$$

$$a^* = 500\left(f\left(\frac{X}{X_n}\right) - f\left(\frac{Y}{Y_n}\right)\right) \tag{2.24}$$

$$b^* = 200\left(f\left(\frac{Y}{Y_n}\right) - f\left(\frac{Z}{Z_n}\right)\right) \tag{2.25}$$

where

$$f(t) = \begin{cases} \sqrt[3]{t} & ,t > 0.008856 \\ 7.787\text{t} + \frac{4}{29} & \text{, otherwise.} \end{cases}$$

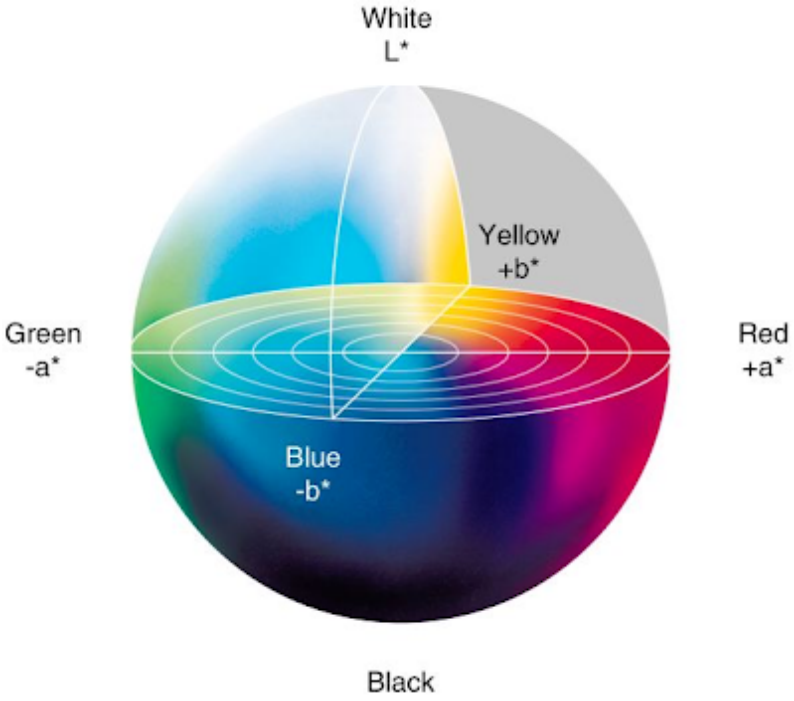

Figure 2.7: Representation of CIELAB color space

XYZ are the tristimulus values, which are obtained, once again, by applying the equation 2.4. X_n, Y_n and Z_n are the tristimulus values of the reference white point. The values are normalized so that $Y_n = 100$.

For the example of illuminant D65, that portrays standard illumination conditions at open-air, with normalization $Y_n = 100$, the values are

$$\begin{aligned} \mathrm{X}_n &= 95.047, \\ \mathrm{Y}_n &= 100.000, \\ \mathrm{Z}_n &= 108.883 \end{aligned}$$

Regarding the three coordinates of CIELAB, which is represented in Figure 1.1,

- L* represents the lightness of the color, where L* = 0 corresponds to black and L* = 100 corresponds to white,
- a* represents the degree of red/magenta versus green (a* < 0 = green value or a* > 0 = red/magenta value) and
- b* represents the degree of yellow versus blue (b* < 0 = blue value or b* > 0 = yellow value).

Table 2.1: Practical interpretation of ΔE^* color difference by Abrardo *et al.* [AVMA96]

ΔE^*	Effect
$0 < 1$	Limit of perception
$1 < 3$	Very good quality
$3 < 6$	Good quality
$6 < 10$	Sufficient
> 10	Insufficient

When comparing two colors, such as $[L_1^*, a_1^*, b_1^*]$ and $[L_2^*, a_2^*, b_2^*]$, the CIELAB color difference is calculated as the Euclidean distance in the CIELAB space, as follows [IB99, KOO+99]:

$$\Delta E^* = \sqrt{(\Delta L^*)^2 + (\Delta a^*)^2 + (\Delta b^*)^2}, \tag{2.26}$$

where

$$\Delta L^* = L_1^* - L_2^*$$
$$\Delta a^* = a_1^* - a_2^*$$
$$\Delta b^* = b_1^* - b_2^*$$

The XYZ values are directly related to the spectral power distribution of the colored light. What this means is that when these tristimulus values are changed, the human eye will perceive a difference in color only after the difference reaches a certain amount. The interpretation of ΔE^* is hardly perceptually meaningful. According to Kang [R.97], the Just Noticeable Difference (JND) is equal to 1. On the other hand, Mahy *et al.* [MVEO94] found a JND of 2.3. Two other practical interpretations can be found in the work of Abrardo *et al.* [AVMA96] and Hardeberg [Y.99] and are represented in Table 2.1 and Table 2.2, respectively.

2.4.3 CIEDE2000 color Difference

The CIEDE2000 definition [SD05] is considered to be the CIE recommended color difference formula, which includes new terms to improve the colorimetric error ($\Delta\ E_{00}$*) [14201, SWDA05]. The corrections added are:

- A hue rotation term, to deal with the blue region (hue angles around 275^o)
- A compensation for neutral values

Table 2.2: Practical interpretation of ΔE^* color difference by Hardeberg [Y.99]

ΔE^*	Effect
< 3	Hardly perceptible
$3 < 6$	Perceptible, but acceptable
> 6	Not acceptable

- A compensation for lightness
- A compensation for hue
- A compensation for chroma

Comparing two colors, such as $[L_1^*, a_1^*, b_1^*]$ and $[L_2^*, a_2^*, b_2^*]$, the modifications result in the equation displayed bellow:

$$\Delta E_{00}^* = \sqrt{\left(\frac{\Delta L'}{k_L S_L}\right)^2 + \left(\frac{\Delta C'}{k_C S_C}\right)^2 + \left(\frac{\Delta H'}{k_H S_H}\right)^2 + R_T \frac{\Delta C'}{k_C S_C} \frac{\Delta H'}{k_H S_H}^2} \tag{2.27}$$

Chapter 3

Measurement and Software tools

3.1 Measurement tools

In this section the tools provided by the institute will be described. Since the final objective of this thesis is the comparison of our smartphone device to higher cost devices, it was crucial to have access to such.

3.1.1 Spectroradiometer

A spectroradiometer is a device designed to measure the spectral power distribution of incoming light. The output provided by this device, controlled with Matlab, is utilized to create a color graph in a given wavelength.

Any spectroradiometer will integrate four basic units [Ltd14]:

1. The input optics, to gather the electromagnetic radiation from the source.
2. A monochromator, to separate the radiation into its component wavelengths.
3. A detector, to measure the radiation at each wavelength.
4. A control system, to define gathered data and store it.

Regardless of how technically good the system is, the produced data is only as good as the calibration of the instrument and the measurement conditions. This means that a lot of factors contribute for a better measurement, but not all can be controlled, such as noise from the detector, internal electronics, humidity and temperature variations, for instance.

These devices are used in different applications, such as plant research and development and LED measurement [Sci15].

3.1.2 Spectrophotometer

The measuring process of a spectrophotometer is as follows:

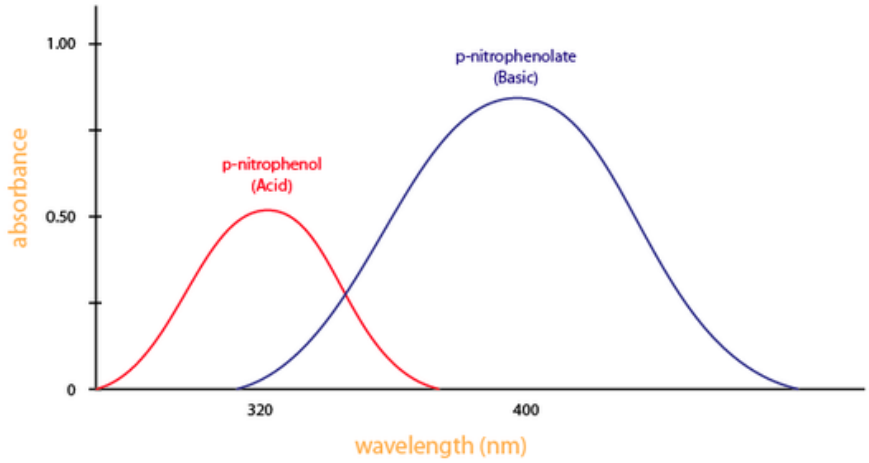

Figure 3.1: Absorbance of p-nitrophenol and p-nitrophenolate

1. A light source in a certain wavelength shines onto the surface.
2. The surface reflects or absorbs light.
3. The device detects how much of the incident light is reflected from the surface.
4. The device converts how much light the surface reflected into a numeric value.

This allows for a graph to be drawn, in the wavelengths that the light source provided. As an example, if we take the visible light spectrum and measure every nanometer that is reflected/absorbed, it is possible to determine the intensity of each wavelength in the resulting graph.

With this in mind, there are several applications for this device, such as medicine or agriculture [C.15], among others. Since different compounds absorb best at different wavelengths, it is possible to identify them, as in Figure 3.1, for instance. Nitrophenols are poisonous [Boo00] and contaminate soil near former explosive or fabric factories and military plants. Identifying these substances allows for a solution to be developed based on the data, for healthy and safe soil usage.

3.2 Software tools

3.2.1 Android vs iOS

Although the iOS platform has a solid environment when it comes to development, Android was chosen for this project. Having close to no background in smartphone application development was also part of this choice. Android supports the developer with very useful tools such as the Integrated Development Environment (IDE) named Android Studio. Knowing that this has been a replacement for Eclipse and that it utilizes Java, which I was more familiarized with, finalised the decision.

3.2.2 Android Studio

Android Studio is the official IDE for Google's Android operating system. Android Studio supports Java and the layouts are designed in Extensible Markup Language (XML), which are very easy to manipulate due to its' drag-and-drop visual editor. It is a tool with great flexibility and support, providing simple tutorials and general information to new developers [Goo13].

3.2.3 MATLAB

MATLAB was developed by MathWorks which combines a desktop environment with a programming language that expresses matrix and array mathematics [Mat84].

Since most of the provided algorithms were written for MATLAB, as well as the control for the measurement tools, it is essential to utilise this software, not only for its flexibility but also for the easier communication with Java.

MathWorks helps users allowing files to be shared for common usage, providing support and accurate data such as spectral and *XYZ* color functions [Mat10] and color properties toolboxes [Wag07], which prevent errors in large but precise calculations.

3.2.4 Java

Java is a concurrent, class-based and object-oriented programming language [Gos15]. It is very useful for multiple platform use, since it doesn't require recompilation, as long as said platform supports the language.

It gives the user access to different libraries, two of which had a bigger role in this project.

3.2.4.1 MatlabControl

MathWorks supports calling and using Java objects from Matlab, but it does not support calling Matlab commands from Java. MatlabControl allows the user to call Matlab commands from Java.

It is essential for the measurements control as well as managing results, considering that it is simpler and wiser to use minimum number of different programming languages. It could be possible to communicate with Matlab through Python, for instance, but it would disregard the previous statement, since Python would not be used for anything else.

3.2.4.2 JFreeChart

JFreeChart is a Java chart library that allows the display of charts or graphs [Gil00]. When in need of visualising multiple results, it is easily accessible to work with. It has an extensive feature set, of which the line graph proves to be the most useful for the visualisation and comparison of results.

3.3 Conclusions

Overall, the thesis mostly utilizes Java and Matlab. The interaction between the two acts as a user interface to facilitate obtention of faster results. It gives the possibility to either analyse a small amount of the data, as well as much larger amounts, with ease.

Minimum human interaction with Matlab is required, but it is also possible to deeply analyse results, if desired, based on the fact that the Matlab session is opened and available for use during the entire measurement/calculations process.

Chapter 4

Implementation

This chapter is divided into seven different parts: the Android application development, the smartphone's screen calibration, the spectral estimation methods, the spectral and colourimetric difference, the object reflectance reconstruction, results and conclusions.

4.1 Android application development

The main objective of the application is to be able to act as the illuminant and take a picture of the object reflecting the incident light. The smartphone is held with its display towards the desired surface, having the screen to be fully illuminated with predefined sets of chromatic colours. After the display has been set to a specific colour, a picture can be taken, which is then automatically saved to a specific folder in the smartphone's memory. This process is to be repeated with as many different colours as the user desires, with the intention to then transfer the pictures taken to a computer, for further work and is represented in Figure 1.1.

4.1.1 Initial mockups

Initially, some simple mockups were created to try to understand what was important for the application to contain.

The initial idea was to have a main screen, as presented in Figure 4.1, where you can select predefined color sets to go through, as an illuminant. There are options for 1, 3, 6 or 9 distinct colors, which a user can select and define. There is also a gallery and a transfer button. The first one, to browse through the taken pictures and the second one to be able to select the desired pictures to be able to transfer it to a computer.

Figure 4.2 shows the outcome of pressing the "9" button in the main screen. In this screen, the user can visualise the color hexcodes defined for the picture taking process, as well as start said process, by clicking the "Begin" button.

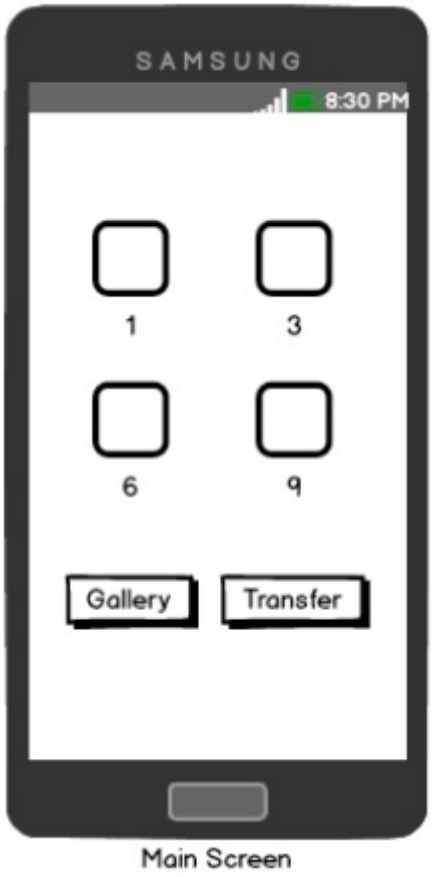

Figure 4.1: Main Screen Mockup

Although the current prototype (which will be thoroughly described next) is not working as such, pressing the color hexcodes buttons would lead the user to another screen that would allow to define the exact colors to be used, as well as their showing sequence, as seen in Figure 4.3.

In this screen, not only can the color be selected in the RGB color model, but also as a hexcode, allowing a preview of said color. Upon confirmation, the user is returned to the previous screen, enabling the definition of the remaining colors.

The gallery (Figure 4.4) and transfer (Figure 4.5) screens are almost identical, with the only difference being the selection on the transfer screen, since it enables the user to select the desired pictures, before transfering them.

The main part of the application, which is the screen serving as illuminant, is only showed in the prototype section, since it only contains the full screen illuminated with the selected color.

A last mockup that was designed is for the screen immediately after the picture taking process is finished.

Given that, for instance, the "9" button is selected in the main screen (Figure 4.1), followed by the beginning of the activity, only nine colors would be available. This means that when changing to the next color, after going through the first nine, which does not exist, the photo process is

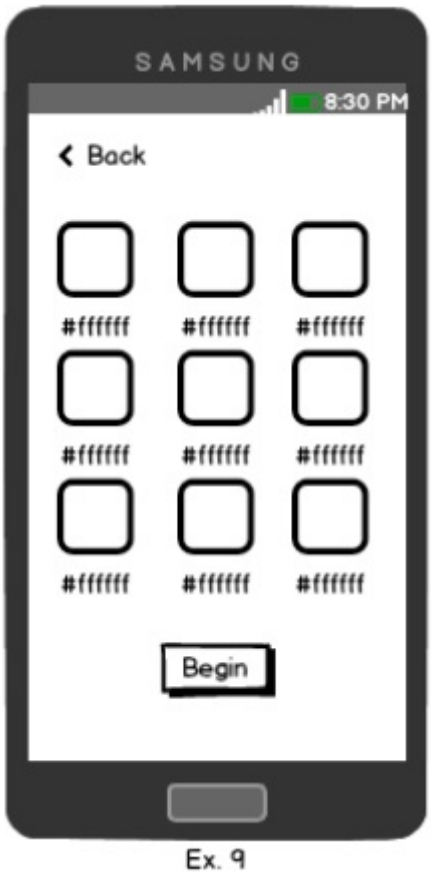

Figure 4.2: Example 9 Mockup Screen

considered finished. We are finally presented with a results screen, as shown in Figure 4.6. In said screen, the user is able to immediately transfer the pictures taken, repeat the whole process or simply confirm the end of process, which will result in the main screen being presented again.

4.1.2 Functional prototype

The prototype started with the mockups as inspiration, but with some adjustments.

From the main screen, as seen in Figure 4.7, the major differences to the original idea are the "Calibration" button and the "Presets" button. The screen calibration is described in the next section, but the "Presets" button is an extension of the application.

Instead of choosing every color, one by one, like in Figure 4.2, this button leads us to the preset activity screen (Figure 4.8). Here the user can either select, edit (Figure 4.9) or delete a previously created preset, as well as create a new one, by clicking the "Create preset" button (Figure 4.10). The main advantage of this system, compared to the original mockup, is that one can paste the color's hexcodes, instead of defining them one at a time. For measurements containing hundreds of different colors, this method proves to be much more efficient. For file reading purposes, it is

Figure 4.3: Example color 1 Mockup Screen

requested that the user writes down the colors in the format described on the top of the screen. The format should be one hexcode per line (Appendix A).

When the user wants to use a preset, a new activity begins: the brightness is brought up to 100% and the front camera becomes active and visible in the screen. This step was included to be able to aim the camera to the desired object. By changing to the first color, the camera preview is replaced with said color and the process begins.

To avoid moving the smartphone as much as possible, the original idea revolved around an automatic timer, that would change to the next colors and take a picture in between. This idea was discarded for two main reasons:

- Efficiency, since the user may or may not want to go through the whole preset list. In case there are only a few colors that the user might want to repeat, it is possible to skip unwanted ones, saving time and phone memory. Obviously, it would be fairly easy to create a new preset, but slower in the long run.

- Freedom of usage, since the user may want to adjust settings in between pictures, such as room lighting and smartphone positioning, for instance.

Instead, the picture taking and color changing process is done using the up and down volume buttons, respectively. With the click of the "Up" button, the screen changes to the first color defined, identifying it by briefly showing the hexcode associated with it, as demonstrated in Figure 4.11. The user is then able to take a picture by clicking the "Down" button. A big advantage of this method is that even tho the picture cannot yet be previewed, there is the opportunity of taking as many pictures as one might want. When satisfied, simply click the "Up" button again to change to the next picture and so on and so forth, until there are no more colors to change to, which leads us back to the main screen.

4.2 Device's Screen Calibration

It is necessary to have some sort of screen calibration, to be able to analyse the exact colors that are being used.

The reason why this is needed is because we are utilising cheap hardware, which is not trustworthy. As observable in the results of this calibration, the measurements have small errors. As we will demonstrate, just because we define a specific color to be used in the screen, which might be perceived as perfect to the human eye, it doesn't necessarily mean that it is.

This section is divided into three subsections, where the first one focuses on the connection and activity among the smarthphone, computer and spectroradiometer. The second subsection

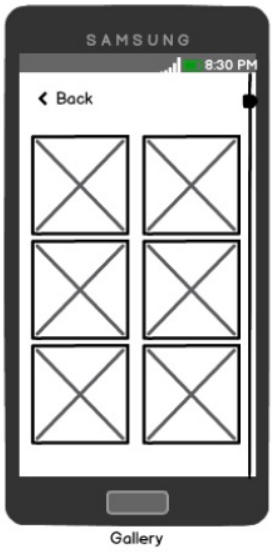

Figure 4.4: Gallery Mockup Screen

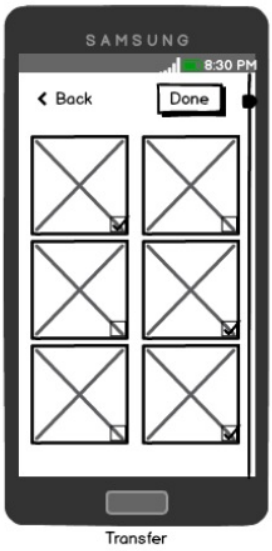

Figure 4.5: Transfer Mockup Screen

will focus on the measurements taken and their visualisation with the jfreechart library. The third subsection will be a brief analysis and conclusion based on the graphical results.

4.2.1 Connection

First, it is important to know how the connection was implemented. We have three different devices that somehow need to communicate with each other. The whole interaction can be seen in Figure 4.12. Each device has different programs running in them. Let us describe them one by one.

4.2.1.1 Computer (Host)

There needs to be a host throughout the whole communication. A JAVA program was developed to manage this communication. The software communicates directly with the spectroradiometer and with the smartphone, since these two never communicate with each other. Both of the devices are connected via USB port to the host.

There is some other software that enables the connection which is needed for this process, such as Android Debug Bridge (ADB) and JAVA SE Development Kit (JDK). There should also be a file previously created with the colors that need to be measured.

The host[1] creates a socket to be able to communicate with the smartphone device, by sending and receiving messages. There is a "Connect" button in the smartphone calibration screen, which after pressed listens for clients. After connecting through the ADB, the host opens a new Matlab session, to begin the measurements. The full measurement process works as such:

[1] We use the term "host" since, in the end, the computer is in charge of the commands. In reality, the smartphone is the host, since it is there that the connection is initialized.

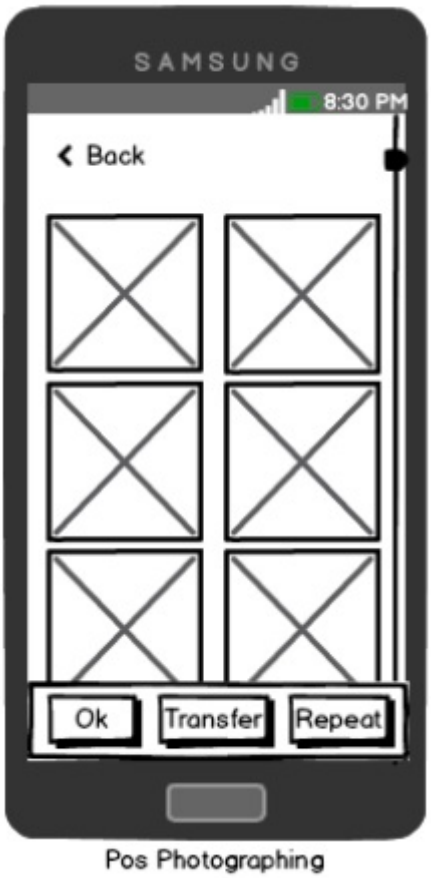

Figure 4.6: Results Mockup Screen

1. The smartphone initializes the connection and waits for the JAVA code to begin.

2. The host completes the connection, opening a new Matlab session and loading the code to control the spectroradiometer.

3. While the host has lines with color hexcodes, it sends that same line to the smartphone.

4. The smartphone changes the screen background to the colors recieved and informs the host about it.

5. Utilizing MatlabControl library, the host sends the commands necessary to take a measurement.

6. The Matlab code replies with a "Measurement taken" type of message and we go back to number 2, until the end of the colors file has no more colors to read from.

7. When there are no more colors to be read from the colors file, both the Matlab session and the socket with the smartphone device are closed.

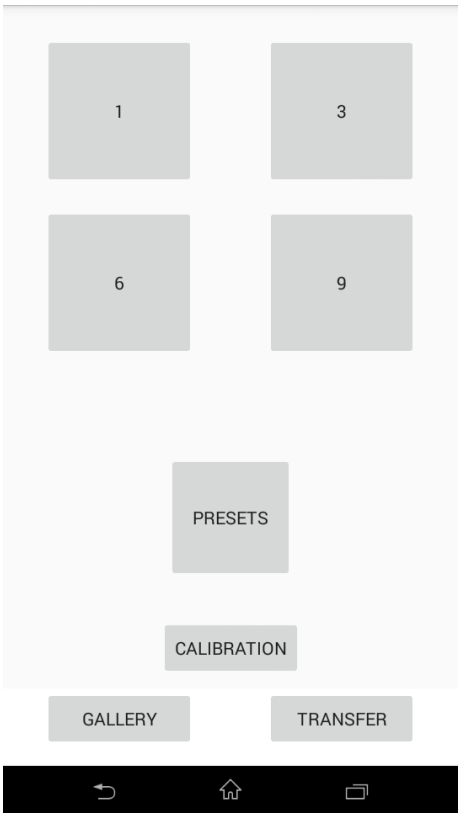

Figure 4.7: Prototype Main Screen

The full JAVA file (ScreenCalibration.java) for the communication can be found in the appendix.

4.2.1.2 Spectroradiometer

The spectroradiometer available already had Matlab code provided for its utilisation. It allows measurements to be taken, on command, providing a long list of data that can later be transformed into graphs, for better understanding and visualisation.

For the calibration itself, it is important that the device only measures when told so, meaning when the smartphone is displaying the desired color. This interaction is described in the last subsubsection.

The MatlabControl library allows a JAVA file to send commands to a Matlab program. For this specific program, the command lines used are:

- cd('Y:\\Spectroradiometer') - which opens the directory where the program is;
- addPathWithSubpathes - which readys the correct program for usage;
- comPort = 'COM3'; - which has to be set manually, according to the port that the device is connected to;

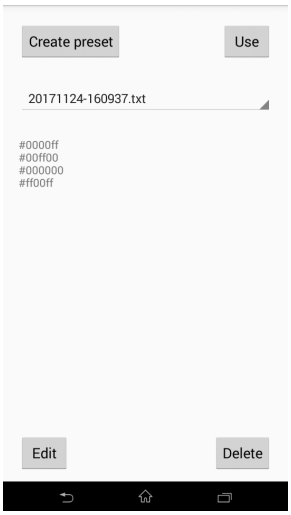

Figure 4.8: Prototype Preset Activity Screen

Figure 4.9: Edit button on Preset Activity Screen

- CS2000_initConnection(comPort); - which initializes the connection;
- CS2000_terminateConnection(); - which terminates the connection.

Specific to measurements, since the host needs to wait for the measurement to be taken, the function returningEval is used. Each time a measurement is requested it is done as in Listing 4.1, where "proxy" is the name of the Matlab session and "res" contains a 401×1 double, which is then saved into a new text file, named after that specific measurement.

4.2.1.3 Smartphone

The smartphone device part just needed some minor adjustments to work its role in the calibration process.

It initialises the connection on button click and listens for 60 seconds to see if a client connects. If no one connects, it launches a timeout. If the connection is successful, it sends a "READY" message to the client and waits for the colors to be send, so it can change the background. After it receives an "END" message, which indicates there are no more colors to change to, it replies with "ENDED", returning to the main screen.

```
1 proxy.eval("[message1, message2, cs2000Measurement, colourimetricNames] =
      CS2000_measure();");
2 res = proxy.returningEval("cs2000Measurement.spectralData", 1);
```

Listing 4.1: Commands to take and save 1 measurement

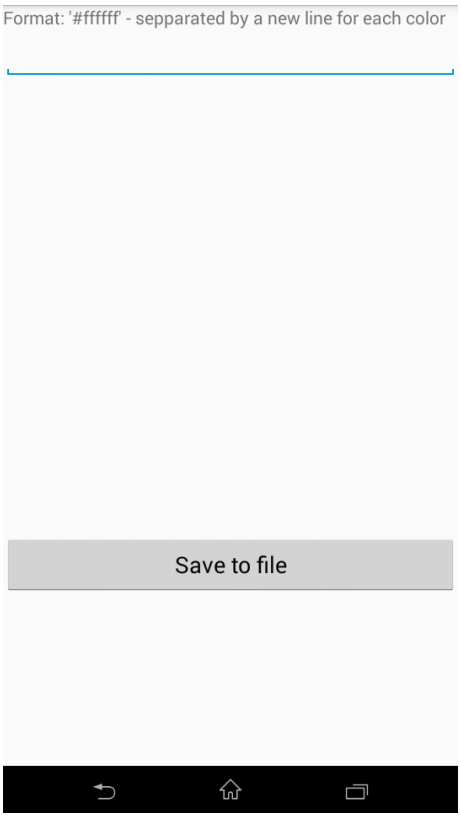

Figure 4.10: Prototype Create Preset Activity Screen

4.2.2 Measurements Visualisation

After executing ScreenCalibration.java and measuring the colors from our selected text file, a folder with the same name is created in the same directory, which contains a text file for each color measured.

Considering that we have measured, for instance, *#ff0000*, *#00ff00* and *#0000ff*, three text files with the same name as the color were created, with 401 values in each, which covers the visible spectrum, from 380 to 780 nm. Having such files allows us to create a graph for better visualisation of what the spectrophotometer has measured, to be able to compare to the ideal wave that that specific color should have. The results will be analysed in the next subsection.

GraphMaker.java works as an extension of the ScreenCalibration.java, since it reads from the folder containing the measurements. With the help of JFreeChart library, we open each measurement text file and create a JPEG graph of each color (following the example mentioned above, three graphs would be created). The library offers many different choices for graphs, but the one that seemed most adequate was a simple line graph.

After executing the program, we are left with a new folder named JPEG with the resulting pictures.

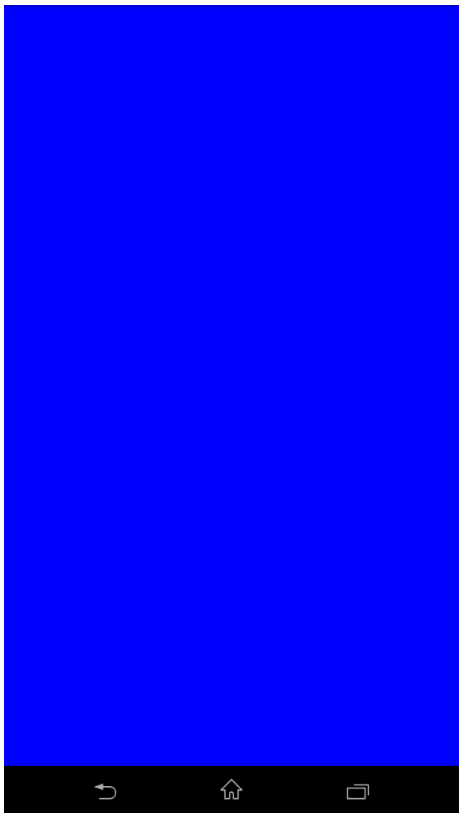

Figure 4.11: Example of a preset being used

4.2.3 Results

The results were fairly positive. For instance, in Figures 4.13, 4.14 and 4.15 we can see represented the measured values of what was meant to be a pure red colour (#ff0000), pure yellow colour (#ffff00) and a darker red colour (#790000), respectively. As we can also see, comparing to the wave length colours that we should have (Figure 4.16), there is a blue influence in the graphics created. Considering that the measurements were taken in a black room, with minimum light interference, we can only assume that the screen itself always illuminates with some shade of blue, since other colors measured that belonged to the non-blue wavelength were also influenced.

4.3 Spectral Estimation Methods

It is based on the measured data from the previous section, that the spectral object reflectance will be calculated. There are different methods that can be applied to estimate a spectral object reflectance. The three considered methods were the Wiener estimation method, which provides accurate estimates [HK07] from low-dimensional data into high-dimensional data, a linear estimation method and an Eigenvectors method.

We need to consider that there are three main parts to the whole scenario (Figure 1.1): a light source, an object and a camera.

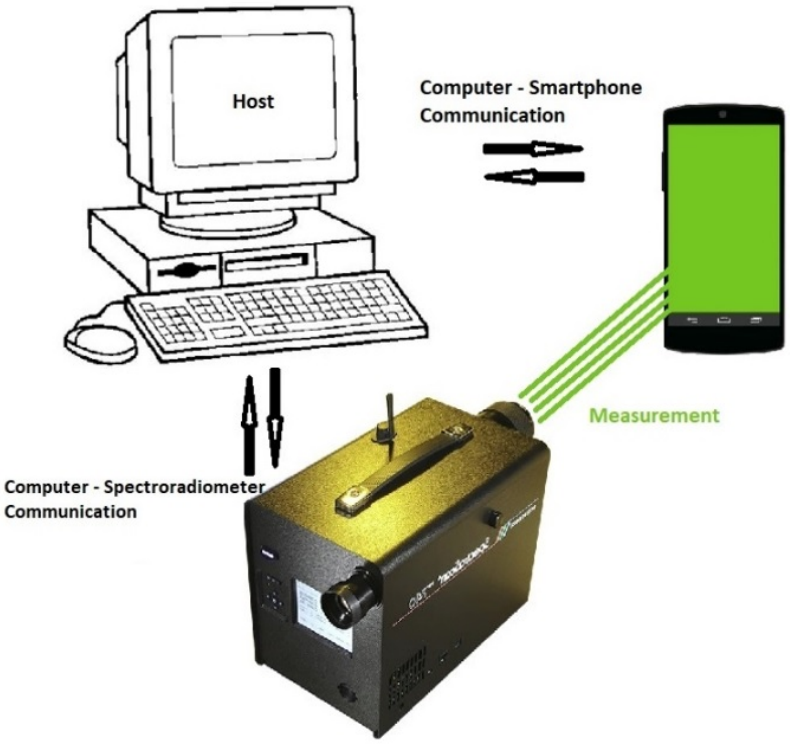

Figure 4.12: Connection setup for smartphone screen calibration

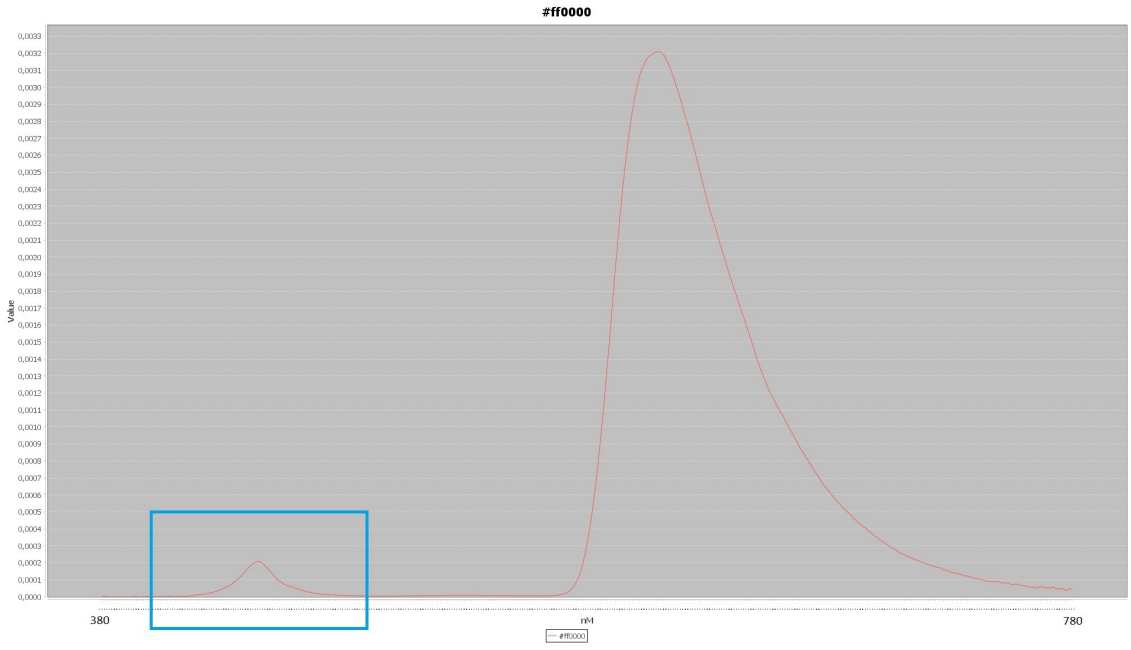

Figure 4.13: Measurements of #ff0000 with minor mistake

The above methods need some prior data, such as the spectral sensitivity of the camera and the spectral power distribution of the illumination. Being **L** the illuminant (smartphone screen), which

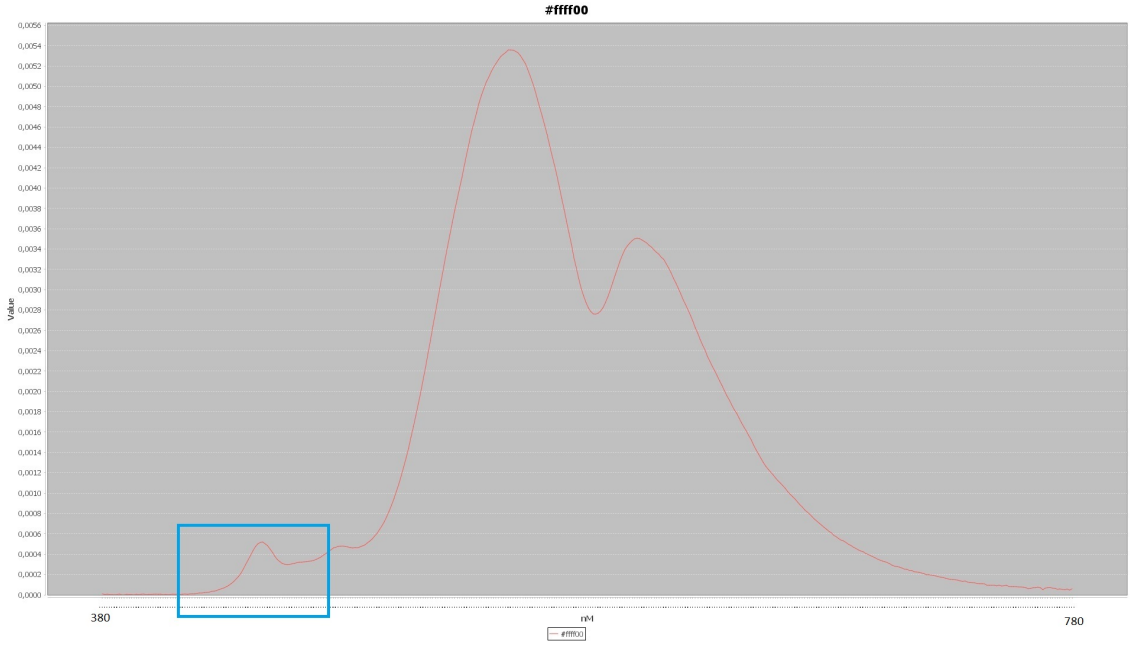

Figure 4.14: Measurements of #ffff00 with minor mistake

was already measured, **r** the spectral reflectance of the object O being imaged, and **s** the spectral sensitivity of the camera, then the response **y** of the channel **k**, can be represented as Equation 2.8.

For this case, we are sampling an RGB camera, which only has three camera channels (Red, Green and Blue), meaning that it was considered that **k** = 3. e_k is the camera system noise.

4.4 Object reflectance reconstruction

To be able to reconstruct the spectral reflectance, SpectralReflectanceReconstruction.java was developed, working with MatlabControl. By executing this program, we can add different illuminant values to work with the provided camera sensitivity and the objects' spectral reflectance. Having the Munsell color data set with the spectral reflectances of multiple objects (Figure 4.17) allows this study to have multiple examples, to compare the original values with the reconstructed ones, in Matlab. We can do so, using the spectral estimation methods described above.

The script starts off with a request for a color file, which is a required input for the execution of the script. Having found the file, it shows the colors presented in the file and opens an option menu: to add a new color measurement or to calculate with a spectral reflectance. Choosing the first one adds the measurements to the Matlab session and saves them there, for later use, if necessary.

Calculating with a spectral reflectance utilises the colors given as the illuminant, for the calculations. When selecting that option, the user can choose a specific spectral object reflectance from the data set given or all of them. Choosing either case allows a new option to be selected, which defines the type of reconstruction to be used. In order to try to replicate more realistic conditions the noise value e_k in Equation 2.8 can be disregarded completely or added with three different

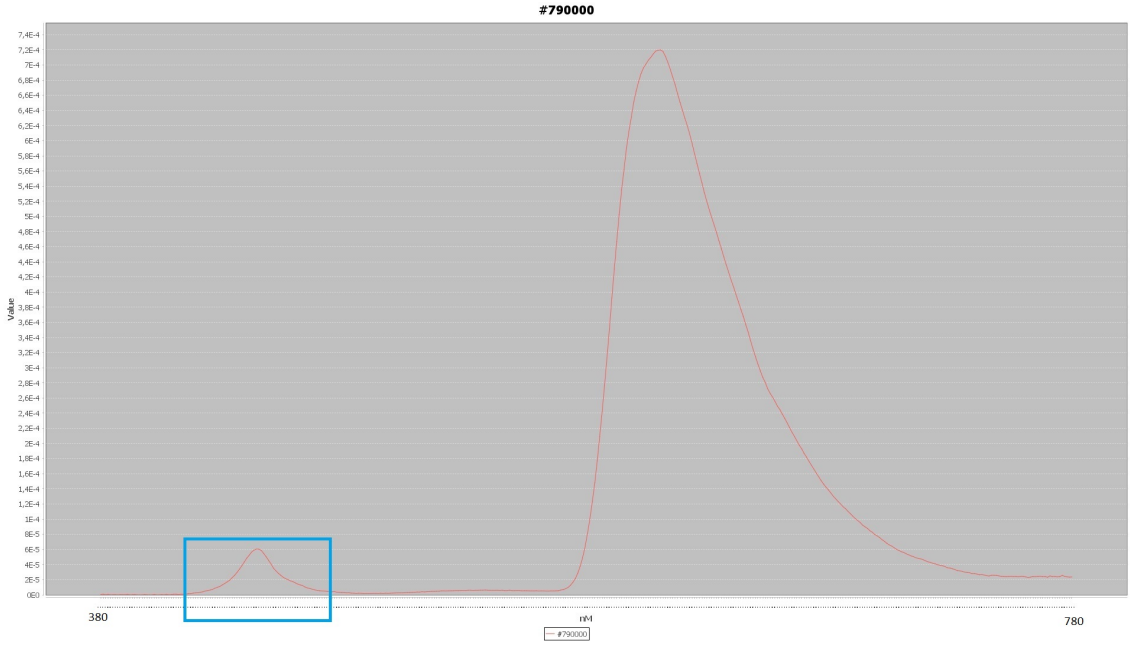

Figure 4.15: Measurements of #790000 with minor mistake

types of smoothness, with the *vng* Matlab function. With this function, we can generate White Gaussian noise, with a smoothness of 0.1, 0.01 and 0.001, which represent its' power in decibels relative to a watt. With a single reconstruction, we obtain a graph, created by Matlab, to be able to compare the differences between the original wave and the estimated one. This option could be considered to be the most irrelevant of them all, only having usability when it proves necessary to analyse a certain reconstruction in detail. More interesting is to have several reconstructions in the same graph, to be able to see how different their estimations are, compared to the original curve. In Figure 4.18, Figure 4.19 and Figure 4.20 we can observe the results with object 800 for the Wiener estimation, linear estimation and principal eigenvector estimation, respectively, disregarding noise e_k.

For this specific spectral object reflectance , we can conclude that the algorithm used to implement the linear estimation has the least amount of error compared to the original curve.

In order to understand how the reconstruction works, there are some values that need to be obtained, using equation 2.8.

Having the illuminant, the sensitivity of an RGB camera and a spectral object reflectance, we can calculate a response for each channel. Since there are three channels being considered and 21 colors in total in the example used (Appendix B), for each known spectral object reflectance we will obtain 21x3 as the measurement data of the camera. Having these values, we can use the estimation methods provided, making an estimation for each.

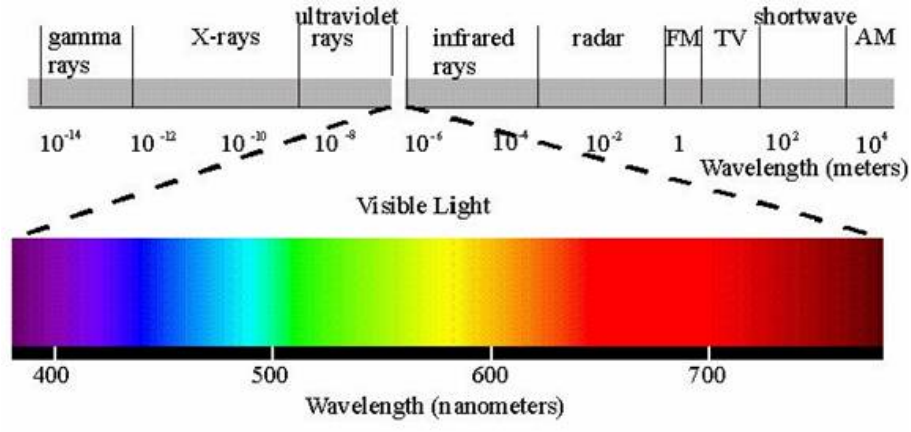

Figure 4.16: Visible spectrum with correspondent wavelength values

4.5 Experimental Results

The colors measured from the smartphone for the calibration were equidistant throughout the visible spectrum, defined every 20 nm, as shown in File20colors appendix. The reason why there's a repetition of the *ff0000* color is due to the color separation margin being so small. If the colors were measured every nm, it is very likely that more colours would be repeated, due to the conversion between the visible light spectrum and its correspondent hexcode.

Having the measurements for the example file, we end up with a 401×21 matrix, representing the spectral power distribution. Utilising this, along with a standard RGB camera sensitivity *sens* (401×3) and a sample object's O spectral reflectance (401x1), then the response of each *Red*, *Green* and *Blue* channel can be calculated (equation 2.4), creating a 21×3 matrix.

With the calculated tristimulus values, it is possible to reconstruct the spectral reflectance of object O, utilising the estimation methods, which results in three 401x1 matrixes that represent the Wiener, Eigenvectors and Linear estimations. We can then compare these with the original curve via rms (equation 2.22), that gives us an error value for each, for the spectral reflectance of object O.

To calculate the colorimetric error for the spectral reflectance of object O, a pair of new tristumus values are calculated (equation 2.4), one of them utilising the original wave and the other utilising the reconstructed one and both of them using a standard human sensitivity (as the *"camera"*) and CIE illuminant D65 (as the illuminant). Converting the RGB triplets to XYZ values and applying ΔE_{00}*, we obtained the colorimetric error for each estimation method.

To be able to have a robust set of results, the colorimetric and rms error were calculated for each spectral reflectance from the Munsell color data set (1600 different spectral object reflectances). Table 4.1 compares the spectral and the colorimetric errors of the different reconstruction methods used, disregarding the noise factor. For the colorimetric error, a value was considered ideal, for a trustworthy reconstruction that cannot be detected by the human eye, as any value between 0 and 1.0. Anything between 1.0 and 6.0 would still be acceptable. Given that

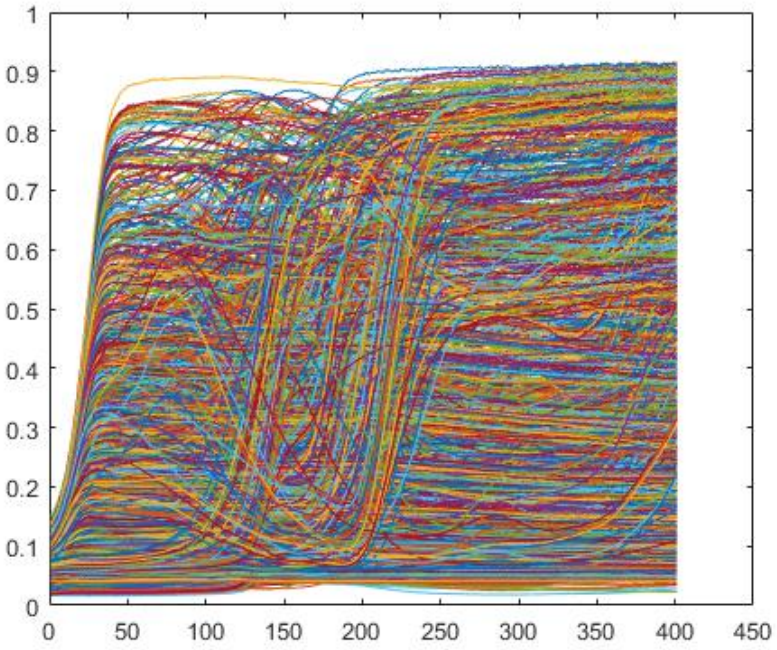

Figure 4.17: *Munsell* color data set with $0<\lambda<400$

with the Linear estimation the mean value and the standard deviation for the colorimetric error is 0.0442 and 0.0397 shows great potential for further development. The noise was introduced with three different smoothless levels (0.1, 0.01 and 0.001). Since it consists of a random value every time, 10 different noises were applied for each smoothness level, as a first approach. As seen in Appendix C, we can observe that the mean and standard deviation of the estimations increment immensely, proving to be useless, apart from the Wiener estimation, that still has an acceptable error. To ensure the results' robustness, the same process was applied with 100 different noises instead, as seen in Appendix D. The results proved to follow the same pattern.

4.6 Conclusions

The results show that the absence of noise for the spectral object reflectance estimation provides an accurate reconstruction of the original spectral object reflectance, utilising the linear estimation. With the introduction of noise, the linear estimation proves to be useless, being the Wiener estimation the best one to consider.

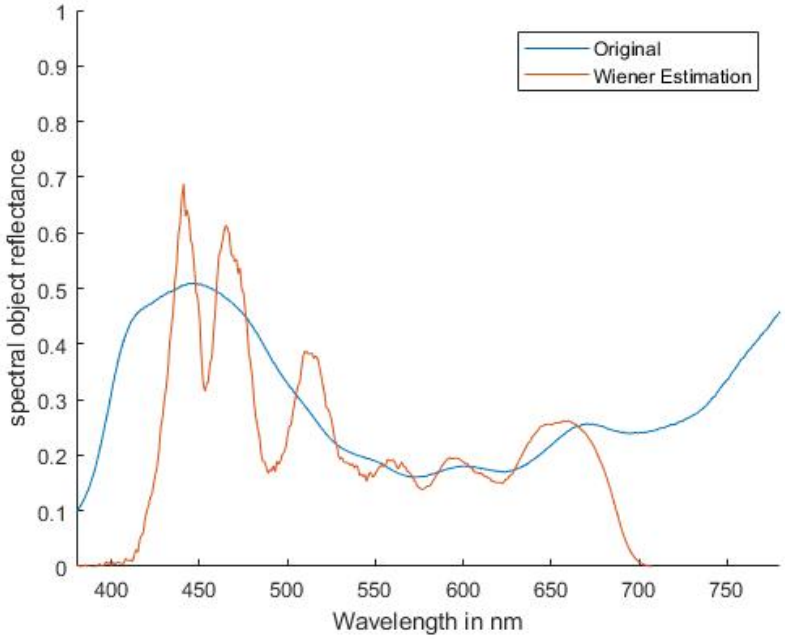

Figure 4.18: Reconstruction of the spectral object (800) reflectance using Wiener estimation

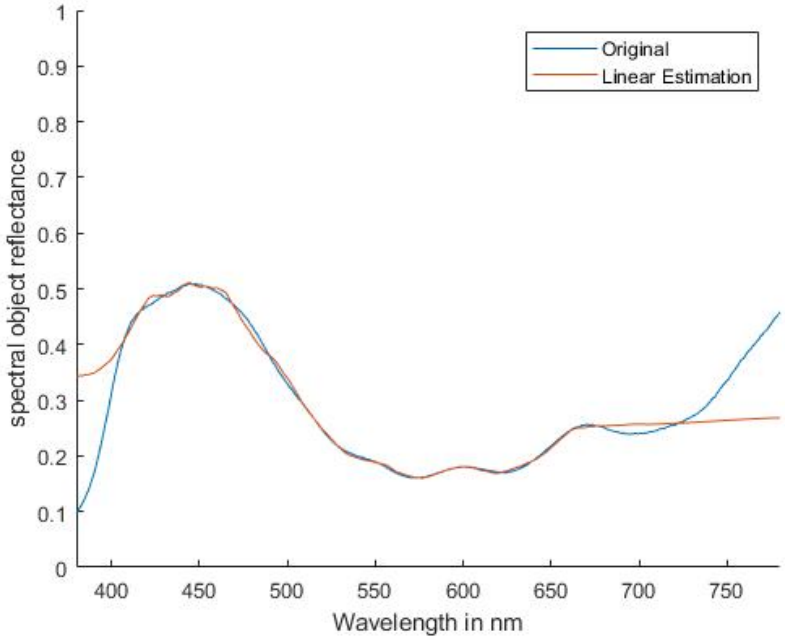

Figure 4.19: Reconstruction of the spectral object (800) reflectance using Linear estimation

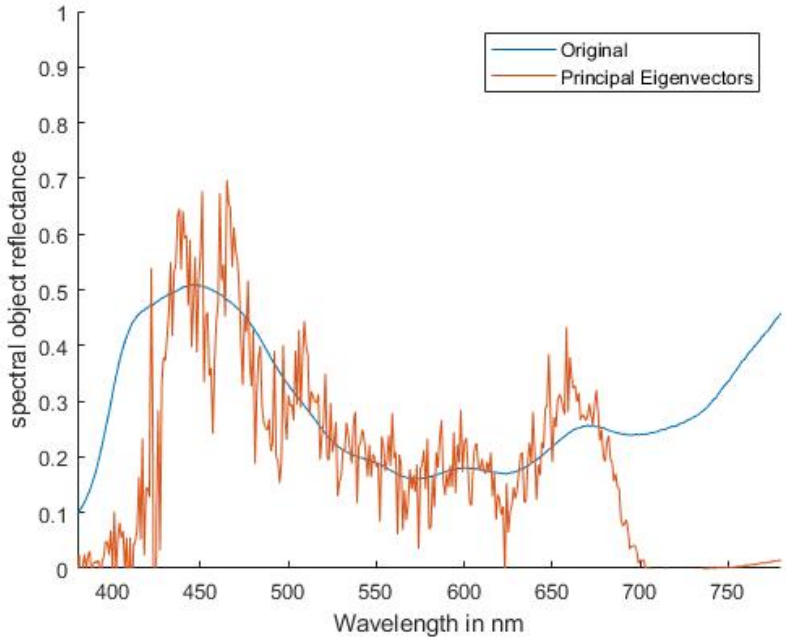

Figure 4.20: Reconstruction of the spectral object (800) reflectance using Principal Eigenvectors estimation

Table 4.1: Comparison of the color accuracy of the different estimations, using the *Munsell* color data set

Method	spectral rms			ΔE_{00}* under D65		
	mean	std.	max	mean	std.	max
Wiener Estimation	0.1978	0.1244	0.4783	4.2618	2.3104	18.8325
P. Eigenvectors	0.2034	0.1284	0.5210	5.6382	3.6179	26.3720
Linear Estimation	0.0247	0.0219	0.1520	0.0442	0.0397	0.6118

Chapter 5

Conclusions and Future work

5.1 Conclusions

An Android application was developed to be able to take pictures with the front camera of a smartphone while the screen serves as the illuminant, which goes through different wavelengths of the light spectrum.

A calibration process was developed in order to accurately measure the spectral distribution of each color displayed by the screen, with the help of a spectroradiometer.

With the measurements taken, utilising a training data set of different spectral object reflectances (Munsell Color Palette) and a standard RGB camera sensitivity, we were able to apply three different spectral object reflectance estimation methods: Wiener estimation, linear estimation and Principal eigenvectors.

To have more accurate results, the spectral (root mean square) and colorimetric (CIEDE2000) errors were applied to the three estimation methods over the *Munsell* color data set. The max, mean and standard deviation were applied over these results.

A random noise component with three different smootheness levels was added to the response tristimulus equation in order to try to replicate more realistic conditions.

The results proved to be excellent using the linear estimation, although the addition of noise changed this predictment. The Wiener estimation provides a better reconstruction, in this case.

5.2 Future work

Since the results proved to be good, it would be very interesting to add the other half of the components to the simulation: the front camera of the smartphone. Doing so would require another calibration process, to measure the camera channels sensitivity. After having this measurement taken, we can add it to the response tristimulus equation. Having good results, we can pass from a simulative environment to a real conditions one, utilising real objects.

Going through the whole process with higher or lower quality smartphones would also be interesting to look into, to see if the hardware difference gives justifiable different results.

References

[14201] CIE Publication 142. Improvement to industrial colour-difference evaluation. Vienna, Austria, Commission Internationale de L'Éclairage, 2001.

[AVMA96] Abrardo A., Cappellini V., Cappellini M., and Mecocci A. Art-works colour calibration by using the vasari scanner. pages 94–96, 1996.

[Boo00] G. Booth. *Nitro Compounds, Aromatic. Ullmann's Encyclopedia of Industrial Chemistry*. Published online, 2000.

[C.15] Katrina C. Applications of spectrophotometry in agriculture: Quantitative analysis of fertilizer properties. Available at https://www.hunterlab.com/blog/color-chemical-industry/applications-of-spectrophotometry-in-agriculture-quantitative-analysis May 2015.

[CD04] Hardeberg J.Y. Connah D. Comparison of linear spectral reconstruction methods for multispectral imaging. International Conference of Image Processing, 2004.

[G.82] Wyszecki G. *Color Science: concepts and methods, quantitative data and formulae*. John Wiley & Sons, 1982.

[Gil00] David Gilbert. Jfreechart. Available at http://www.jfree.org/jfreechart/, February 2000.

[Goo13] Google. Android studio. Available at https://developer.android.com/index.html, May 2013.

[Gos15] Bill; Steele Guy; Bracha Gilad; Buckley Alex Gosling, James; Joy. The java language specification. Available at https://docs.oracle.com/javase/specs/jls/se8/jls8.pdf, February 2015.

[HHH+] Hideaki Haneishi, Takayuki Hasegawa, Asako Hosoi, Yasuaki Yokoyama, Norimichi Tsumura, and Yoichi Miyake. System design for accurately estimating the spectral reflectance of art paintings2, month =. *Appl. Opt.*, pages 6621–6632.

[HK07] P. StigellK. MiyataM. Hauta-Kasari. Wiener estimation method in estimating of spectral reflectance from rgb images. 2007.

[I.72] Newton I. Philosophical transactions of the royal society, 1672.

[IB99] Francisco H. Imai and Roy S. Berns. Spectral estimation using trichromatic digital cameras. 1999.

REFERENCES

[K.07] Koskinen I. K. *Mobile multimedia in action*. New Brunswick, N.J. : Transaction Publishers, 2007.

[KOO+99] Yasuhiro Komiya, Kenro Ohsawa, Yuri Ohya, Takashi Obi, Masahiro Yamaguchi, and Nagaaki Ohyama. Natural color reproduction system for telemedicine and its application to digital camera. 3:50–54, January 1999.

[Ltd14] Bentham Instruments Ltd. A guide to spectroradiometry instruments & applications for the ultraviolet. Berkshire, United Kingdom, available at https://www.bentham.co.uk, January 2014.

[Mat84] MathWorks. Matlab. Available at https://uk.mathworks.com/products/matlab.html, 1984.

[Mat10] Jeff Mather. Spectral and xyz color functions. Available at https://uk.mathworks.com/matlabcentral/fileexchange/7021-spectral-and-xyz-color-functions, Updated November 2010.

[Meh11] Akul Mehta. Jfreechart. Available at https://pharmaxchange.info/2011/08/introduction-to-the-electromagnetic-spectrum-and-spectroscopy/, August 2011.

[MM99] Y. Miyake and K. Miyata. Color image processing based on spectral information and its application, 1999.

[MVEO94] M. Mahy, L. Van Eycken, and A. Oosterlinck. Evaluation of uniform color spaces developed after the adoption of cielab and cieluv. *Color Research & Application*, 19(2):105–121, 1994.

[OR06] N. Ohta and A. R. Robertson. *Colorimetry: Fundamentals and Applications*. Published in Association with the Society for Imaging Science and Technology, 2006.

[R.97] Kang H. R. *Color technology for electronic imaging devices*. SPIE Optical Engineering Press, Bellingham, Washington, USA, 1997.

[RWGH11] M. R. Pointer R. W. G. Hunt. *Measuring Colour*. Fourth edition edition, October 2011.

[S.00] Berns R. S. *Principle of Color Technology*. Third edition edition, 2000.

[Sci15] Gamma Scientific. Gamma scientific spectroradiometers. California, United States of America, available at http://www.gamma-sci.com, May 2015.

[SD05] Wu W. Sharma, G. and E. N. Dalal. The ciede2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. 2005.

[SWDA05] Gaurav Sharma, Wencheng Wu, Edul N. Dalal, and Mecocci A. The ciede2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. pages 21–30, 2005.

[SX04] Hui-Liang Shen and John H. Xin. Spectral characterization of a color scanner by adaptive estimation. *J. Opt. Soc. Am. A*, (7):1125–1130, July 2004.

[Wag07] Jerker Wagberg. optprop - a color properties toolbox. Available at https://uk.mathworks.com/matlabcentral/fileexchange/13788-optprop-a-color-properties-toolbox, Updated March 2007.

[Y.99] Hardeberg J. Y. *Acquisition and reproduction of colour images: colorimetric and multispectral approaches*. PhD thesis, Ecole Nationale Superieure des Telecommunications, Departament TSI, Paris, France, 1999.

[ZW08] Dai D.Q. Zhang W.F. Spectral reflectance estimation from camera responses by support vector regression and a composite model, September 2008.

REFERENCES

Appendix A

Presets

A.1 Preset user input

This is an example of how the format should be while creating a preset file:
#00ff92
#36ff00
#81ff00
#c3ff00
#ffff00
#8300b5
#6a00ff
#0000ff
#007bff

Appendix B

Files

B.1 File 21colors.txt

This is an example of how a color file to be measured looks like. The colors used for the main experiment were defined as presented bellow, followed by the standard "*END*" in every color file:

#610061
#8300b5
#6a00ff
#0000ff
#007bff
#00d5ff
#00ff92
#36ff00
#81ff00
#c3ff00
#ffff00
#ffbe00
#ff7700
#ff2100
#ff0000
#ff0000
#ff0000
#D20000
#A50000
#790000
#4C0000
END

Appendix C

Result tables with 10 random noises

C.1 Color accuracy of estimations with 10 random 0.1 noises

Table C.1: Comparison of the color accuracy of the different estimations, using the *Munsell* color data set, with 10 random 0.1 noises

Method	spectral rms			ΔE_{00}* under D65		
	mean	std.	max	mean	std.	max
Wiener Estimation	0.3488	0.0238	0.7020	12.8898	2.7444	87.4252
P. Eigenvectors	26.3329	6.5364	49.0512	103.7107	16.6108	763.5683
Linear Estimation	21.8646	1.9797	50.4556	56.7662	13.8932	470.5374

C.2 Color accuracy of estimations with 10 random 0.01 noises

Table C.2: Comparison of the color accuracy of the different estimations, using the *Munsell* color data set, with 10 random 0.01 noises

Method	spectral rms			ΔE_{00}* under D65		
	mean	std.	max	mean	std.	max
Wiener Estimation	0.2860	0.0127	0.5662	11.7813	4.9470	96.6953
P. Eigenvectors	21.1729	7.4602	47.7494	98.4413	11.4997	192.0052
Linear Estimation	10.5043	0.0017	17.0297	53.5189	15.2589	507.5044

C.3 Color accuracy of estimations with 10 random 0.001 noises

Table C.3: Comparison of the color accuracy of the different estimations, using the *Munsell* color data set, with 10 random 0.001 noises

Method	spectral rms			ΔE_{00}* under D65		
	mean	std.	max	mean	std.	max
Wiener Estimation	0.2910	0.0119	0.5710	15.9011	4.7114	91.7429
P. Eigenvectors	18.4494	6.1948	23.4281	99.9923	12.5909	186.7350
Linear Estimation	13.5200	0.0018	23.7580	56.2202	18.3263	457.4602

Appendix D

Result tables with 100 random noises

D.1 Color accuracy of estimations with 100 random 0.1 noises

Table D.1: Comparison of the color accuracy of the different estimations, using the *Munsell* color data set, with 100 random 0.1 noises

Method	spectral rms			ΔE_{00}* under D65		
	mean	std.	max	mean	std.	max
Wiener Estimation	0.2970	0.0182	0.7044	14.7103	4.8312	110.0032
P. Eigenvectors	20.3960	0.0015	50.9206	100.9960	15.2854	763.5683
Linear Estimation	12.8469	0.0015	50.4556	59.0401	18.5210	697.1504

D.2 Color accuracy of estimations with 100 random 0.01 noises

Table D.2: Comparison of the color accuracy of the different estimations, using the *Munsell* color data set, with 100 random 0.01 noises

Method	spectral rms			ΔE_{00}* under D65		
	mean	std.	max	mean	std.	max
Wiener Estimation	0.2805	0.0164	0.7952	12.5631	4.2876	104.7896
P. Eigenvectors	17.9545	0.0011	42.8496	100.9843	14.0006	727.6076
Linear Estimation	10.6668	0.0022	33.7916	47.3502	15.7018	566.4717

D.3 Color accuracy of estimations with 100 random 0.001 noises

Table D.3: Comparison of the color accuracy of the different estimations, using the *Munsell* color data set, with 100 random 0.001 noises

Method	spectral rms			ΔE_{00}* under D65		
	mean	std.	max	mean	std.	max
Wiener Estimation	0.2953	0.0181	0.7003	14.5896	4.8292	109.2473
P. Eigenvectors	20.1649	0.0015	50.3424	100.9012	15.5293	876.6078
Linear Estimation	12.7013	0.0015	49.8855	58.6717	18.4542	683.1399