Master Erasmus Mundus
Color in Informatics and Media Technology (CIMET)

Color calibration and spectral reflectance estimation using optimum reconfigurable Transverse Field Detectors with full spatial resolution

Master Thesis Report

Miguel Ángel Martínez-Domingo
m.martinez.domingo@gmx.com

Supervisors:

Dr. Eva M. VALERO
University of Granada,
Granada, Spain

Dr. Giacomo LANGFELDER
Politecnico di Milano,
Milan, Italy

Dr. Ville HEIKKINEN
University of Eastern Finland,
Joensuu, Finland

Jury:

Dr. Alain TRÉMEAU
University Jean Monet, Saint Etienne, France

Dr. Damien MUSELET
University Jean Monet, Saint Etienne, France

Defended at the University of Granada, Granada, Spain

June 15, 2012

Department of Optics
University of Granada
Edificio Mecenas
C. Fuentenueva s/n
ES-18071 Granada Spain

Color calibration and spectral reflectance estimation using optimum reconfigurable Transverse Field Detectors with full spatial resolution

Miguel Ángel Martínez-Domingo

July 2012

Abstract

During this master thesis, we have studied the performance of a new kind of sensors based in a revolutionary emergent technology called Transverse Field Detectors (TFDs). The work is divided in two main tasks: color correction or color calibration, and spectral reflectance estimation, both studied for general purpose applications. We propose different approaches depending on the task to be done with the TFDs and on the type of application we aim to use these systems for.

The experiments are based in simulations which account for the noise processes present in this technology in a realistic way. The study is focused in algorithms which are recent proposals. These are Kernel-based regression methods. They have been implemented, optimized and compared with other typical models frequently used in these two tasks, such as Neural Networks and Polynomials for color correction, and Wiener and Pseudoinverse for spectral reflectance estimation. We have proved that regularized Kernel models work better than the rest.

The raw performance found for the TFD has been enhanced by proposing, implementing and optimizing improvements in the system's set-up. These enhancements consist basically in taking advantage of the tunable nature of TFDs, to get up to 24 channels information by using up to 8 shots, plus narrowing down the spectral responsivities of the sensors by including a filter, without eliminating the full spatial resolution property of this technology. We compared the performance of TFDs with that of RGB Bayer-filtered sensors with narrower responsivities.

We have also compared the direct color calibration versus the indirect color calibration through spectral reflectance estimation under the same conditions and found that indirect calibration works better.

We have reduced the average error down to 70% of the initial performance, reaching average ΔE^*_{00} error values below 1 unit, and outperforming the RGB sensors with the additional advantage of full spatial resolution. This work is the proof that TFD sensors have potential enough to become the future of the spectral imaging.

Preface

This work would not have been possible without the patience and the spread of the most pure polychromatic light over the darkness, of my main supervisor Eva Valero, the help from far lands of my co-supervisors Giacomo Langfelder and Ville Heikkinen, and the daily support received from my colleagues and family. Thank you all for being a necessary part of this experience.

"When you open your mind to the impossible, sometimes you find the truth". Walter Bishop, Fringe Division, Federal Bureau of Investigation (USA).

Miguel Ángel Martínez-Domingo, 15 of July 2012

Contents

List of Figures

List of Tables

1 Introduction

This work explores and improves the capabilities of a new kind of sensors, called Transverse Field Detectors (TFDs) [1], for being used as spectral imaging devices. We designed different experiments to study the potential of this new technology still under development, and improve its performance by implementing different methods. We investigated two different but related applications (color calibration and spectral reflectance estimation), from a different and new perspective that has never been studied before. This project faced problems that were never addressed by other authors in the same way, due to how this newborn technology works, in a different way than what is existing so far, the novel models used, etc. The results obtained pave the way for the design and optimization of accurate spectral imaging devices using TFDs, improving its raw potential found and confirming this technology as a very promising approach for spectral image capture systems. We have demonstrated how this new technology can be optimized to exploit its capabilities to span the future of the spectral imaging in a smart and revolutionary way.

Spectral science is a branch of optical sciences and engineering that is becoming increasingly important in the last years. Within it, spectral imaging (i.e. acquisition of spectral information from every pixel of an image)is one of the most active fields [2]. Its possibilities are wide in number and variety: medical imaging [3], food industry [4][5], military applications [6], remote sensing [7], and everyday new doors are opened for the study of the spectral properties of the world surrounding us. Studying the spectral properties of objects, we can describe many of their characteristics. In addition, retrieving the spectral reflectance of objects in a scene, allows us to get rid of illuminant metamerism issues and facilitates the color constancy of images captured disregarding the effect of the illuminant. Regardless of these swarm of potential applications, capturing spectral information poses many practical problems. There is unanimous agreement in the fact that the ideal solution would be to achieve full spatial and spectral resolution in one single shot [8]. Many efforts have been dedicated to this aim, but it is far from reality still. Currently, there are some operative solutions that can be divided in different approaches. We distinguish between two types. Hyperspectral capturing systems, have usually monochrome sensors and either some dispersive elements such as prisms or gratings to decompose light into its chromatic components or some narrow band filter able to separate the polychromatic light into several wavelength bands. Multispectral capturing systems, have a reduced number of channels and need to be combined with some estimation algorithms to estimate the spectral reflectances (with many wavelengths) from few sensor's responses (which means an ill-posed problem). Some representative examples of these kinds of systems are described below:

Hyperspectral systems give very accurate information for many spectral bands, but usually they are expensive and very complex devices which need to be used under controlled conditions. Thus they need more time for the capturing process. Among them we can find different

approaches like (see Figure 1): monochrome sensors with a wheel of up to 31 color filters, which need as many shots as filters and the mechanical change of filters between shots; Line spectral scanner systems like Imspector developed by Specim Ltd. (Finland), which needs the controlled spatial scanning of the scene; Liquid Crystal Tunable Filters (LCTFs) plus monochrome sensors, that need several shots and as the other approaches with multiple filters, have focusing issues for different wavelengths; or systems based on volume Bragg gratings, like the widefield hyperspectral camera S-EOS developed by Photon ETC; which is bulky size, expensive and also need several shots and a complex optical engineering to get the final corrected spectral image. One shot approaches based in a prism or grating plus a spatial mask [9], they can be used for real time applications, but their spatial resolution is very reduced so we can only take very small images with poor sharpness.

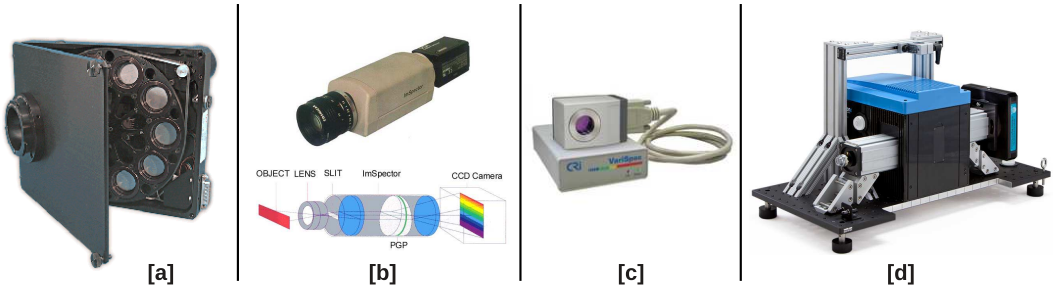

Figure 1: [a] Color filter wheel for 10 filters by LEP Ltd (USA). [b] Imspector by Specim Ltd. (Finland). [c] LCTF plus monochrome sensor by CRI Varispec. [d] Widefield hyperspectral camera S-EOS by Photon ETC (USA)

Multispectral systems are usually less accurate, but they allow faster and less complex captures, although in general, they have no full spatial resolution. We can find some approaches among others like: RGB sensors taking few shots with different broad-band color filters in front of them, which need same number of shots as filters, the mechanical change of filters between shots and do not have full spatial resolution; Cameras with three different CCD sensors (3-CCD), each one with a different color filter in front of it. These ones have full spatial resolution, but they are expensive, not robust and can have alignment issues (see Figure 2 [a]) or Hybrid-resolution systems like the one proposed in [10], which have poor spatial resolution and also may have alignment problems in the 3-CCD case (see Figure 2 [b]).

This a brief summary of the state of the art in the spectral imaging technology used so far. As we have explained, all these capture devices are valid, but they have limitations due to their intrinsic characteristics. What if we could overcome all these drawbacks with a portable, single sensor, inexpensive, robust, quick and full spatial resolution capturing system? Here is where TFDs come into scene. They are multispectral sensors, based in CMOS technology [11], that take advantage of the wavelength-dependent penetration depth of photons in silicon, like the Sony Foveon X3 technology [12]. But how do they work? To explain this in a clear way, it is easier to compare them with the well known Bayer-filtered sensors and then explain the full resolution layered sensors.

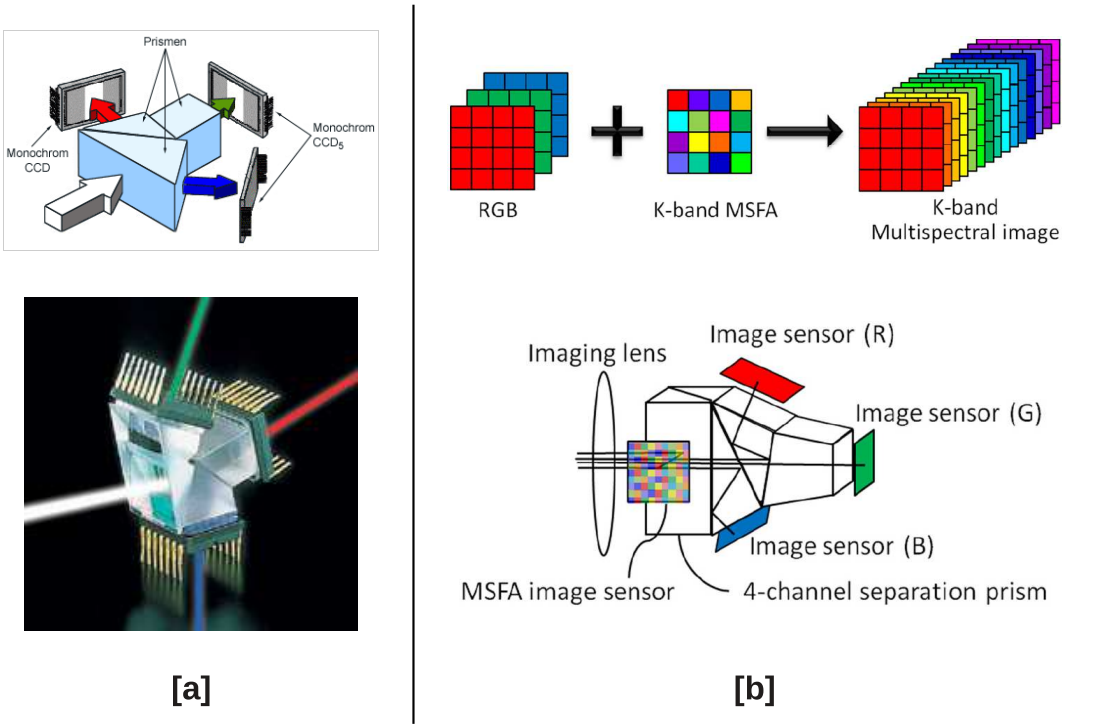

Figure 2: [a] Triple CCD color imaging system. [b] Hybrid-resolution system based in Multispectral Filter arrays

Silicon material has the physical property of absorbing photons in the visible range with a quite uniform sensitivity. A Bayer-filtered sensor, is a bi-dimensional matrix of pixels, whose sensitivities are all the same. In front of this matrix, a set of color filters is put, in such a way that each pixel has in front of it a single color filter (either red, or green or blue only). Out of the total number of pixels in the sensor, 50% of them have green filter, 25% of them red and the remaining 25% of them blue. this is done like this to mimic the human visual system's characteristics, which has higher responsivity to green and yellowish green areas of the spectrum in the photopic range. Thus we only have one channel information in each pixel. In order to get the three channels information in all the pixels, interpolation algorithms are needed. This process is called demosaicking [13][14][15]. These algorithms are complex and each different company has its own way to make these computations (many of them are kept secret). As a revolutionary new approach for capturing systems, changing totally the architecture, we have the layered sensors. Besides the wavelength range of absorption of light in silicon, there is another interesting physical property that these sensors take advantage of. The penetration depth of photons in silicon is higher, the higher the wavelength of the radiation is. This means that red component of light will be absorbed deeper inside the surface of incidence than the green component and both deeper than the blue component of light. We can see the wavelength dependency of the

absorption coefficient of silicon on the left side of Figure 3.

Figure 3: Left: Wavelength dependency of the absorption coefficient of silicon. Center: luminance image retrieved from an image taken with a Beyer-filtered sensor. Right: luminance image retrieved from an image taken with a Sony Foveon X3 sensor

The lower the absorption coefficient is, the deeper the photons penetrate the substrate. In this way, if we collect the electrons generated from the incident photons in different depths, we can separate different channels in every pixel, and we will have information for each channel in all pixels without the need of interpolation. This is called full spatial resolution, and the quality of the image is clearly better than the quality of an interpolated image taken with a Bayer-filtered sensor (see Figure 3 center and right [16]). This technology has a clear advantage over Bayer-filtered sensors (the full spatial resolution without interpolation). The Sony Foveon X3 sensors work like this, and their spectral responsivities are unique. They are always the same for all pixels. But, what if we could go one step further, having the possibility of increasing the number of channels without loosing the full spatial resolution we gained? This is what TFDs do. They are still a newborn technology. The idea is to apply a tunable transverse biasing voltage to each pixel, that changes the responsivities of the channels very fast. Depending on the value of this applied voltage, different electric field configurations are obtained. Carrier's paths are determined by the electric field lines, so different biasing values lead to different collections of electrons at different depths and therefore to different spectral photoresponses [11]. In this way, we can take one shot with three channels, and afterwards change the biasing voltage and take a new shot with three different channels (see Figure 4 extracted from [17]).

We would have now six channels in only two shots without the need of any scanning, or mechanical device moving or any complex rectification algorithm to correct misalignment or any other undesirable effects. Furthermore, TFDs have the possibility to retrieve up to five channels in only one shot, collecting also the photons of the infrared area of the spectrum. We can see an example of the the streamlines of the different contacts in one pixel for a four channels configuration in Figure 5 extracted from [18].

In this work only the three channels corresponding to the visible range will be taken into consideration. In order to have a schematic idea of how the spectral responsivities look like for a typical three channels TFD sensor, compared with those of Bayer-filtered and Foveon X3 sensors, we present in Figure 6, the spectral quantum efficiencies of: (a) Beyer-filtered sensor, which have quite narrow responsivities due to the fact that each channel can be filtered separately

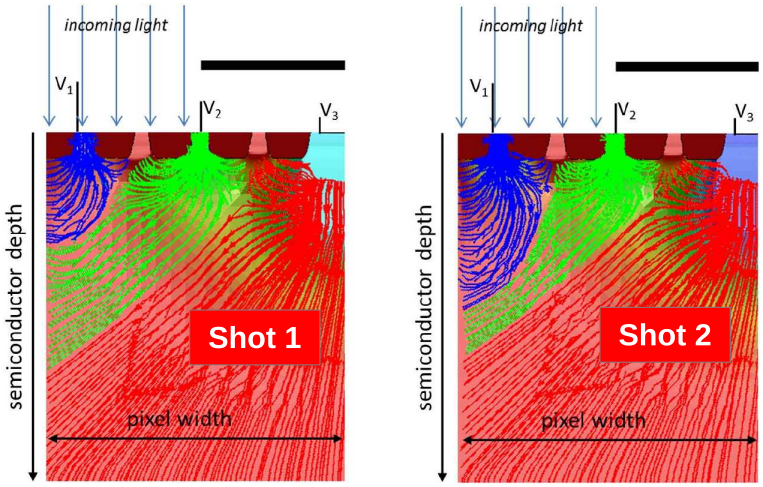

Figure 4: Collection trajectories inside the volume of the pixel of TFD for two different biasing conditions

from others by a different color filter, with the drawback of the need of demosaicking to get the full spatial resolution; (b) Foveon X3 sensor, whose quantum efficiencies are quite broad spectrally, but they have no need for interpolation, although their responsivities are unique and they can not be changed; (c) TFD sensor, which have also broad quantum efficiencies like those of Foveon X3, but in this case they are not unique and they can be tuned applying different voltages without loosing the full spatial resolution without the need of interpolation.

TFDs properly belong to the multispectral capture groups of systems explained before. Therefore, as we said, it needs the use of estimation algorithms to get the spectral reflectance information from the sensor's responses. Some previous studies have shown the feasibility of using spectral estimation algorithms along with TFD sensor responses [19] using some complex proposals among others like Matrix R [20] or Projection onto Convex Sets (POCS) [21]. We will introduce instead in this work a recent proposal: the Kernel methods. Not many authors have used them before for spectral reflectance estimation [22][23], and they have never been used before with TFDs. In order to evaluate comparatively the performance of these models, we have also implemented some widely used simple estimation algorithms such as: Wiener [24], Pseudoinverse, Neural Networks [25] or Polynomial fitting [26] (all of them explained in Chapter **??**).

Apart from the models mentioned here, some others were considered for this study. They all need dimensionality reduction. They are: Maloney-Wandell [27] which is simple and fast, but does not account for noise in the system; Imai-Berns [28] which is also simple, but needs two different training sets, and Shi-Healey [29], which accounts for noise but is rather slow for big training sets. Due to our time constraints for this work and the need of performing this optimal dimensionality reduction plus the implementation and optimization of these older models, we decided to focus only in the most recent and promising proposals and compare them with the typical and well known models.

Once the spectral reflectance estimation is done, we can convert straightforwardly the es-

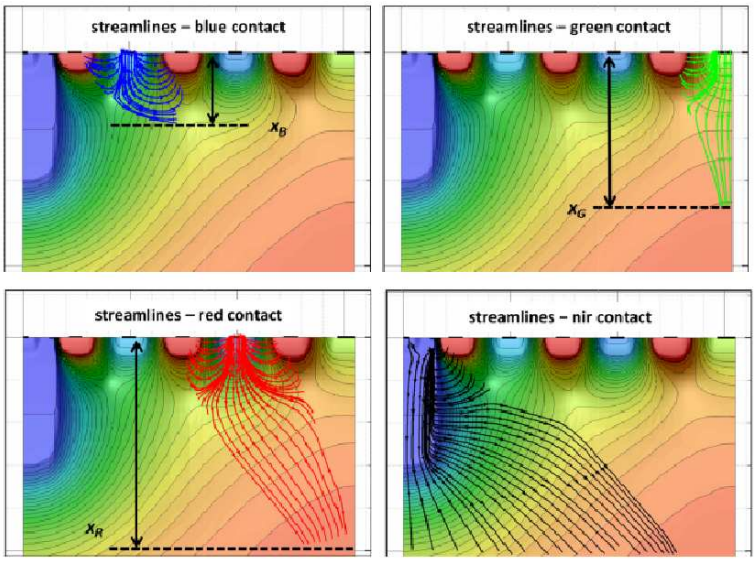

Figure 5: Stream lines for the four contacts of a four channels configuration of a TFD pixel

timated reflectance spectra into tristimulus values and color coordinates. This is called color calibration, or color correction. This process leads to convert or map from sensor's responses, which are in a device-dependent space, to standard color spaces. This is a very important task because standard color spaces describe the human perception of color. There are many studies done in this fields (see [30][31][32] as representative instances of them), but none of them for TFDs. In most previous related work, the color correction is performed directly, without the need of passing through spectral reflectance. We can use algorithms to map directly from sensor's responses to tristimulus values. This is what we call in this work direct mapping (while indirect mapping is passing through spectral reflectance). We have also implemented the direct mapping for TFDs and used for the first time Kernel models for this mapping.

In the following chapters, we have subdivided the contents into the two main applications developed for the TFDs: color correction (direct mapping) and spectral reflectance estimation. We have studied them separately, and also comparing them directly for the color calibration purpose.

The novelties of this work are many. We have studied the performance of color correction and spectral reflectance estimation using a new and unexplored technology (TFDs). For these two application, we have implemented Kernel models which was never done so far for the first application and never with TFDs for spectral estimation. In addition, since previous works show that increasing the number of channels better results can be found [33] we have proposed, implemented and evaluated a multishot capturing system able to get multispectral images with some shots in a very quick time and transform them into color corrected images or estimated hyperspectral images (61 wavelength bands). Besides, since some previous works propose that the idea of narrowing down the responsivities of the sensors yields better colorimetric results

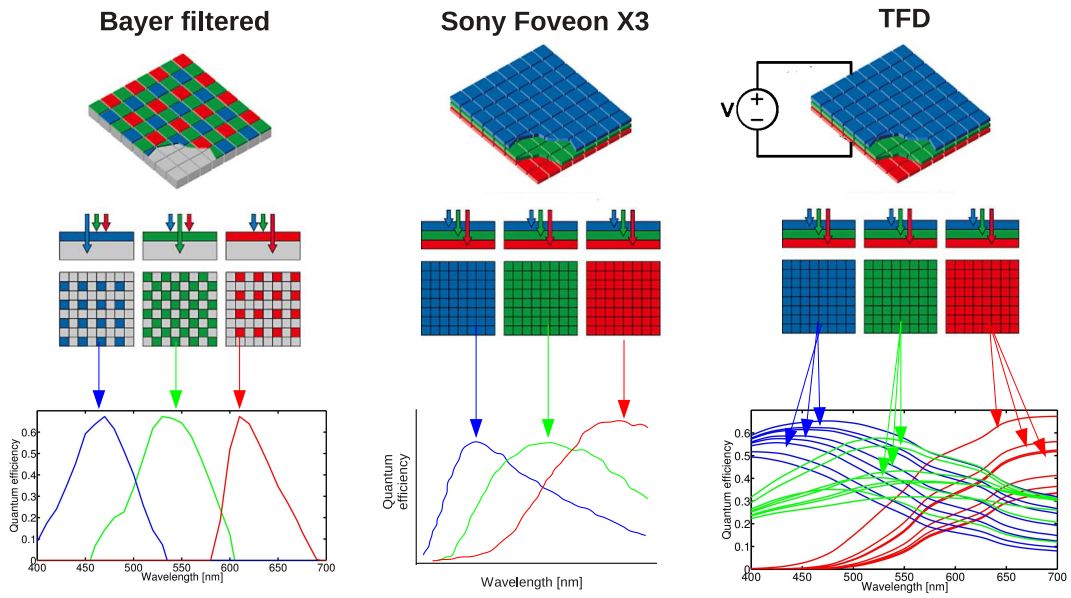

Figure 6: Comparison between Bayer-filtered (left), Foveon X3 (center) and TFD (right) sensors

[34][35][16], and TFD sensors have quite broad responsivities, we have proposed, implemented and evaluated a method for this purpose. There are approaches existing so far where each channel is filtered separately [36][37][38]. This is useful for Bayer-filtered sensors, since we can filter each single pixel with a different color filter. However, this approach is not valid for TFD technology, because whatever we do to one of the channels, will affect in the same way to the rest of them. Our solution filters all channels with a unique filter, narrowing down the responsivities and improving the colorimetric results. Furthermore, we have compared the performance of direct and indirect color correction for these sensors as well.

Summing up, we have studied the initial potential of TFDs for two different applications and we have improved it, proposing and implementing different approaches depending on the type of application the sensors can be used to. We have opened the doors for the future of the real-time one-shot full-resolution multispectral imaging.

2 Methods

2.1 Camera responses simulation

TFD technology is still under development. At present, there are still no cameras existing using such technology for imaging. The only existing prototype is mounted in a proto-board, and, even though it works as a sensor, it has neither the optics nor the rest of components needed for capturing an image. Therefore, although the simulated sensitivities that we are using in this work have been compared with those empirically obtained from the prototype, and they show a reasonable agreement (see Figure 7), we cannot have real camera response data to perform our study. Thus, we have implemented simulations of the camera captures, taking into account all noise processes present in the TFD sensors inherent to its technology.

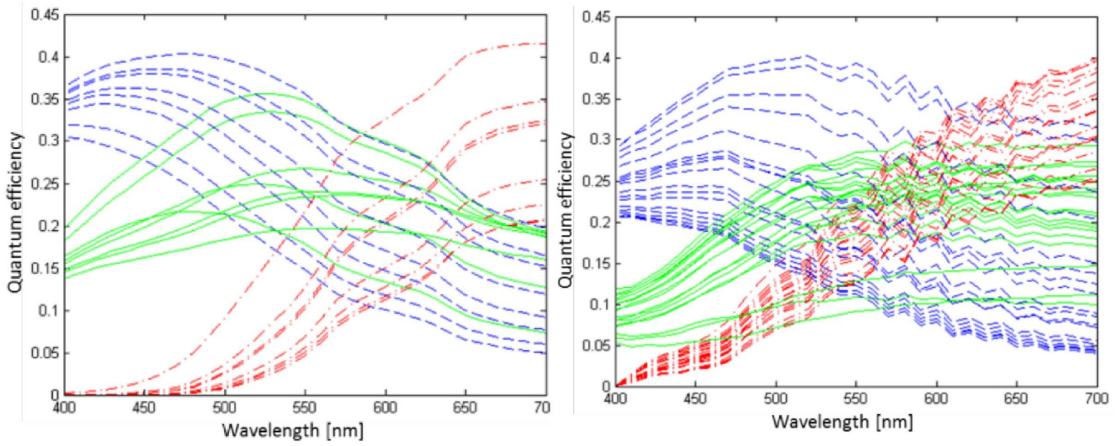

Figure 7: Simulated (left) and experimentally measured (right) TFD's quantum efficiencies

As this work is divided into two different applications, and each application is using different algorithms, two different approaches were used to simulate the capture process. The reason for this is to be able to study the algorithms performance in exactly the same conditions: the set of algorithms selected for spectral estimation requires preprocessing steps that are not necessary for the algorithms used for color correction, and these preprocessing steps make necessary to modify the way computational camera responses are calculated. In subsection 2.1.1 the capture process used for the color calibration application is explained in detail, as well as the system's set-ups used for it. On the other hand, the capture process used for the spectral reflectance estimation is explained in subsection 2.1.2.

2.1.1 Color calibration application

In this application, the algorithms used do not require the spectral sensitivities of the camera as inputs. This means that there is no need to perform any kind of scaling, normalization or stan-

dardization to the sensitivities. It is like this because these are empirical models. The empirical models try to map from sensors' responses to tristimulus values, only using as inputs the data empirically obtained from the sensors. So, in this case, the only required data are the tristimulus values and sensors' responses of the samples in the training set, and the sensors' responses of the samples in the test set. As we mentioned before, the sensors' responses were simulated. The simulations were done adding noise. In this way, the behavior of the TFD sensor is simulated more realistically. The noise processes involved in the TFD prototype are well characterized, so a realistic simulation can be achieved. The general equation for the calculus of the noisy responses is the following Equation 2.1:

$$\rho_{ik} = R_i \times L \times S_k \times \tau + \eta_{ik} \tag{2.1}$$

Where ρ_{ik} is the noisy sensor response of channel k to the i^{th} sample, R_i is the reflectance of the i^{th} sample, L is the spectral power distribution (SPD) of the illuminant used (the product of reflectance times illuminant is what we call color signal), S_k is the spectral sensitivity of the channel k, τ is the spectral transmittance of the filter (in case a filter is not used we drop this term), and η_{ik} is the additive noise component simulated for channel k and sample i.

In Figure 8, we show a flow-chart of the sequence of noise addition.

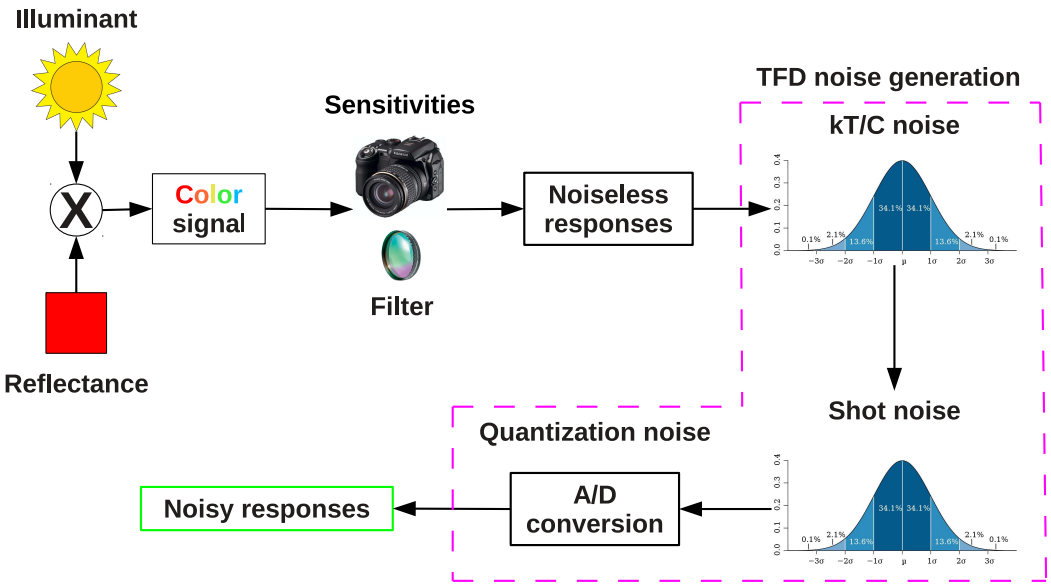

Figure 8: Flow-chart with the sequence of addition of noise for color calibration application

The first step of this process is the calculation of the noiseless sensors' responses. For this, we have a model which simulates the analog behavior of the TFD technology itself (for more information see [39]). Thus, when the color signal composed by the product of the illuminant times the reflectance being imaged (SPD) enters the sensor (see Equation 2.1), it is multiplied by the quantum efficiency of each of the channels separately (Q_{ek}) and integrated over the whole spectrum (from λ_{max} to λ_{min}). This yields a number which is scaled by the elementary

charge ($q = 1.6 \times 10^{-19}$C) and the illuminated area of the TFD sensor. This illuminated area is the product of the total area of the TFD sensor ($A_{total} = 10^{-6}$m $\times$ 10^{-6}m), times the fill factor which is inherent to this CMOS technology ($FF = 0.2$). The fill factor is the ratio between total area of the sensor and the active area collecting the photons. This will give as a result the photocurrent for channel k (I_k) calculated as (Equation 2.2)

$$I_k = q \times A_{total} \times FF \times \sum_{\lambda_{min}}^{\lambda_{max}} PSD \times Q_{ek} \tag{2.2}$$

Once the photocurrent is calculated, the next step is to convert it to output voltage. This is shown in Equation 2.3.

$$V_k = \frac{I_k \times T_{int}}{C_f} \tag{2.3}$$

Where C_f if the pixel feedback capacitance and T_{int} is the integration time.
Here an important issue is to be addressed. As it can be seen in Equation 2.3, the output voltage, which is the equivalent to the analogical response of the sensor, is directly proportional to the integration time. This means that the longer the integration time is, the higher the sensor response is. In the other hand, TFD technology has a saturation output voltage of $V_{dd} = 1.3$V. Therefore, there will not be a higher output than this value, and if the integration time is too long, the sensor's response will saturate and thus it will be clipped to V_{dd}. For this reason, for each of the 8 TFD set ups studied, a different integration time was calculated, in such a way that the output voltage is the highest possible without reaching any saturation. Thus, we set a value of $0.85 \times V_{dd}$, and if the voltage output was above this value, the integration time was reduced. In this way, we will have the highest possible signal level (corresponding to the highest possible SNR) without saturation. Since the sensors are supposed to be used to capture a whole scene, in this scene we may have very light or very dark reflectances. If we consider that our set of reflectances to be studied represent all possible pixels in one image, the simulation will be similar to a real scene capture. Since the integration time is biased by the brightest reflectance (because it is the one which can give saturation problems generating the highest sensor responses), the darkest reflectances will suffer lower signal-to-noise ratios (SNR). The lower the SNR is, the higher the noise is compared to the noiseless signal. This phenomenon is present in all imaging devices. We can explain it very clearly if we consider that each pixel has a component of noise which is fixed, and other which is variable and depends on the signal level. The darkest reflectances, will generate a lower variable noise component, but the fixed noise component will be the same than that of the brightest reflectances, so in the end the final SNR will be lower.
Coming back to Figure 8, we have already explained the first step, which is the simulation of the noiseless responses. Now the noise calculation and addition will be explained too. As commented in the beginning of Section 2.1, the model of capture will include three different kinds of noise. Each of them has a different variance. These variances are calculated as following.
For the kT/C noise, the variance is calculated as shown in Equation 2.4.

$$\sigma_{kT/C} = \sqrt{\frac{k \times T}{C_f}} \tag{2.4}$$

Where $k = 1.38 \times 10^{-23}$J/K, is the Boltzmann constant and $T = 300$°K is the absolute temperature (300°K is equivalent to 26.85°C). With this variance, a Gaussian distribution is generated centered in the noiseless response (V_k). From this distribution, a random value is chosen and this will be the mean for the Gaussian distribution generated for shot noise in the next step.
For shot noise, the variance is calculated as shown in Equation 2.5.

$$\sigma_{shot} = \sqrt{2 \times q \times (I_k + J_d \times A_{total} \times \alpha_k) \times \frac{T_{int}}{C_f^2}} \tag{2.5}$$

Here $J_d = 7 \times 10^{-5}$A/m^2 is the dark current density for this technology, and α_k a scaling parameter different for each channel k, with values of 0.5 for red channels, 0.3 for green channels and 0.2 for blue channels. These values are characteristic for TFD technology based sensors.
Once these two noises have been generated and added, the last step is to perform quatization. For this we assume $B = 12$ bits, which give us 4096 different possible levels of output. Since this is an analogical-to-digital conversion, some information is lost, and equivalently, some noise is added to the response after rounding from voltage to digital counts.
The variance of the quatization noise is also calculated as shown in Equation 2.6.

$$\sigma_{quant} = \frac{V_{dd}}{2^B \times \sqrt{12}} \tag{2.6}$$

Where V_{dd}, as we explained before is the saturation voltage in the output. The total variance for each channel k, including the 3 noises simulated is calculated as Equation 2.7 shows.

$$\sigma_{total_k} = \sqrt{\sigma_{kT/C}^2 + \sigma_{shot}^2 + \sigma_{quant}^2} \tag{2.7}$$

Therefore the signal-to-noise ratio for each channel k is calculated as shown in Equation 2.8.

$$SNR_k = 20 \times \log \frac{V_k}{\sigma_{total_k}} \tag{2.8}$$

After this process we finally obtain the noisy sensors' responses. Besides, in this simple way, we can control either if our integration time is too long, so we get clipping, or too short, so the SNR is very low. A minimum acceptable limit for SNR is 26dB as used previously by other authors [40]. This limit corresponds to a noise variance of 5% the value of the noiseless signal. Special care must be taken in the cases when we place an optical filter in front of the sensor. In these cases, the simulation is exactly the same, but the quantum efficiencies included in Equation 2.2 would be weighted by the spectral transmittance of the filter as explained in next subsection 2.1.1. Since the values of transmittance are in the range $[0 \cdots 1]$, the signal level will always be lower than in the unfiltered case, and thus a higher T_{int} can be set without saturation.

Systems' set-up

Up to this point, we have assumed a 3 channels single shot capture without considering what changes could be introduced in the responsivities to improve the colorimetric accuracy of the system. However, many authors in the literature show different approaches for improving the colorimetric performance of sensors. Sensor sharpening [34] [41] has been addressed in many papers. It refers to using a linear transformation of camera responses (RGB) in order to improve

computational color constancy methods based on independently scaling the three sensor channels. Alternatively, many proposals have addressed the possibility of filtering the sensitivities of our sensors by applying different color filters for each channel. There are some techniques for selecting the best among a set of available filters [42] [43], or for computing optimized filter transmittances [37] [36]. All of these approaches assume that we have a Bayer-filtered sensor. The difference between Bayer-filtered sensors and TFD sensors, is that the first ones don't have full spatial resolution for each channel and they need demosaicking methods to get the complete color information for each pixel. In the other hand TFD sensors have full spatial resolution without the need of interpolation. In our case, we need to preserve this full spatial resolution, so we cannot add different filters to each channel. All the responsivities have to share a common filter. In [16], a color-optimum pre-filter is proposed to improve the color performance of the Sony Foveon X3 sensor, which is based in the same technology of wavelength-dependent absorption depth of silicon as the TFD sensor and thus their sensitivities are also quite broad [12]. Unfortunately the authors in this reference do not provide any details on how this filter was designed. Therefore, following what it was done before for TFD sensors [17], we introduced a filter which is a combination of an IR and UV cut-off filters (see Figure 9), with the purpose of narrowing down the responsivities of our channels.
Also it was demonstrated in [33] that using more than 3 channels for color correction in presence of noise is helpful. Thus, we also included a multishot approach in our computations. Summarizing, for the first application, we have experimented with different set-ups possibilities of the system. We have used 4 different set-ups: single shot (*single*), single shot plus filter (*single filtered*), double shot (*multishot*) and double shot plus filter (*multishot filtered*).
The *single* shot approach is the one explained so far in this section. We take only one shot which gives us 3 channels and compute the color calibration from them.
The *single filtered* set up, is exactly the same as the *single* one, but, before calculating the sensors' responses, we weight the quantum efficiencies of the sensors with the spectral transmittance of the filter ($\tau_{filter}(\lambda)$ shown in Figure 9). So if we have a look to Equation 2.2, we would substitute Q_{ek} by Q^{F}_{ek} being $Q^{F}_{ek}(\lambda) = Q_{ek}(\lambda) \times \tau_{filter}(\lambda)$.

In the *multishot* approach, the same sequence followed in Figure 8 to calculate the sensors' responses for the *single* approach is repeated 2 times. One for each set of 3 sensitivities. Actually this simulation would be like in the real case since they would be two independent shots taken in a very short time and varying the applied voltage (and so the sensitivities) between shots.

Finally for the *multishot filtered* set up, the way to simulate is to combine the previous two. We repeat for two independent shots the sensors' responses calculation multiplying the quantum efficiencies by the spectral transmittance of the filter.

We can observe the impact of adding the optical filter in the spectral quantum efficiencies of the sensors, weighted by the illuminant (D65. See Section 2.3.1) in Figure 10.

Comparing the filtered and unfiltered version of each channel we see how using the filter makes the spectral quantum efficiencies be narrower. This effect would be similar to the one obtained by a conventional spectral sharpening approach.

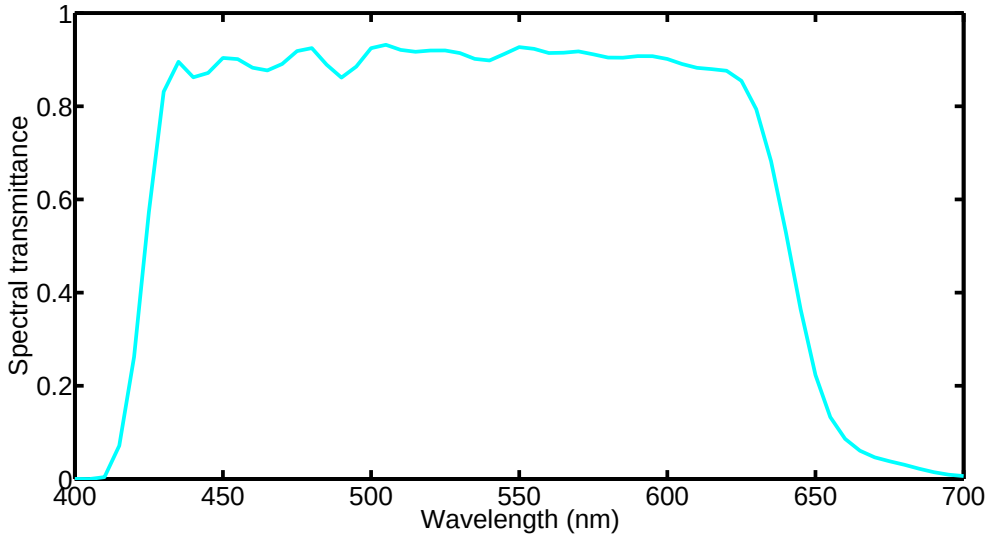

Figure 9: Spectral transmittance of the optical filter applied in front of the sensor $\tau_{filter}(\lambda)$

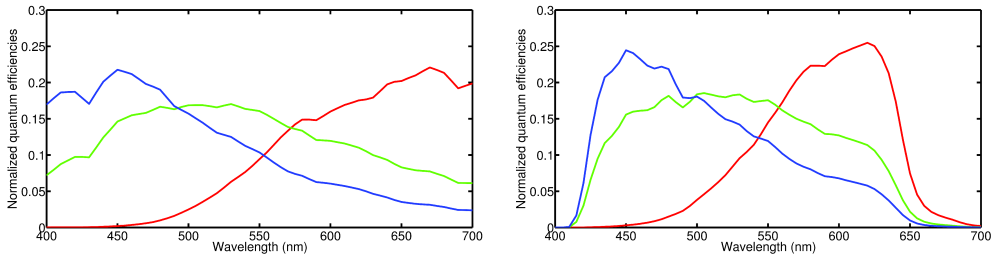

Figure 10: Spectral quantum efficiencies weighted by D65 for the unfiltered (left) and filtered (right) cases

2.1.2 Spectral reflectance estimation application

We have seen in the previous subsection how the sensors' responses are simulated for the color calibration application. In the case of the spectral reflectance estimation application they are simulated differently. The process of adding the noise is equivalent, but they are computed in a slightly different way due to the needs posed by the preprocessing steps applied for some of the algorithms.

For this application, the algorithms studied are Wiener, Pseudoinverse and Kernels (Polynomials and Gaussian). In subsection 2.2, all the algorithms will be explained in detail, but in order to understand why we calculate differently the responses here, we need to explain briefly some characteristics of the Wiener model. This regression model is a physical model. This means that it is using explicitly the sensitivities of the sensor to calculate the regression. But before using them as input to the algorithm, we need to perform a normalization as a preprocessing step. In this case the normalization needed is just to set the quantum efficiencies to norm 1 channel by channel. This is easily done with Matlab's routine *normc.m*.

Since the model explained in previous Subsection 2.1.1, uses physical parameters which cannot be scaled because they would not correspond any more with the real behavior of the TFD, the way of simulating is slightly different. (see the flowchart shown in Figure 11).

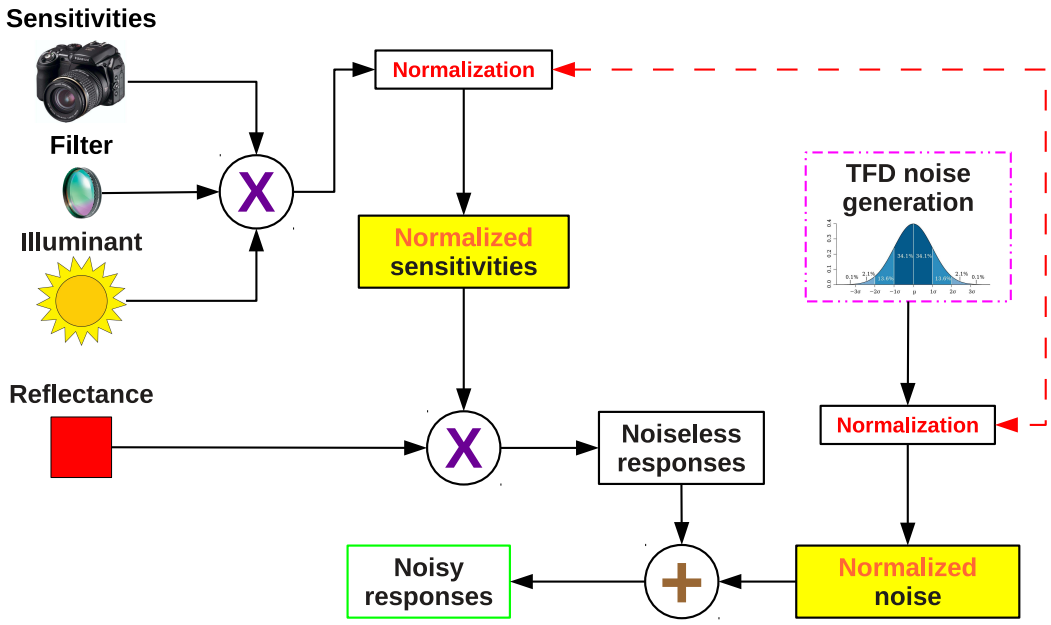

Figure 11: Flow-chart with the sequence of addition of noise for spectral reflectance estimation application

As we can see, in the calculation of these noisy responses, there are two components calculated separately and added. These components are the noiseless responses and the noise. For the calculation of the noiseless responses (ρ_k), the camera sensitivities weighted by the illuminant, are normalized to norm 1 channel by channel. This normalization is just scaling separately each of the sensitivities by one number. Equation 2.9 shows how to calculate these responses.

$$\rho_k = R \times \text{normc}(Q_{ek} \times I_{D65}) \tag{2.9}$$

Where I_{D65} is the spectral power distribution (SPD) of the illuminant used (D65). If a filter is included, quantum efficiencies of each channel Q_{ek} will be substituted by the ones including the filter transmittance (Q^F_{ek}) as explained in subsection 2.1.1.
On the other hand, the noise is generated as before, calculating the variance of total noise as shown in equation 2.7, and scaling this variance with the same factors used for the responsivities' scaling. Therefore in Figure 11, the two normalization blocks in red letters (joined by a dashed red line) are equivalent.
In this application, three measurements of noise impact were calculated. The first one is the signal-to-noise ratio of the responses of the brightest reflectance calculated as shown in Equation 2.8. The second one is the signal-to-noise ratio of the noiseless responses ρ_k calculated in Equation 2.9, over the scaled variance of Gaussian noise added to them to calculate the noisy

responses (See Equation 2.10).

$$SNR_{k_{scaled}} = 20 \times \log \frac{\rho_k}{\sigma_{total_k} \times \beta_k} \tag{2.10}$$

Where β_k is the scaling factor for each channel k and is calculated a shown in Equation 2.11.

$$\beta_k = \frac{normc(Q_{ek} \times I_{D65})}{Q_{ek} \times I_{D65}} \tag{2.11}$$

And the third one is the ratio or percentage between the absolute difference between noisy and noiseless responses, over the noiseless responses. This measurement can be called RAN (Relative Amount of Noise).

$$RAN(\%) = 100 \times \frac{|\rho_{k,noiseless} - \rho_{k,noisely}|}{\rho_{k,noiseless}} \tag{2.12}$$

Once the sensor responses are calculated with and without noise, we are prepared to use them as input for the different algorithms for the spectral estimation. In next section we will explain the details about each of the algorithms used for both applications.
The first two SNR measurements, are equivalent. The first measurement is the ratio between the signal corresponding to the brightest reflectance (the one with higher integral over the visible range, which will generate the higher signal-dependent noise component), and the variance of its noise (σ_{total_k}). This variance is used to calculate a Gaussian distribution, and from it, we choose a random value (to mimic real conditions). This means that, this value of SNR would be an average value, and in each of the single cases for every sample and every channel, the real SNR will be a bit higher or a bit lower than this value. This measurement is used then to adjust the value of the scaling factor (β_k in Equation 2.11) if needed. In this way, if we adjust β_k so that we equalize both SNR measurements, we ensure that the noise calculated for the sensors' responses generated with the scaled responsivities is in the same relative level as that physically generated for the brightest sample of the whole set of samples. Thus we can say we are simulating in "pessimistic" noise conditions (worst case). The last measurement in percentage is calculated to know exactly the ratio between the noiseless response and the noise that finally was added after the random selection among the Gaussian distributions' values generated.

2.2 Models

In this section we are going to talk about the different algorithms used in both applications studied in this work. We were specially interested in the study of the kernel methods, since they are a recent proposal and the use of these algorithms is not widely extended yet. Besides, it has been demonstrated that their performance is really good for the calculus of non-linear regressions [44][22][23]. However, only studying Kernels we are not able to ensure that their performance is better or worse than that of other more popular algorithms.
For the color calibration application, many authors have addressed the problem by means of Polynomials and Neural Networks among others [26][45][46][31]. For the spectral reflectance estimation, methods like Pseudoinverse or Wiener are two of the most popular ones used by several researchers [40][47][48]. We decided to use these algorithms together with Kernels, and

compared their performances, pros and cons of each of them respect to the others. We will divide this section into two subsections, one for each application. The criteria to select the algorithms was based in the fact that none of them use dimensionality reduction techniques such as Principal Component Analysis (PCA). The use of these techniques adds an additional parameter (number of basis vectors) which needs to be fixed during the training phase. We have preferred to focus instead, on the parameters inherent to the algorithms themselves. Summarizing, we included in our selection of models widely used and relatively simple ones together with other more computationally-heavy regression algorithms.

2.2.1 Color calibration application

As mentioned before, the most widely used algorithms for this application so far are Polynomials and Neural Networks. We will explain these two first and then we will give some details about the more recent proposal called the Kernel approach.

Least squares fitting with polynomially expanded sensors' responses

Polynomial regularized least squares fitting is a method which has been used not only for color calibration [26] but also for spectral reflectance estimation [49]. It has been proved to work better than some more complex algorithms like certain kinds of neural networks [45]. This method was studied only for this first application though. As we will see in the results chapter, Kernel methods outperform polynomials, and since the optimization of the polynomials was quite slow, we stopped using them. Moreover, the use of Polynomial Kernels can be seen as a mathematical equivalent of this polynomial regression where every monomial term has a certain weight. But what is a Least Squares fitting and how can it be done?. To answer this question let us start from the simplest explanation. Our problem, as stated in chapter 1, is to map from device-dependent space (RGB responses of our camera) to device-independent space like the standard CIEXYZ. In the simplest case we assume that our feature vector is $\Phi = \Phi_1$, where

$$\Phi_1(x) = (x_1, x_2, x_3)^T = x \tag{2.13}$$

x_i correspond to device dependent coordinates for the three channels of the camera (RGB), and the symbol T denotes the transpose operator of a matrix. Therefore we can say that the matrix of device dependent coordinates is

$$X = (\Phi_1(x_1)^T \dots \Phi_1(x_m)^T)^T \in \Re^{m\times 3} \tag{2.14}$$

where m is the number of samples. On the other hand, the matrix of device independent coordinates $c_i = (c_{i1}, c_{i2}, c_{i3})^T$ (CIEXYZ) for the m samples is

$$Y = [y_1, y_2, y_3] = (y_1^T \dots y_m^T)^T \in \Re^{m\times 3} \tag{2.15}$$

Thus, least squares solution for $\min_{v_1,v_2,v_3} \sum_{i=1}^{3} \|Xv_i - y_i\|_2^2$ is written as a matrix

$$V = [v_1 \dots v_3] = (X^T X)^{-1} X^T Y \in \Re^{3\times 3} \tag{2.16}$$

and if we want to include the regularization term (λ) the equation is transformed to

$$V = [v_1 \dots v_3] = (X^T X + \lambda I_n)^{-1} X^T Y \in \Re^{3\times 3} \tag{2.17}$$

where I_n is the identity matrix of dimension n. Including regularization to polynomial models is not usually applied to polynomial fitting approaches in the context of color calibration. The aim of regularization is to diminish the effect of noise in the data, and provide good generalization for the regressions. If our training data set is small, and we use a high degree polynomial to fit this data, we may fit too accurately to this training set, but after fail to generalize for the test data set (this is called overfitting in machine learning theory). Regularization parameter controls the wigliness of the hypothesis, so that the properties of the resulting function to overfit the data are under control.
Finally, the evaluation or the transformation matrix to compute the device-independent coordinates of the additional test samples is computed as

$$\hat{c} = V^T \Phi_1(x) \tag{2.18}$$

So if we have a training set of samples for which we know both their device-dependent and their device-independent coordinates, we can build the calibration matrix V as shown in Equation 2.17. Then if we get the device-dependent coordinates of a set of new samples for which we want to estimate their device-independent coordinates (test samples), we will use them as $\Phi_1(x)$ in Equation 2.18. An scheme of this work-flow is shown in Figure 12

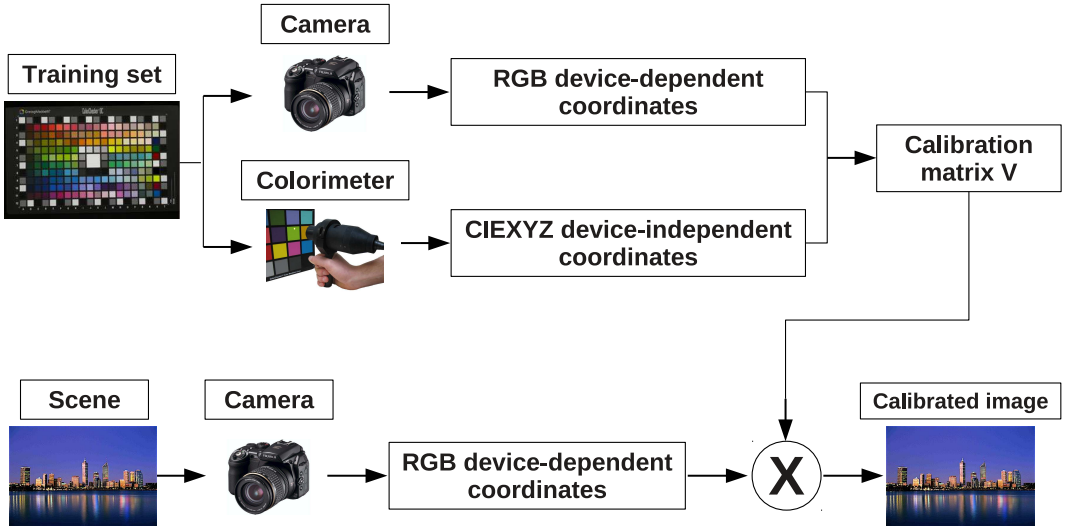

Figure 12: Flow-chart of the linear least squares fitting applied to color correction

This is called the linear regression approach, because to construct the calibration matrix, we are using a feature vector $\Phi_1(x)$ that contains directly the sensors' responses as they are (see Equation 2.14). But what if, instead of using only these sensors' responses, we build a new feature vector $\Phi_2(x)$ in which we include also the polynomial expansions of them?. This is the

philosophy of the polynomial fitting approach. So the way of calculating the calibration matrix V and to recover the CIEXYZ coordinates $\hat{c}$, is the same as explained so far, but now, the new feature vector has a bigger number of terms and looks like

$$\Phi_2(x) = (x_1, x_2, x_3, x_1x_2, x_2x_3, x_1x_3, x_1^2, x_2^2, x_3^2, \dots)^T \tag{2.19}$$

As we can see in Equation 2.19, now we have the monomial terms corresponding to the powers of the sensor responses, mixed with products of them, and up to a certain monomial degree d. The regression is done exactly the same way, taking into account that in the step of evaluation of the calibration (Equation 2.18), we also have to move the sensors' responses of the samples to be calibrated into the polynomial feature space. The scheme of this slightly different approach is shown in Figure 13.

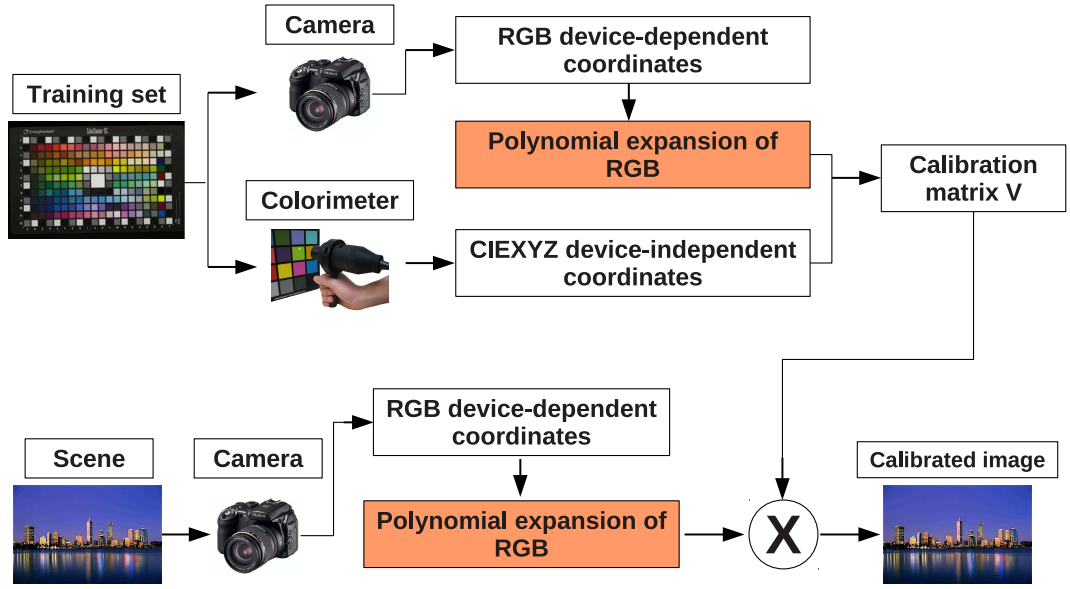

Figure 13: Flow-chart of the linear least squares fitting

An important fact must be mentioned in this point. We have seen that polynomial expansion offers the possibility to increase the number of terms used for the regression. It is clear that including non-linear features, more complex patterns can be found in our data. But an important issue comes with this *a priori* advantage. We have to decide the degree d of the polynomial to be used. And furthermore, we have to select which of the terms of the polynomial expansion to use and which ones to neglect. A carefully selected set of them may perform better than using all of them, but the difficulty here is to choose this set. For our study we just used all the terms up to a certain degree. This is called the complete polynomial expansion of degree d. As we will see in Chapter 3, the performance of this approach is practically the same as that of Polynomial Kernels, so we considered non-necessary to optimize the feature selection step. Actually, it can be demonstrated that Polynomial Kernels are equivalent to a polynomial least squares regression in which each monomial term has a certain weight [50].

Artificial Neural Networks (ANN)

The use of ANN is very popular in machine learning problems [51]. The first proposals arised from trying to mimic the behavior of the brain. They were very widely used in the $80's$ and early $90's$, and their popularity diminished in the $90's$. But they are having a recent resurgence nowadays [52]. Many authors have used neural networks before to perform color calibration in digital imaging in general [25] or more in particular for digital cameras [46] or scanners [31]. The global structure of the networks and the mathematical models underneath vary depending on the problem to be addressed. Let's have a look at the neuron's model and also the architecture of a network and after we will explain the differences between the two kinds of networks used for this study.

The neuron's model extracted from [52] is shown in Figure 14.

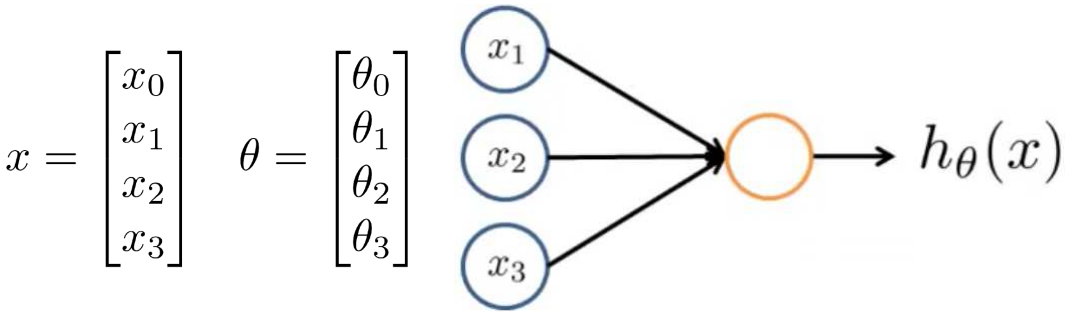

Figure 14: Model for a single neuron of a neural network

As it can be seen, the neuron (orange circle) is modeled as a unit which has 3 inputs on the left, and one output on the right. The neuron is fed by the inputs and performs some computations. The output is computed as a function of the values of the inputs (x vector), and the weights of the input connections (θ vector). We can see how there is also an input value x_0 with its weight θ_0 which are not shown in the scheme. This is called the bias unit. Its value is always 1 and normally it is connected to all the neurons in the network. $h_\theta(x)$ is called the activation function. The most popular one is the sigmoid function shown in Equation 2.20.

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \tag{2.20}$$

Once we have seen how a single neuron works, we can continue with the model of a network. A neural network is just a set of neurons like this one shown, interconnected via inputs and outputs, so that the output of a neuron is an input for others. The architecture of the network is organized by layers. There are three different kinds of layers: input layer (it is only one), hidden or intermediate layers (they may be any number), and output layers (it is also only one). In Figure 15 we can see the architecture of a complex network extracted from [52].

As we see, it is easy to distinguish between the different kinds of layers present in an ANN. For the color calibration problem, since we want to map from camera sensors' responses to CIEXYZ, the number of neurons in the input layer would be the number of channels or sensors, and the number of neurons in the output layer would be 3. One for each tristimulus value.

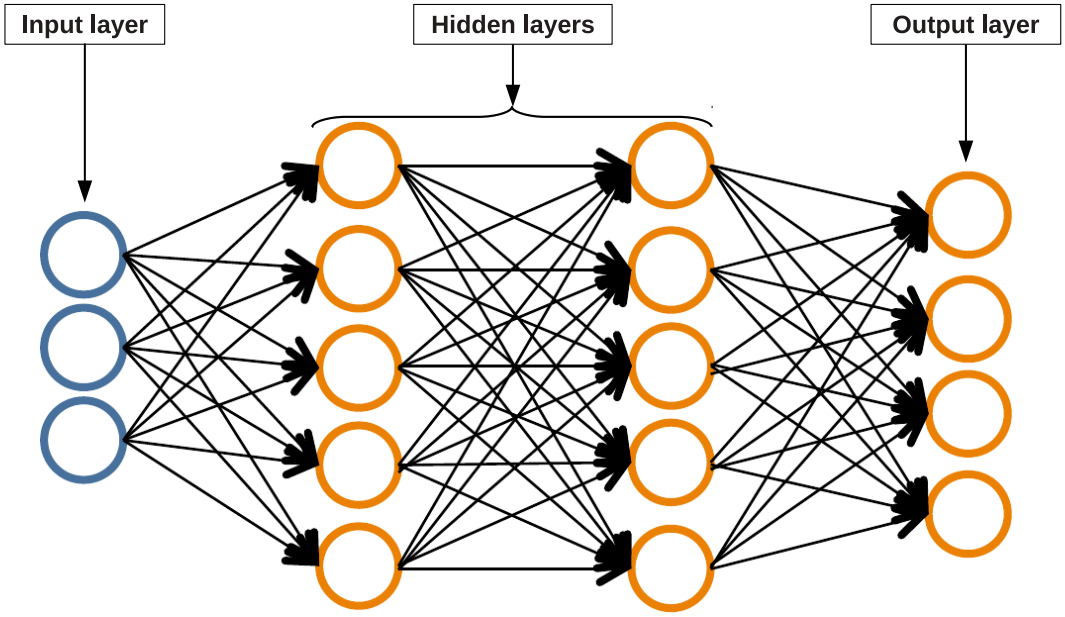

Figure 15: Model for a more complex neural network

To render easy the understanding of how a neural network is fed forward, we present in Figure 16 the example of a simple network with one hidden layer extracted from [52].

As we can see, this network has three neurons in the input layer (or layer one), three neurons in the hidden layer (or layer two) and one neuron in the output layer (or layer three). The nomenclature of the neurons output values is $a_i^{(j)}$, where i is the number of neuron and j is the number of layer. For the weights matrix we use the terms $\Theta_{ij}^{(k)}$. Now i is the number of neuron that this weight matrix is in input to, j is the number of neuron of the previous layer this weights come from , and k is the layer of the neuron that has this weight matrix as input. So the output of each neuron is computed as a function of the weights corresponding to its inputs and the values of the neurons from previous layer ($g(\Theta, x)$). Taking into account that the bias unit (x_0) is not represented, the output values for the neurons in the hidden layer of the network represented in Figure 16 would be calculated as

$$a_1^{(2)} = g(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3) \tag{2.21}$$

$$a_2^{(2)} = g(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3) \tag{2.22}$$

$$a_3^{(2)} = g(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3) \tag{2.23}$$

And once these intermediate layer's neurons' output values are calculated, we can compute the value for the final hypothesis or final output of the network ($h_\Theta(x)$) as

$$h_\Theta(x) = a_1^{(3)} = g(\Theta_{10}^{(2)} x_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)}) \tag{2.24}$$

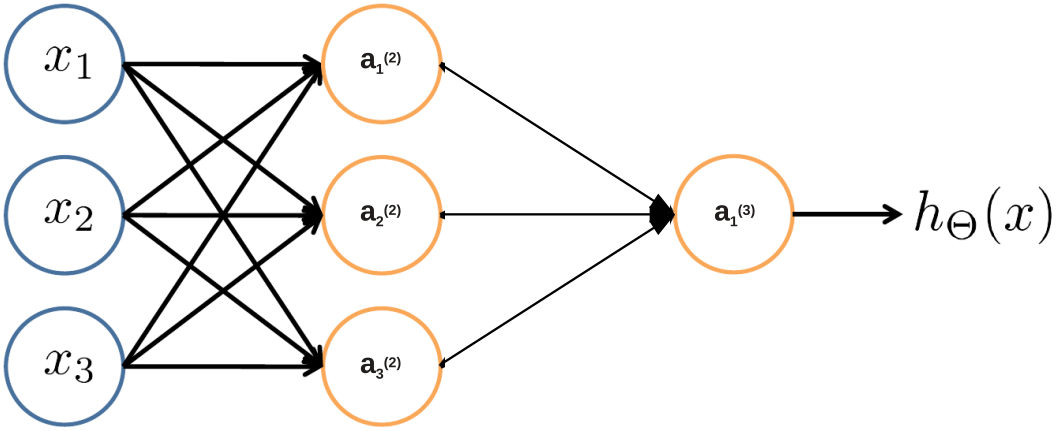

Figure 16: Model for feed forward propagation in a neural network

This procedure is called a feed forward approach, because we feed the network in the input layer, and then we advance forward to next layers. The initial values for the weights matrices are randomly selected. After this we compute the error and update the weights back from the final layer to the first one, and this is where the name Back Propagation comes from. So a neural network which is fed forward (from input to output layer) and for which its error is back propagated (from output to input layer) is called a Feed Forward Back Propagation Artificial neural network.

Now that we know how an ANN works, we can explain the two kinds of ANN used in this study. The first one is the *Radial Basis Functions Neural Network* (RBFNN). In these networks, the input to the activation function (n), is the vector distance between its weight vector Θ and the input vector X, multiplied by the bias. Then, the activation function is

$$\text{radbas}(n) = e^{-n^2} \tag{2.25}$$

The shape of this Gaussian activation function can be seen in the left side of Figure 17.

These networks only have as free parameters the number of neurons to be added to the unique hidden layer and the spread of the Gaussian function represented in Figure 17. The optimization of this spread was studied in [53], but in our case we will follow a more simple way to optimize. The Matlab function used to train RBFNN (*newrb*) also allows to set a goal Root Mean Square Error. The training process is iterative, and in each iteration a new neuron is added to the hidden layer. Then the error is computed between the output of the network and the real color coordinates of the sample whose sensors' responses were fed as input. According to it the weight matrices are updated sample by sample. If in one iteration the error computed is lower than the value set as goal error, then the training stops. This is done to save some optimization time when the output allows some tolerance. In our case we are not deeply concerned about the computation time, so we set this goal error to zero.

For the *Feed Forward Back Propagation* networks (FFBP) [46] [31] [54], there are two changes with respect to the already explained RBFNN. One is that the architecture allows any number of

hidden layers and any number of neurons per layer. Normally the number of neurons is the same for every hidden layer. In this study we considered the ranges from 1 to 5 hidden layer to find the optimal architecture of the Neural network, and from 3 to 8 neurons per hidden layer. The second difference is that the activation function for the neurons is the sigmoid function shown in Equation 2.20. The shape of this function can be seen in the right side of Figure 17.

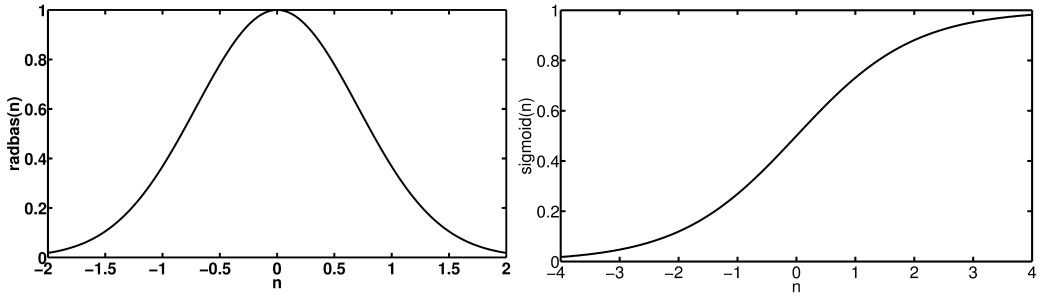

Figure 17: Activation function for FFBP

These networks have as free parameters the number of hidden layers and the number of neurons per hidden layer. They are called FFBP, but actually, the RBFNN are also feed forward networks and the error is back propagated. But in this case the activation function is the typical sigmoid, so there is no special name for them. In both kinds of networks, the training algorithm used was the Matlab's default Levenberg-Marquardt back-propagation (LM) for minimization of functions, which was found to perform better than gradient descent, because the error surface is full of local minima, and the gradient descent was usually not converging to a global minimum. LM interpolates between the Gauss–Newton algorithm (GNA) and the method of gradient descent. The LM is more robust than the GNA, which means that in many cases it finds a solution even if it starts very far off the final minimum even though in some conditions it is slower than GNA. For more details about LM algorithm visit [55].

Kernels

There are many references in the literature which describe the Kernel approach in a machine learning context [44][50]. It is though a quite recent proposal for spectral reflectance estimation not used yet by many authors [22] [23]. We will not give a full detailed description of Kernel theory in this section. We will present the models of the different Kernels used in this study, and also comment on its main advantages over alternative algorithms.

A Kernel method is just a way of mapping some data, from a space in which the patterns are not linear, to a different space, called feature space, where the same patterns mentioned before are linear. This mapping is done in order to be able to use linear regression models to solve the main regression problem. This mapping, or recoding is represented in Figure 18 (extracted from [44]).

So the strategy is to embed the data into a new feature space where the patterns can be discovered as linear relations. This is done in a modular fashion. Any Kernel method solution comprises two parts: the first one that performs this mapping into the feature space, and a

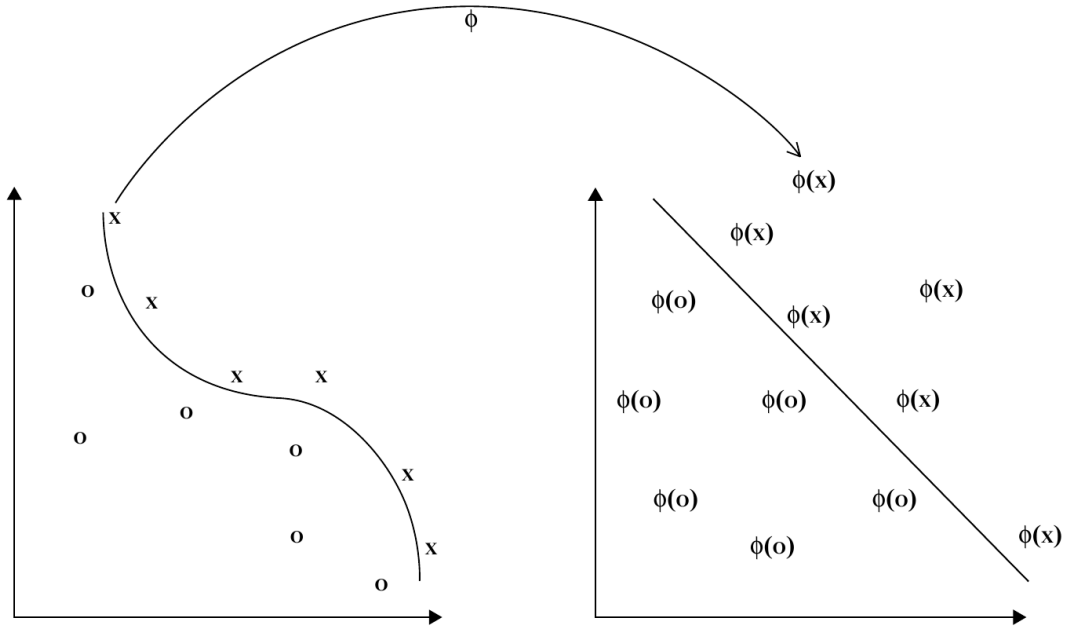

Figure 18: Mapping or recoding of data into feature space via function ϕ for a classification problem

second one which is a learning algorithm designed to discover linear patterns in such space. It is a smart approach because the linear regression methods to be used in the feature space are widely used and very well known and understood, and also because there is a computational shortcut allowing us to efficiently perform this mapping without the need of computing the coordinates of the embedded points. We just need the pairwise inner products, that can be computed efficiently and directly from the original data points using a Kernel function (Kernel trick).
The solution we will use for the regression once the mapping to the feature space is done is the least squares fitting shown in Equation 2.17. But in this case, instead of the data vectors X, we will have their embedded versions K and K^{test}. The Kernel function $\kappa(.,.)$ calculates an inner product in the feature space: $\langle \Phi(x), \Phi(z) \rangle = \kappa(x, z)$. Thus we can compute the embedded versions of training and testing sets respectively as

$$K_{ij}^{train} = \kappa(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle \tag{2.26}$$

$$K_{i}^{test} = \kappa(x_i, x^{test}) = \langle \Phi(x_i), \Phi(x^{test}) \rangle \tag{2.27}$$

Where K_{ij}^{train} in the Gram matrix of the training data in feature space, and K_{i}^{test} contains the kernel evaluations between the training set and the test point Therefore in the end, if we substitute Equations 2.26 and 2.27 in Equation 2.17 the regression will be calculated like

$$\hat{Y} = K^{test}(K^{train} + \lambda I_n)^{-1} Y \tag{2.28}$$

And now it is when the ultimate advantage of the Kernel methods is shown. We can change

the approach by just changing the Kernel functions used ($\kappa(x, z)$). In our study we have used two different functions: the Gaussian and the Polynomials kernels. The first are computed as

$$\kappa(x, z) = \exp(-\frac{\|x - z\|_2}{2\sigma^2}) \tag{2.29}$$

And it can be demonstrated that using this Kernel function, the approach is equivalent to a one hidden layer Radial Basis Functions Neural Network [23]. Here $\sigma > 0$, and it controls the area of the Kernel. If it is too small we reach overfitting and thus loose generalization. If σ is too big then we can risk oversimplification.
On the other hand the polynomial Kernels are computed as

$$\kappa(x, z) = (\langle x, z\rangle + 1)^d \tag{2.30}$$

In this case d is the degree of the polynomial. As we have seen, in both cases we had two free parameters for our models. For the Gaussian Kernels they are the regularization parameter (λ) and the spread of the Gaussian function (σ). For the Polynomial Kernels the free parameters are the same regularization parameter (λ) and the degree of the polynomial (d).
As it was seen here, the Kernel solutions have advantages over the others, and thus even if results are similar, still it would be advantageous to use them rather than other approaches. Simpler models are usually better if same accuracy is achieved. The Kernel models are powerful, but still relatively simple methods if we compare them with some others like Neural Networks for instance. Besides, Kernels unify polynomials and RBFNN (among several other models we haven't used in this study). They have a solid mathematical background. Moreover, regularization is also included in these models, which helps controlling the fitting of the hypothesis against the effect of noise. All these reasons make Kernels be better approach "a priori". In Chapter 3, we will see how the results obtained in the different experiments confirm these facts.

2.2.2 Spectral reflectance estimation application

In this application the same Kernels are used as in the color calibration application (see Section 2.2.1), plus two more simple approaches.

Pseudoinverse

Pseudoinverse is the most direct way of linearly mapping sensors' responses into spectral reflectance. This approach is just a least squares fitting with first degree polynomial and without regularization. It has neither possible parameterization, nor optimization, nor regularization to control the fitting. We just need a training set from which we know the sensors' responses (ρ_{train}) and the corresponding reflectances (R_{train}). With this we calculate the mapping matrix as

$$A = R_{train} \times \text{pinv}(\rho_{train}) \tag{2.31}$$

where pinv(M) denotes the Moore-Penrose pseudoinverse of the matrix M. Once this matrix A is calculated, the recovery of the reflectances is done by simply multiplying it back by the

sensors' responses of the testing set (ρ_{test}) like

$$\hat{R} = A \times \rho_{test} \tag{2.32}$$

This is all from this approach. It is simple, fast to implement, easy to understand and a reasonable approach when the problem we are dealing with can be reasonable solved by using a linear function for fitting and there is relatively low noise present in the system. But this will not always be our case, and we will see in the Chapter 3 how Pseudoinverse is not the best approach under the presence of noise in the system.

Wiener method

The models explained are called empirical models because the only data they need to perform the mappings, either from sensors' responses to color coordinates (color calibration) or from sensors' responses to spectral reflectance (spectral reflectance estimation) is data empirically obtained, plus, the don't use any descriptive model of the behavior of the system. This data mainly consist in a set of samples from which we know their sensors' responses (ρ_{train}) and their color coordinates (or spectral reflectance R_{train}), called training set, and also the sensors' responses for the samples we want to estimate (ρ_{test}). Even though in this study, the sensors' responses are obtained via simulations due to the fact that the TFD system is still a sensor prototype without the components needed for imaging, the models can still be called empirical.
Wiener is instead a physical model [24][56], and the difference is basically that, rather than using directly the empirical data obtained in real (or simulated) cases, it requires as inputs the spectral responsivities of the sensors (S) and the Spectral Power Distribution (SPD) of the illuminant (L). The model computes the recovery matrix (W) as

$$W = (R_{ss} \times SL) \times (SL^T \times R_{ss} \times SL + \lambda I_n) \tag{2.33}$$

Where Rss is the covariance matrix of R_{train}, SL is the matrix of spectral sensitivities of the sensors weighted by the illuminant and λ is a free parameter corresponding to noise in response values. After W is calculated, the estimated samples are computed as:

$$\hat{R} = W \times \rho_{test} \tag{2.34}$$

It appears clearly that Wiener method is directly related with the Pseudoinverse explained in Subsection 2.2.2. There are some papers comparing their performances [48] and even proposing new hybrid methods between them [47]. However there are also some important differences between them, as in the Wiener method, the sensitivities and the illuminant are explicitly used and also we have a parameter representing the effect of noise. The mathematical models are equivalent if instead of computing the conversion matrix using the More-Penrose pseudoinverse as $pinv(\rho_{train})$ in Equation 2.31, we do it like

$$A = R_{train} \times \rho_{train}^T \times (\rho_{train} \times \rho_{train}^T)^{-1} \tag{2.35}$$

and we set the λ parameter in Equation 2.33 to zero. However we decided to use the More-Penrose pseudoinverse calculation, because due to the high number of sensors used (24) and

also the similarity between them, we take the risk of having a matrix which is close to singular and thus non-invertible.
We can finish saying that Wiener is a very well known and widely used physical method which is also a base for more recent proposals [24], and thus it is worth to implement it and compare its performance with others.

2.3 Spectral data

After explaining the important aspects of how the image capture is realistically simulated with TFD sensors and also some details about the algorithms studied, now it is time to study the spectral data we have selected for this study. As it was said in the Section 2.2, all the algorithms, either empirical or physical models, need some spectral data to be trained and also evaluated. It is important that these data is as closely related as possible with the data of the application the system is intended for. It would not make sense if we train the spectral estimation algorithms with reflectances of plants, if afterwards we are going to use it (or evaluate it) with reflectances of asphalt. There are many spectral databases available online for different purposes. Some of them have already been studied for spectral color purposes [57]. For the two applications studied in this work we have selected different reflectances sets. We will divide this section into three parts corresponding to the illuminant used in this study, and the two applications and we will explain the main reasons for having selected each data set.

2.3.1 Illuminant

The illuminant used for the simulations was the standard D65. Its spectral shape can be seen in Figure 19. For the calculation of the TFD responses it is needed to make a small conversion to photon counts, but it is straight forward. It is also used to compute the color coordinates needed for the color metric errors computed. So in the end we are assuming average conditions of natural daylight illumination in our simulations which is adequate for a wide range of possible applications.

2.3.2 Color calibration

For the color calibration application we want to map from sensors' responses to color coordinates. Color calibration is a very general problem for any kind of image capture systems, and so our interest in selecting the data is to build a collection of samples covering many kinds of surfaces and materials. For this reason, a suitable database to be used for general purpose could not only contain one of the standard color charts available in the literature. These color charts are accessible to everybody and also their values are very accurately calculated so we can be sure that the ground truth is quite reliable. For this application then, 5 different color charts have been used (see Figure 20). We eliminated in each chart those reflectances with values out of the range $[0.1 \ldots 1]$. The upper bound is necessary because we assume that reflectances in our work have only positive values below 1. The lower bound is to avoid problems with matrix inversion derived of very low values in our data set. The charts used were: Munsell Book of Color (1269 samples) [46] [58] [38], Natural Color System (NCS, 1750 samples), Pantone ®(922 samples), Greta Macbeth Color Checker (194 samples) and Agfa IT8.7/2 (289 samples) [31]. Then, a total of 4424 samples were used for training and testing the algorithms that were explained in Section

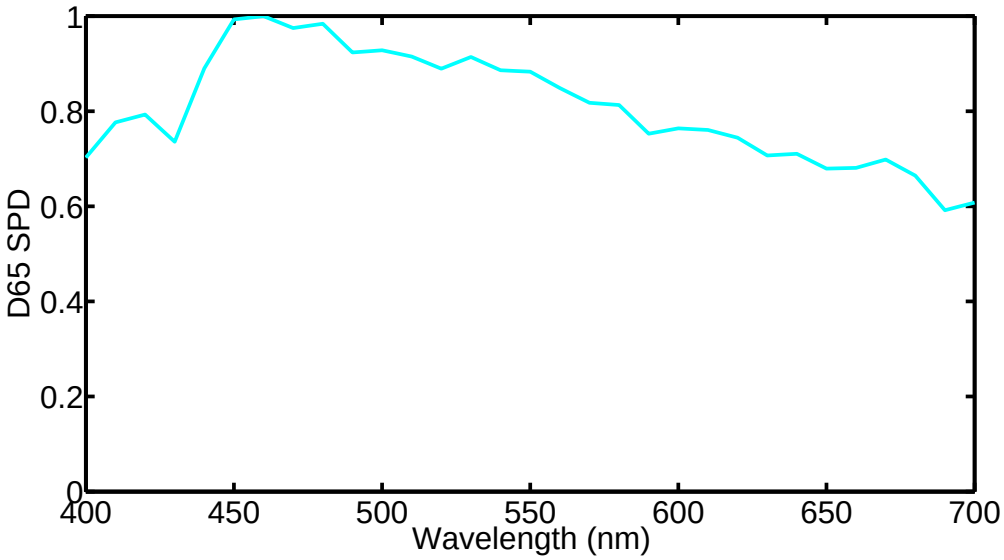

Figure 19: Relative Spectral Power Distribution of standard D65 illuminant

2.2.

Figure 20: Color charts used for the color calibration application

These color charts cover the four quadrants of the L*a*b* color space, offering a good general-purpose database for any real imaging application. In Figure 21 we can see both CIExy diagram and the CIEa*b* diagram with all the samples plotted on it under standard D65 illuminant (see Figure 19).

2.3.3 Spectral reflectance estimation

For this application, we have not only studied the general-purpose utility, but we have intended to be more specific by including some spectral data belonging to natural image databases. For this, We have selected two of the aforementioned databases (Munsell and NCS), and included some data extracted from the very well known Nascimento and Foster's database as well [59] [60] [61]. This database is composed by several hyperspectral images representing natural urban and rural scenes. It has been already used by other authors for spectral imaging studies [10]. We took from it 3 images for urban scenes and other 3 different ones for rural scenes (see Figure 22). Merging together the three scenes from each group, we got $3,119,956$ reflectances for urban scenes and $3,177,003$ reflectances for rural scenes. Since these groups were huge and we wanted

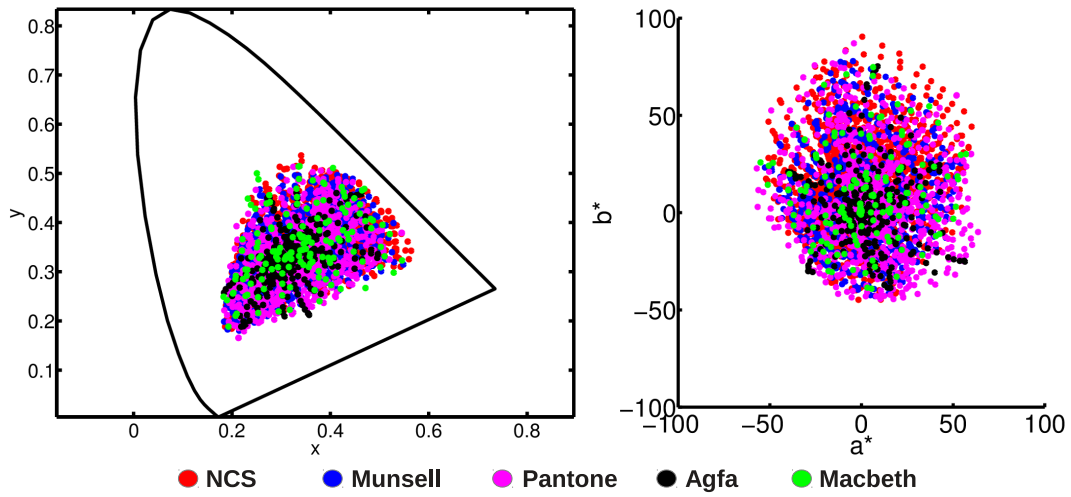

Figure 21: Color coordinates of the whole set of samples used. Left: CIExy color diagram. Right: CIEa*b* color diagram.

to compare results with Munsell and NCS sets, what we did was to clean and down sample these big sets down to 1200 samples each of them. The first cleaning step was to remove all the reflectances which were out of $[0.1 \ldots 1]$ range. Then, being sure that all the remaining ones were in the correct range of values, we did an evenly-spaced downsampling of the whole set. Taking reflectances all over the three scenes of each set. We also did the same subsampling in the RGB color images in order to be able to see how the resulting sets look like. The results can also be seen in Figure 22.

Figure 22: Top: urban (left) and rural (right) chosen scenes. Bottom: resulting sets after cleaning and downsampling the hyperspectral data

The differences between both sets are evident when we look at the RGB color pictures generated from them and also if we have a look at the xy and ab color diagram in Figure 23. It is also clear that there is some spatial arrangement within both sets. One can see how in both cases, the left-most samples look similar in color to the first scene, the central samples to the second scene, and the right-most samples to the third scene. To avoid any possible influence from this

fact when splitting the data into training and testing sets, the first thing which is done when starting the computations, is to scramble the sets randomly, so that any possible casually existing order or pattern in the spatial distribution of color is eliminated.

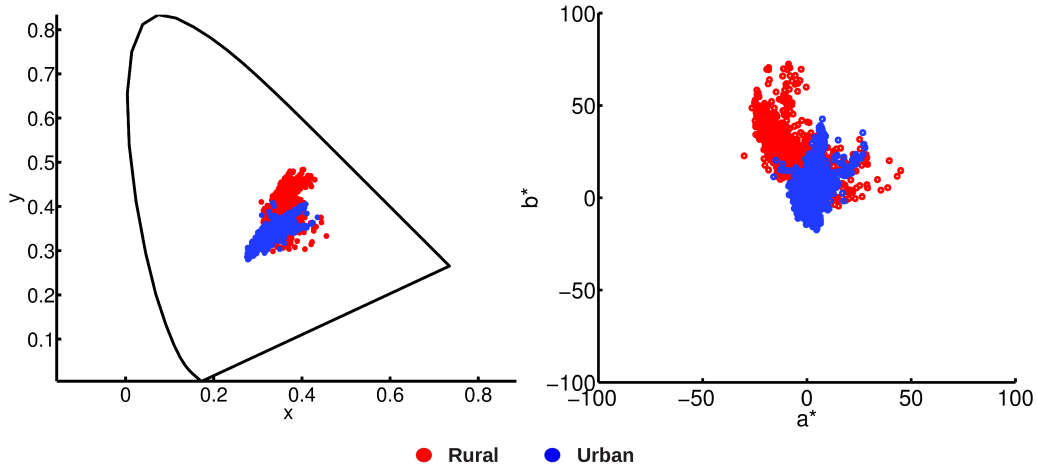

Figure 23: Color coordinates of urban and rural sets of samples used. Left: CIExy color diagram. Right: CIEa*b* color diagram.

2.4 Data preprocessing

The regression methods usually need some kind of preprocessing for the input data they are fed with. This preprocessing is normally either normalization, standardization or scaling of these data. If we want to compare the performance of the algorithms for a given application in fairly the same conditions for all of them, we need to study previously which of the algorithms benefit more from preprocessing or either even requires it necessarily. We need also to keep in mind that different algorithms might perform in optimal conditions for different preprocessing transformations.

2.4.1 Color calibration

In this application, all the models used are empirical models. Besides the parameters each models uses, the inputs for them are the reflectances and sensors' responses for the training set (R_{train} and ρ_{train} respectively), and the sensors' responses for the testing set (ρ_{test}). The reflectances used, as explained in Subsection 2.3.2, are already within the range $[0, 1]$ of values. The sensor responses are calculated in digital counts. We are assuming 12 bits for quantization, so the responses are within the range $[0, 4095]$. Several different standardizations, normalizations and scalings have been tested to see which one performs best for each algorithm. We have tested eight different preprocessing transformations, denoted with the letters A to H. Some of the transformations involve a standardizing procedure which basically shifts the mean of the data to zero and scales its standard deviation (std) to one, either calculating the mean and std for the whole set of responses, or doing it separately for training and testing sets. Other methods involve just a

normalization procedure which scales the data within the $[0, 1]$ range also for whole set or separating into training and testing, etc. The chosen one among the eight approaches, which was performing best for the color calibration application, was approach H (see Figure 24).

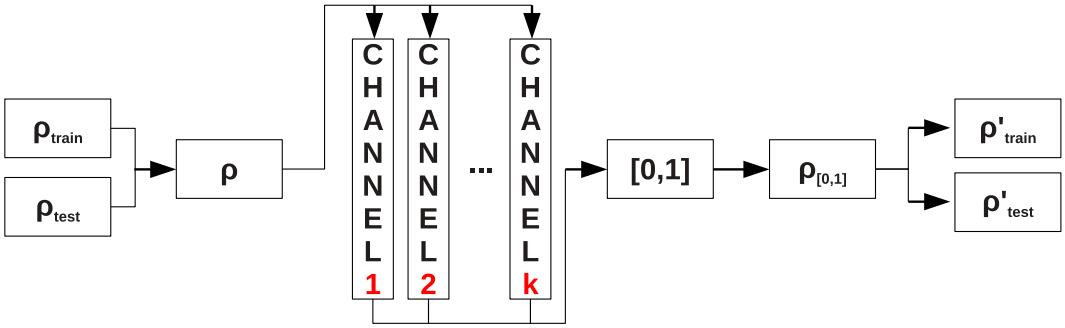

Figure 24: Normalization done in color calibration application

As the figure shows, the training and testing sets' responses are merged together (in other approaches they are kept separately) and separated by channels (not as in other approaches). For each channel a normalization is applied into the range $[0, 1]$ (or shift mean to 0 and std to 1 for other approaches) and they are split back into training and testing sets (not needed if the approach didn't merge them together in the first step). The idea behind this normalization was the fact that the different responsivities of the different channels may have big variations in value, or in spectral broadness. Therefore the responses would be in different scales depending on the channel and this could render the mapping task more difficult for the algorithms.

2.4.2 Spectral reflectance estimation

For this application we are comparing two empirical models (Kernels and Pseudoinverse) with one physical model (Wiener). The Wiener method is conditioning the preprocessing approach in this case, since it requires that the spectral responsivities are normalized to unit norm, as explained in Section 2.1.2. This normalization will be used afterwards for the three algorithms studied. After this normalization there is no need for further data preprocessing. Therefore the algorithms can be fed with the sensors' responses and all of them will be studied under the same conditions so their performances can be compared.

2.5 Error metrics

In this section we will explain the error metrics used in our study. Subsection 2.5.1 present the metrics calculated for the color calibration application, and Subsection 2.5.2 the metrics used for the spectral reflectance estimation application.

2.5.1 Color calibration application

For the color calibration application, since no spectral reflectances were recovered but only color coordinates directly mapped from sensors' responses, the metrics stored and used to compare the performance of the algorithms were always color difference metrics. There have been many proposals for computing color differences related to human visual perception, or Color Difference

Formulae (*CDFs*), (see for instance [62] or [63]). Among them, we have decided to use the ones listed below for the reasons which are also stated briefly in the next paragraphs.

ΔE_{00}*

This color metric is the last recommendation from the CIE. It was published in 2000. It includes correction computations and crossed terms of perceptual correlates of visual attributes, to overcome the problems detected for ΔE_{ab}*. For further information and formulae see [64][65].

2.5.2 Spectral reflectance estimation application

For the Spectral reflectance estimation application, the previously mentioned color metric was computed, together with some others which evaluate mainly the spectral similarity or are defined as combined colorimetric-spectral metrics.

Goodness of Fit Coefficient

GFC is a spectral metric which accounts for the differences in the shape of the spectral curves rather than the differences in their absolute values. Mathematically speaking, it is the cosine of the angle between two vectors, as it is computed as the ratio between the scalar product of the two and the product of their euclidean norms (see Equation 2.36). So, if the vectors (or reflectances) are identical or parallel, the value of GFC is 1 (perfect match). In the worst case the value of GFC is 0 if the two vectors are perpendicular. Optically speaking, two spectral identical curves with an offset in lightness, would still have a GFC equal to 1.

$$GFC = \frac{\sum_{\lambda=400}^{700}(R(\lambda)\cdot\hat{R}(\lambda))}{\sqrt{\sum_{\lambda=400}^{700}R(\lambda)^2}\cdot\sqrt{\sum_{\lambda=400}^{700}\hat{R}(\lambda)^2}} \tag{2.36}$$

Where R is the original spectral reflectance and $\hat{R}$ is the recovered one. Among the quality metrics analyzed, the GFC is the only one for which the similarity between the two samples compared (and so the quality of the estimation) increases with an increasing value of the metric. In order to render easier the analysis of the results, we have computed the complement of GFC or CGFC as $1 - GFC$. The CGFC is related directly to the error in the estimation as the *CDFs* and the other spectral or combined metrics analyzed.

For the sake of intuition, the complementary of GFC ($CGFC = 1 - GFC$) was also calculated, which, as the other spectral metrics, the lower it is the better.

Root Mean Squared Error

RMSE is also an spectral metric, and it is related to the sum over the whole wavelength range of the squared absolute differences between two spectral vectors. We have pointed out before that GFC is not sensitive to differences in scale between two samples, but RMSE clearly is. A perfect match would yield a RMSE equal to 0. RMSE is computed as

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(R(\lambda_i)-\hat{R}(\lambda_i))^2} \tag{2.37}$$

Where n is the number of wavelengths. This metric, together with GFC and ΔE^*_{ab} have been previously used in the context of spectral estimation [66][67][68].

ΔE_{ab}*

It is the euclidean distance between two data points in the tridimensional CIEL*a*b* color space. It was published in 1976. There are more recent recommendations from the CIE, but still in some fields (mainly for industrial quality inspection applications), it is widely used. For more details about the formulae see [69]. It is used in the computation of the combined metric $CSCMA$.

CSCMA

This is an adapted version of $CSCM$ metric. $CSCM$ is the acronym for Colorimetric and Spectral Combined Metric. It was proposed in [70][40].As its name says, it combines both spectral metrics and color difference metrics to take into account the importance of controlling both of them especially in the context of optimization processes where only one cost function is used. It is calculated as

$$CSCM = \ln(1 + (1 - GFC)) + \Delta E_{ab} * + IIE(\%) \tag{2.38}$$

Here $IIE(\%)$ stands for Integrated Irradiance Error [71], which is basically the difference of area below two curves (difference in integrals). $CSCM$ was proposed to measure differences between the SPDs of light sources, and that is why it is formulated using $IIE(\%)$. For the sake of spectral reflectance estimation, we used $CSCMA$, which instead of using the $IIE(\%)$, uses the $RMSE$. Besides, there are three free parameters (α β and γ) to control whether we give more importance (or weight) to GFC, ΔE_{ab}* or $RMSE$. Its formula is

$$CSCMA = \ln(1 + \alpha \cdot (1 - GFC)) + \beta \cdot \Delta E_{ab} * + \gamma \cdot RMSE \tag{2.39}$$

And the optimal values found for the parameters are: $\alpha = 1000$,$\beta = 1$ and $\gamma = 100$ (see [19]).

2.6 Training and evaluation of models

We can consider the algorithms studied in this work as regression methods. We have a set of data, and we divide it into two parts. One for training the algorithm and one for testing it. It is important to do so in such a way that the selection of these samples would not bias the final results of evaluation. We will divide again the section into two subsections, one for each application. We will explain the machine learning strategies used for each algorithms considered.

2.6.1 Color calibration

The machine learning designs used for ANN and for Polynomial Least Squares Fitting were the same. In both cases we have two free parameters to optimize. In the FFBPNN we have the number of hidden layers and the number of neurons in each hidden layer. For the RBFNN we have the number of neurons in the unique hidden layer, and the spread of the Gaussian function (see Section 2.2.1). For the polynomials we have the regularization parameter λ, and the degree of the polynomials d (see Section 2.2.1). So the first thing we did was to construct a bi-dimensional grid with a range of values for these parameters. In one dimension of the grid we have a number of values for one of the parameters, and in the other dimension for the other. So for example if we have 10 values for one parameter and 10 values for the other, we will have a total number of 100 different combinations of parameters. We will study one by one each of these combina-

tions and determine the optimal combination of parameters by an exhaustive search procedure. For each of the combinations, we take the set of reflectances and scramble them randomly 10 times. The scrambling process was random, but controlled in such a way that always the same 10 randomizations were used for every combination of parameter values. Then we divide the scrambled set into training and testing sets (75% for the first and 25% for the second), and then train the algorithm with the training set and evaluate it with the testing set. When all the ten randomizations for one combination of parameters are done, the error metrics are averaged, and the process starts again for the next combination of parameters. So, in the end, for each combination of parameter values we get its averaged error metrics, and we can choose the combination giving the best performance. A small scheme of this machine learning design is represented in Figure 25.

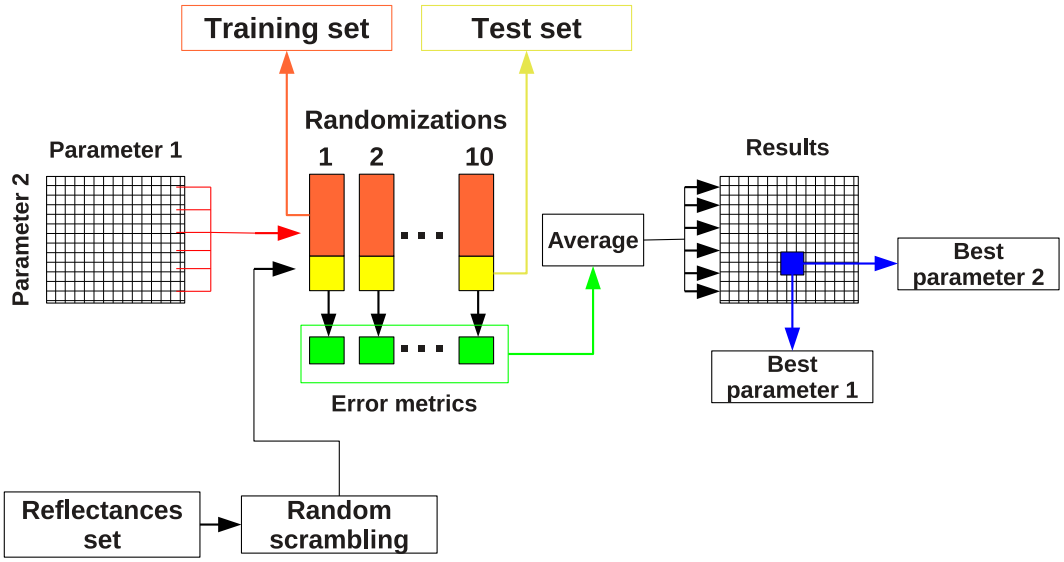

Figure 25: Machine learning design for ANN and Polynomials in color calibration application

Exhaustive search was used rather than any other more intelligent optimization algorithm. This was done because the error surface for the grid of parameters is full of local minima. Due to this fact, finding a global minimum is a very complex task which needs quite complex and computationally demanding algorithms to be done. We left this task for future work. For Kernels the approach is similar but not identical as we can see in Figure 26.

First it scrambles the data 10 times in the same way as before, and then for each randomization we also divide it into training (orange) and testing (green) sets. We take the training set and optimize the parameters using a k-folds method. With these best parameters found in each randomization, we evaluate using the testing set and then average the error metrics to get the final results.

The k-folds method, consists in dividing the data into k parts of the same size. Then we do k iterations taking in each of them one of the parts as testing set and the rest as training set. A popular value for k is 10. In this study we tried all values of k going from 2 to 20, and no signifi-

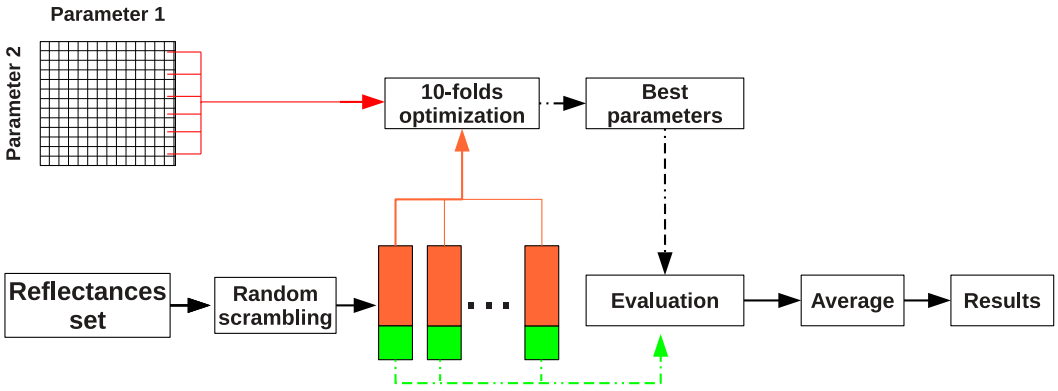

Figure 26: Machine learning design for Kernels in color calibration application

cant differences were found, so we just set the value to 10 finally. Inside the k-folds optimization, for each iteration all the grid of parameters is tested, and finally the combination of parameters with lower average error in all iterations in chosen.

2.6.2 Spectral reflectance estimation application

For this application, the same machine learning strategy was used for all the algorithms tested. Briefly explained this strategy consist in a double k-folds methods. K-folds for optimization and a separate k-folds for evaluation. An scheme of this approach is shown in Figure 27.

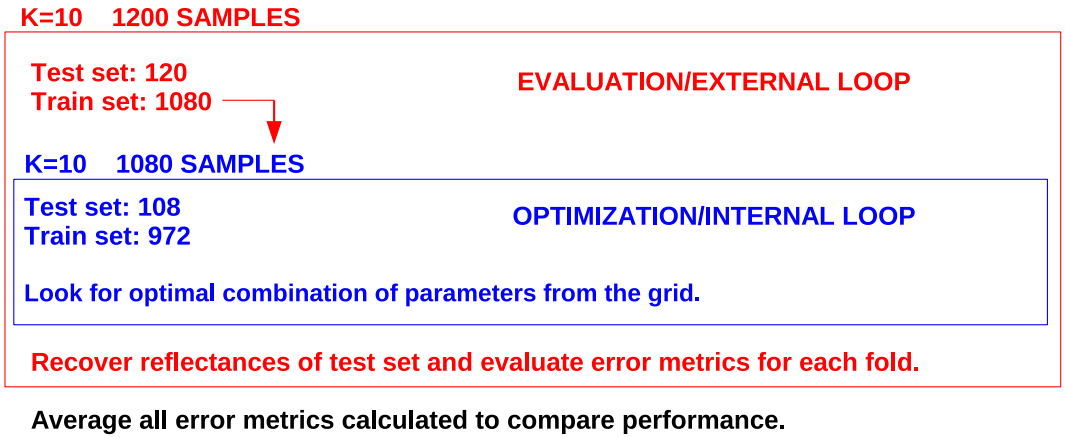

Figure 27: Machine learning design for all algorithms in spectral reflectance estimation application

We can see how all the 1200 samples are divided into $k = 10$ parts (120 samples each part), and 1 part is taken as testing set (in each of the k iterations) and the remaining 9 parts (1080 samples) for training set. Then these training set is again divided into 10 parts, and the optimization is done using 1 part (108 samples) as testing and remaining 9 parts (972 samples) as training. This inner k-folds loop (in blue in the figure), looks for the optimum parameters for Wiener and Kernels. Pseudoinverse does not have any parameters to optimize so this inner loop

is not present. For Wiener there is only one parameter to be optimized (regularization parameter, see Section 2.2.2), and for Kernels two (see Section 2.2.1).

3 Results and discussion

In this chapter, we will summarize the most relevant results obtained in our study. For the sake of clarity, we will present a selection of representative data in the different subsections, and leave out most of the bulk of computational results generated; some of them will be separately provided in the Appendix 5. The results in each section are divided in experiments to show the different aspects of the behavior of the system that we analyzed.

Sections 3.1 and 3.2 are dedicated to the two main applications studied. Finally, in section 3.3 we will show and discuss comparatively direct versus indirect color calibration.

3.1 Results for the color calibration application

This section is divided in five experiments which analyze different relevant outcomes of this first application. In subsection 3.1.1 we present the results of a series of tests aiming to decide which is the best combination of sensors for the single and double shot approaches. This selection was our first task, since we had to discount 21 or 18 sensors out of the 24 available for our simulated TFD system. Later on, in section 3.1.2, we show the results for the experiment we did to study which is the optimal size of the training set to train and test all the different models under the same conditions. In this way the results we obtain from all of them are comparable. Then, after we knew the optimum size of the trainng set, we first optimized the models for the simplest case, which is the single shot unfiltered TFD, to see the capabilities and limitations of the TFD before studying the ways of improving them. The results for this experiment are shown in Subsection 3.1.3. After that, the results of the experiment done for the evaluation of the performance of the rest of approaches are shown in Subsection 3.1.4. Finally, in Subsection 3.1.5 we show the results obtained during the optimization of Polynomials and Neural Networks to control the overfitting of the solutions and ensure the models have good generalization properties.

3.1.1 Experiment I: Sensor set selection

In this application we have used single and double shot approaches. The developers of the TFD gave us eight different sets of responsivities for the TFD, and we will only need one or two of them for the single shot and double shot set-ups respectively. This means that we have to select which two sets to use in our simulations. To carry out this selection process, we fixed the size of the training set to 50% of the total set of samples, and optimized coarsely the Kernel models and Neural Networks. The results found are shown in Figure 28.

We can see in this figure the sum of the ΔE^*_{00} errors for all the four models studied for each of the TFD sets,and the contribution of each model to this global error. We have included the Retiga sensors here to have an idea of the performance of sensors with narrower responsivities (although without full spatial resolution). We can see how out of the eight different TFD sets, the number two is the one giving best results, both globally and for each of the algorithms

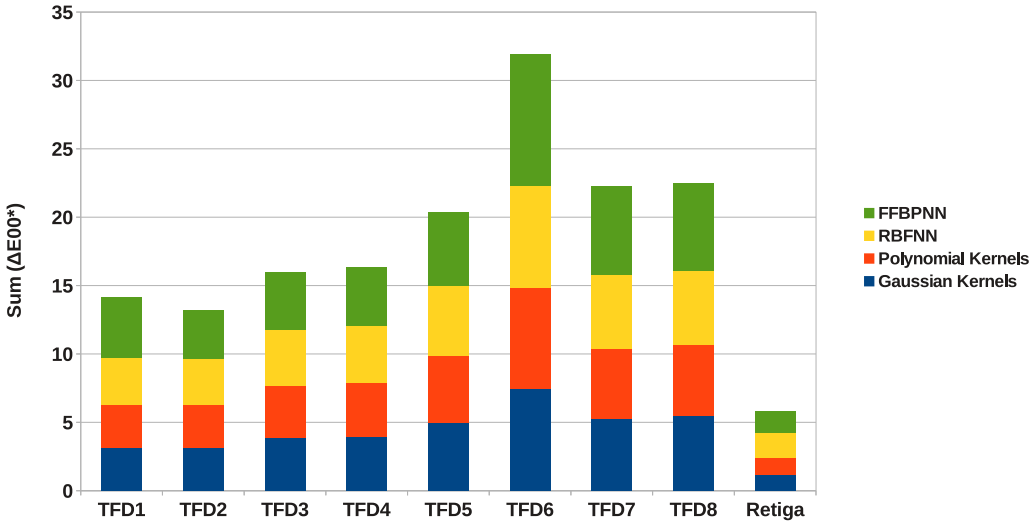

Figure 28: Performance of the different sensor sets

individually as well. For the single shot approaches (with and without filter) we will use this set. The second best performing set is the number one. Thus, for the double shot approaches we will use the sets named TFD1 and TFD2. The fact that these two sets are the best separately, does not mean that they are the best combination of two sets taken together. There are 28 possible combinations of two different sets, and calculating all of them would be a very long process even for coarse optimizations of the models. Therefore we did not try all the possible combinations. This task is pendant for future work. Once the sets of responsivities were chosen, we proceeded with the next experiments.

3.1.2 Experiment II: Influence of the size of the training set

As a preliminary step before starting with the fine optimization of the models, we studied the influence that the size of the data sets used for training the algorithms (training set) can have on the performance of the different models. We wanted to analyze and select which was the optimal size for the training set for all the algorithms, because we wanted to compare all the models under the same conditions (including the same size of training and test sets). We expected that the different algorithms would behave similarly (under some tolerance) once the stability region for the training set size was reached. For this, we did a first coarse optimization of Kernels and Neural Networks using different sizes of the training set (5%, 20%, 50%, 66% and 75% of total data set) with TFD2 sensor set. The results found for ΔE_{00} are shown in Figure 29.

As we can see, for Kernels and RBFNN (equivalent to Gaussian Kernels), for smaller sizes of training set, the performance does not have a significant decrease when compared with higher sizes. However, for FFBP, smaller sizes of training set do not give very good results. If we increase the size of the training set, we see how the performances of all models stabilize and get closer.

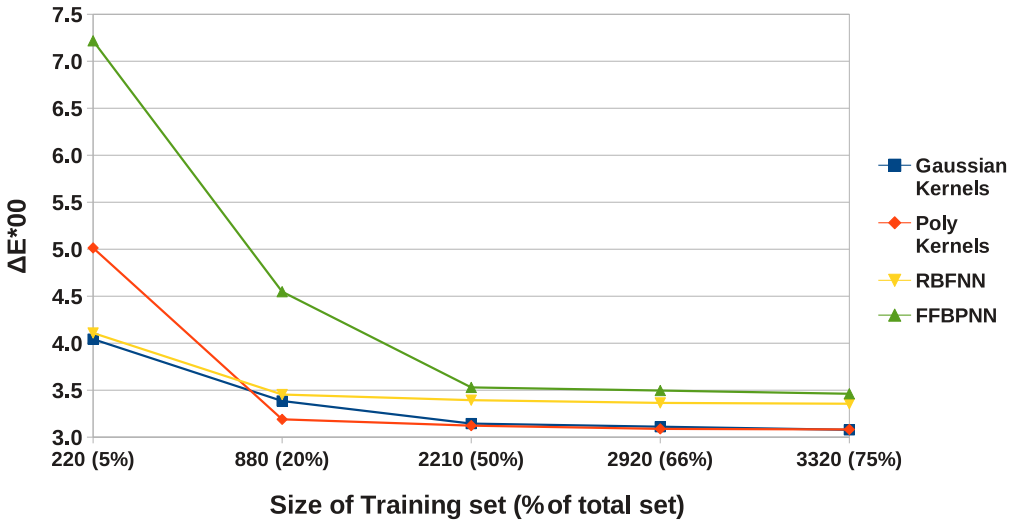

Figure 29: Influence of the size of the training set for Kernels and ANN

If the training set is very small, we can have problems of generalization, due to the fact that the set does not represent properly the sets which we are going to test the algorithms with. If this happens, the performance would be bad because the mapping or hypothesis found in the regressions is not valid for the samples we are testing with. If, on the other hand, the training set is too big, we can have saturation. It means that no matter how many more samples we add to the training set, the performance is not improving anymore. The undesirable effect of this is that the bigger the training set is, the longer the time it takes for the algorithms to be trained. We chose as final size for the training set 66% for two main reasons: the first is that it is big enough to ensure that none of the models have any problems of generalization; the second is that the optimization times are reasonable still for this size.
Even if the optimization performed in this experiment was somewhat coarse, we can still attempt to establish a preliminary ranking between the models: the two Kernel-based algorithms perform best, followed by RBFNN and FFBPNN in last position of ranking.

3.1.3 Experiment III: Color calibration using one shot of TFD sensors

In this experiment we were using five different algorithms to find the map from three sensor responses got in one single shot with the TFD and the XYZ tristimulus values. Once the size of the training set was set to 2920 samples (66% of the global data size, according to the results of the previous experiment), we optimized five different models (the four shown in Figure 29 plus Polynomials). As we explained in the previous chapter, with the optimization procedure we aim in finding the values of the parameters for each model resulting in the best performance. The optimized parameters were: regularization term for Kernels and Polynomials, spread of Gaussian function for Gaussian Kernels and RBFNN, polynomial degree for Polynomial Kernels and Poly-

nomials, number of neurons per hidden layer for RBFNN and FFBPNN, and number of hidden layers for FFBPNN. This is a total of two parameters for each model, so out of a set of values given to each parameter covering a wide range, the best combination of two parameters' values was found by exhaustive search, and used to obtain the best performance for each model (see Table 1 for the $DeltaE_{00}$ values obtained).

Table 1: ΔE^*_{00} results for single shot with TFD, a training set size of 2920 samples (66%) and average SNR of 42.78dB

Algorithms	ΔE^*_{00}
Gaussian Kernels	3.09
Polynomials Kernels	3.08
RBFNN	3.08
FFBPNN	3.31
Polynomials	3.08

The performance of the Kernels is almost the same as that of RBFNN and Polynomials. The best results are those of polynomial Kernels, but the difference between models is very small. It is very interesting to observe how, using Kernel models, we only need to change a small piece of the implementation, which is where we select the type of Kernel to be used, and with this easy action, we are jumping between totally different models like Polynomial regularized least squares fitting and one-hidden-layer RBFNN. This is one of the important advantages of using Kernels.
We also see how, for FFBPNN the error increases roughly in 10% respect to the results obtained by the rest of the models. For the next experiments, we did not continue optimizing the polynomials, because it was a long process, and as we have explained, the model is similar to polynomial Kernels (as we saw in the results), so there is no need to continue using them anymore.

3.1.4 Experiment IV: Results for the rest of proposed system configurations

After optimizing all the models for the single shot set-up with the TFD, and deciding to drop the polynomials, we proceeded optimizing the remaining four models for the rest of set-ups: single shot filtered, double-shot unfiltered, double-shot filtered and Retiga. The results are shown in Figure 30. The numerical data corresponding to this figure is shown in Table 11 in the Appendix 5.

Regarding the models, we see how for every different set-up, all of them have practically the same performance, but FFBPNN which is slightly worse in all cases. This is due to the optimization. In FFBPNN we are optimizing the number of hidden layers of the network and the number of neurons per hidden/layer, as we will see in Subsection 3.1.5. For double shot and Retiga approaches, some local minima were found in the error surface and the combination of parameters which was globally best was not chosen. For the rest of the approaches this does not happen. It is not an important issue because FFBPNN are performing the worst in any case and this local minima effect does not alter the ranking of the models for the different approaches.

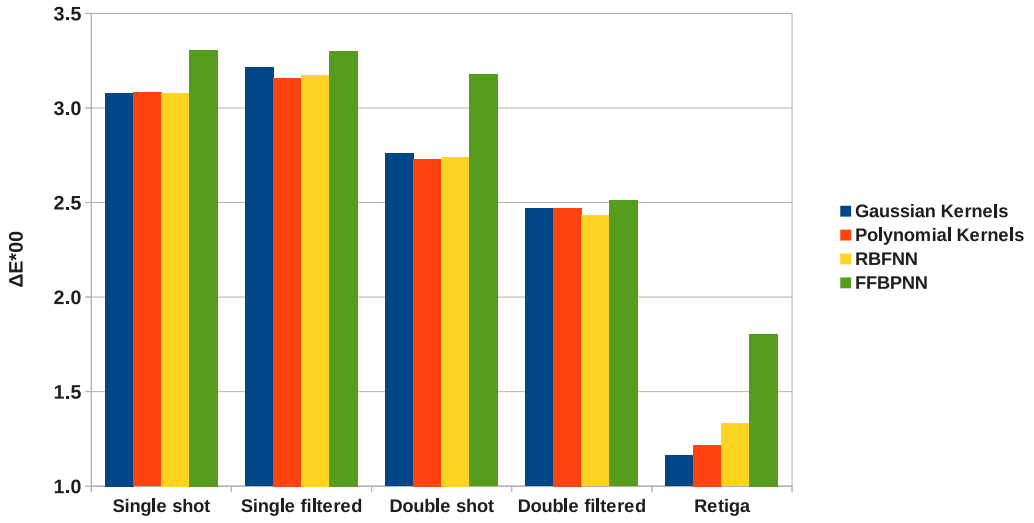

Figure 30: ΔE^*_{00} for all set-ups studied in the color calibration application.

In Table 2 we can see the average ΔE^*_{00} error for all the algorithms in each of the different set ups. We included also the single shot unfiltered approach seen in previous section for reference. Here we can have an idea of the improvement of adding more shots and the filter disregarding the model used for the regression.

Table 2: Average ΔE^*_{00} results for each set up used in color calibration

Set up	Average ΔE^*_{00}
Single shot unfiltered	3.14
Single shot filtered	3.21
Double shot unfiltered	2.85
Double shot filtered	2.47
Retiga	1.38

We can observe how adding a second shot, the accuracy improves roughly in average $0.3\Delta E^*_{00}$ units in the unfiltered case. Besides, when we only have one shot, and we add the filter, the results are not better (they are slightly worse). However, if we increase the number of channels to six, only adding one more shot, the performance improves around $0.4\ \Delta E^*_{00}$. As we will see in Subsection 3.2.1, for higher number of channels the filter also improves the results for the color difference errors. Finally, if we compare the performance of the Retiga sensor (only one shot), with that of the TFD, we see how results are almost two ΔE^*_{00} units better in average for the single shot unfiltered TFD. If we consider applications in which we can only afford to take few shots (one or two), then we can have a higher accuracy using a RGB system, but we will not

have full spatial resolution. If we want full spatial resolution then the color accuracy will drop off around 1.1 ΔE^*_{00} units in average.
We have improved the performance of the TFD for the color correction application (direct mapping) on an average of 0.7 ΔE^*_{00} units. In general, we see that even for the Retiga sensor, we do not get any color difference below 1 ΔE^*_{00}. Regarding the tolerance of the ΔE^*_{00} color difference formula, there is not an established threshold yet. One unit is taken as a rule of thumb, to asses whether a measurement is good or bad, but of course, it would be nonsense to say that a slightly lower value is good and a slightly higher value is bad. Anyhow, we are conscious that TFD results are here still far from accurate colorimetrically speaking.

3.1.5 Experiment V: optimization of parameters

In this subsection we will see how the performance of polynomials and neural networks vary for different values of their parameters.
We saw in the previous chapter that polynomial models have two free parameters, and one of them is the degree of the polynomial (see Subsection 2.2.1). This free parameter needs to be optimized together with the regularization parameter, in order to get the best performance. One could think that the higher the polynomial degree is the better, because the curve will fit more the data points in the training set. However this is not true, due to the fact that a very high degree could make the hypothesis fit so much the data points in the training set (cause overfitting). If this happens, the model looses generalization properties and then when we evaluate for the test set (whose points are different from those of the training set), the performance is bad. To illustrate this overfitting effect we show in Figure 31 the colorimetric error as as function of the polynomial degree for the single-shot unfiltered case. For the computation, we have set the regularization parameter to 1 $times10^{-12}$. The same behavior is found for other regularization constant values and system's set-ups.

We can see in the figure how for this value of regularization parameter, the best performance was found for the degree 7. As we increase the value of the degree from 2 on, we get better results, but if we choose more than 7, then the overfitting is present, and we see how the results are slightly worse.

For the case of RBFNN, the parameters to be optimized were the number of neurons in the unique hidden layer and the spread of the Gaussian function. In Figure 32, we show the colorimetric error obtained for different values of these two parameters.

We see how after a certain number of neurons (around 90) in the hidden layer, the performance stabilizes and there is no further improvement when more neurons are added. Besides, we can observe how for smaller number of neurons, the influence of the spread of the Gaussian function is high, but as we increase the number of neurons, there is less and less difference between low and high values of the spread. High values of spread make the neural network converge faster, without the need of adding extra neurons and smaller values of spread need more neurons to be added to make the network converge.

Finally, for FFBPNN, the parameters to be optimized were the number of hidden layers and

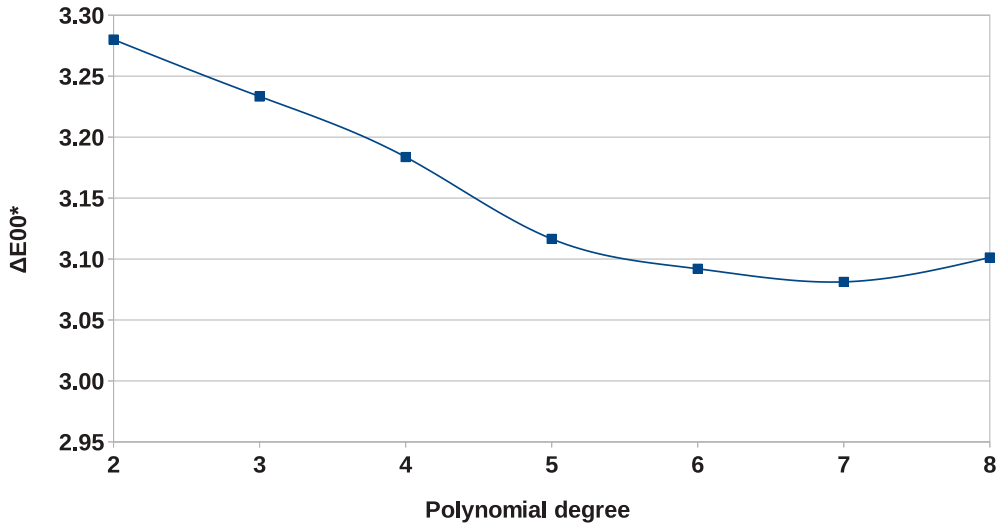

Figure 31: ΔE^*_{00} versus polynomial degree for single shot color calibration

the number of neurons per hidden layer. We can see in Figure 33 how the combination of both parameters influence the performance of the model. In general, only one or two hidden layers for the network are enough and if we increase the number of layers, the performance is worse.

The different colored lines correspond to different numbers of neurons per hidden layer. We see how for one hidden layer, the performance is almost the same for every number of neurons used, but as we increase the number of layers, the influence of the number of neurons increases. If we have a look at the results for five hidden layers, despite the case of 8 neurons per layer, for the rest we can see how the more neurons per layer the better. However, if we add the eighth neuron to each hidden layer, the performance is not better anymore, pointing out that to many neurons per layer is not beneficial.

3.2 Results for spectral reflectance estimation application

In this section we show the results corresponding to three different experiments done to assess the influence over the spectral reflectance estimation of: the introduction of the cut-off filter 3.2.1, the data set used 3.2.2 and the models studied 3.2.3. As in the color calibration application, where we saw how the results improved when including more shots and the filter, here we wanted to know how better the results can be including the filter, but now not only from the colorimetric point of view, but also from the spectral point of view, since now we are recovering spectra and we can use also spectral metrics to analyze the performance of the estimations. Besides, even in the color calibration application we did not study the influence of the data set used because we had only one big data set. In this case, we have four different sets. Two of them (rural and urban), are extracted from a database of spectral images. These images were captured with LCTF plus monochrome sensor. We thought that this could be a reason for them to be noisier

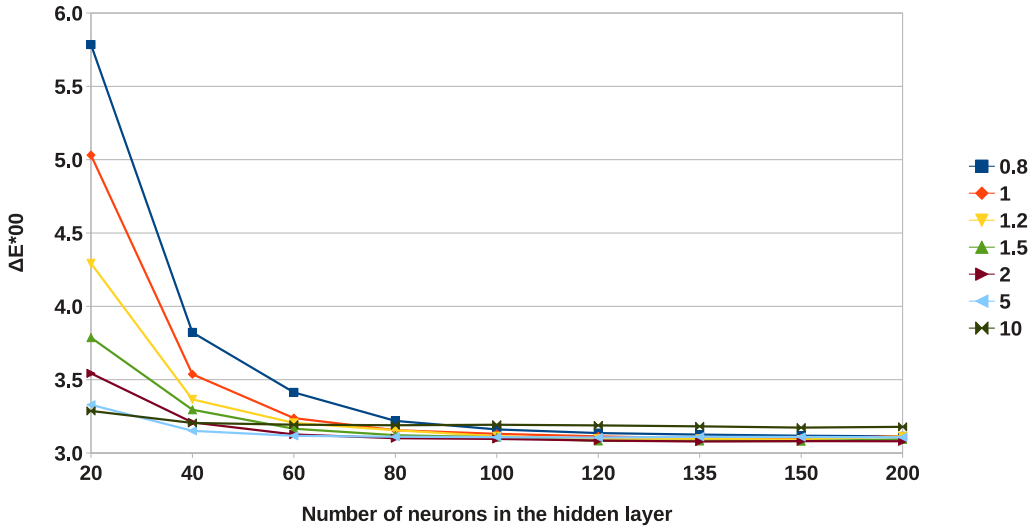

Figure 32: ΔE^*_{00} versus number of neurons for RBFNN

or have more artifacts. Thus they could give worse results than the other two sets. In addition, the comparison of models was done again, because for this new application we have introduced two new models that were not used in the color calibration application. They are Wiener and Pseudoinverse, and we wanted to study how better or worse Kernels can be compared with these two widely used methods.

3.2.1 Experiment VI: Influence of filtering

As we explained in previous chapters, for the spectral reflectance application two approaches were studied: eight shots and eight shots with filter. This was done because some references point out that better colorimetric accuracy can be reached sharpening the responsivities of the sensors [34][36][38].

In Table 3 we present the color and spectral estimation quality metrics for TFD, with and without the filter for the Munsell set. The same behavior was found for the rest of data sets (see Tables 12 - 14 in the Appendix).

For all the models, the colorimetric quality improves when we include the optical filter. However, the spectral metrics are slightly worse with the filter than without it. In Table 4, we can find the average of all models for the unfiltered and filtered cases and so the global effect of adding the filter across all the computations performed for these set-ups.

The filter is helping colorimetrically but not spectrally. We have a large colorimetric improvement (average of $1.57\Delta E^*_{00}$ units) and a spectral worsening of 0.002 for GFC and 0.005 for RMSE in average. To explain this result we need to have a look at Figure 34, where the quantum efficiencies of one of the shots of the TFD, weighted by the illuminant, are compared to the Color Matching Functions (CMFs) used to calculate the color coordinates of the samples, for both fil-

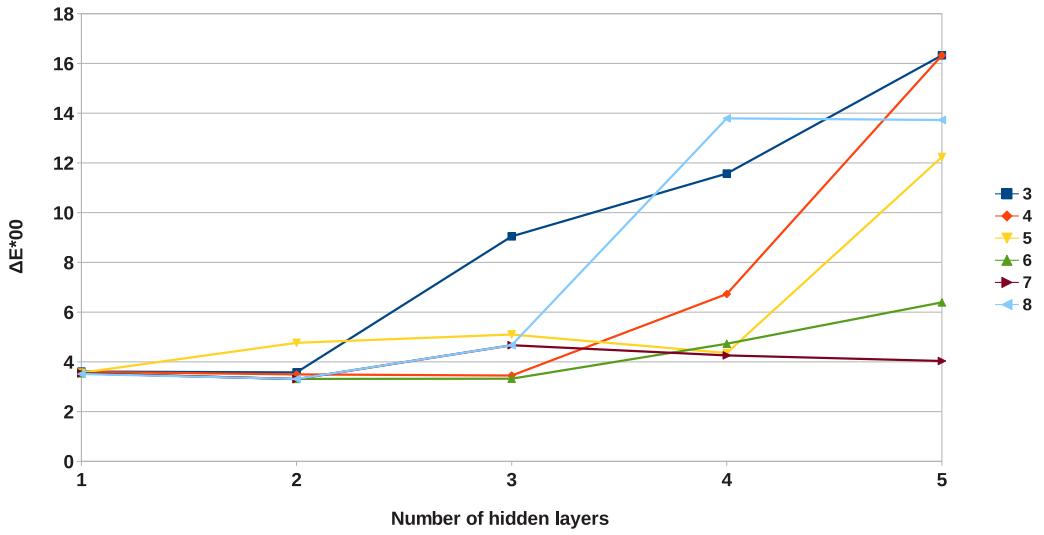

Figure 33: ΔE^*_{00} versus number of hidden layers for all numbers of neurons for FFBPNN

Table 3: ΔE^*_{00}, GFC and RMSE results comparing unfiltered and filtered approaches for 24 channels and Munsell data set

	ΔE^*_{00}		GFC		RMSE	
Model	**Unfiltered**	**Filtered**	**Unfiltered**	**Filtered**	**Unfiltered**	**Filtered**
Gaussian Kernels	1.39	0.97	0.999	0.997	0.010	0.012
Polynomial Kernels	1.63	0.97	0.998	0.997	0.012	0.015
Wiener	2.46	1.58	0.995	0.993	0.020	0.023
Pseudoinverse	2.51	1.13	0.995	0.994	0.018	0.021

tered and unfiltered case.

If we look at the unfiltered quantum efficiencies (left), we see how, in the extremes of the visible spectrum, the blue and red channels have relatively high sensitivity. This will help the models to estimate the reflectance spectra in these areas. However, without the filter, the quantum efficiencies of each of the channels are broader, and they have higher level of overlapping. We have quantified this calculating the area of overlapping between curves for both filtered and unfiltered cases. We can see the results in Table 5 for both cases and measuring the overlap of curves two by two.

This overlapping area was computed finding the wavelength where the two curves crossed, and integrating, along each side of this wavelength, the area of the curve which was lower than the other (and so contained in the area of the other curve). We see how for each pair of channels, the overlapping is lower when the filter is included. A higher overlapping makes more difficult

Table 4: Average results for all models and data sets comparing filtered and unfiltered cases

Approach	Average ΔE^*_{00}	Average GFC	Average RMSE
Filtered	1.16	0.993	0.031
Unfiltered	2.73	0.995	0.026

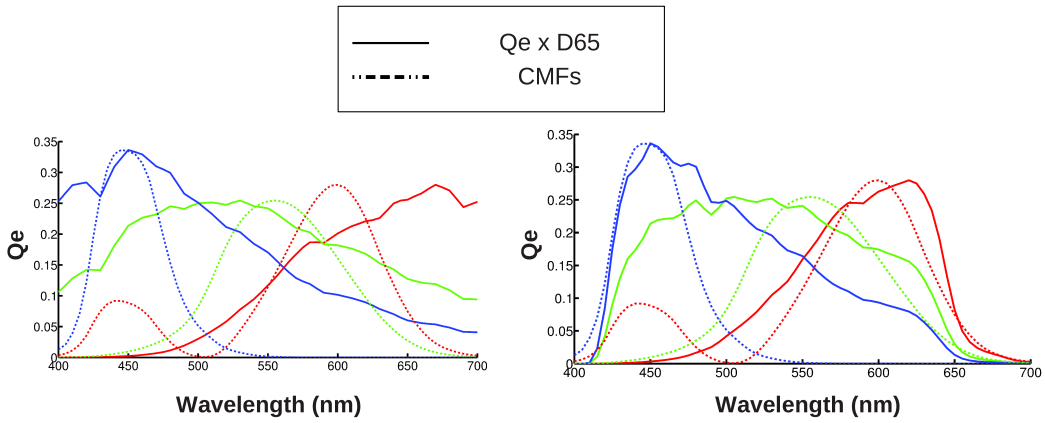

Figure 34: Quantum efficiencies compared with CMFs for unfiltered (left) and filtered (right) cases

to separate the information from each part of the spectrum in the colorimetric sensitive area of it (where the color matching functions have higher values). On the other hand, if we look at the filtered quantum efficiencies in Figure 34, we see how they do not have sensitivity in the extremes of the spectrum which is bad for recovering these areas of the spectral reflectances, but in the central area, each channel is narrower and more similar to the CMFs, and there is less overlapping between channels. This fact makes the recovery of this part of the spectrum (the one which is colorimetrically relevant), easier for the algorithms.

To analyze the areas of the spectrum where the recovery was more problematic, we calculated the average of the absolute error between original and recovered reflectance spectra for each wavelength in both the filtered and unfiltered cases. In Figure 35, we have the relative error for both cases depending on the wavelength found for polynomial Kernels (which was the model giving best results), and Munsell, but a similar behavior was found for the rest of models and data sets.

The errors in each case (filtered and unfiltered) are normalized to maximum one. We can observe how, for the unfiltered case, the error has higher values in the central part of the spectrum. On the other hand, for the filtered case, the relative error in the central part (colorimetrically relevant) of the spectrum is much lower than that in both extremes of it. This analysis supports our interpretation of the effect of filtering in our results.

To go one step further in the analysis of this interesting behavior of the system when we apply the filter, we implemented the evaluation method proposed by Vora and Trussel in [42]. This

Table 5: Area of overlapping between curves, two by two, for filtered and unfiltered cases

Unfiltered	R, G	6.1596
	G, B	11.5444
	R, B	4.6931
Filtered	R, G	5.1948
	G, B	9.7902
	R, B	4.2904

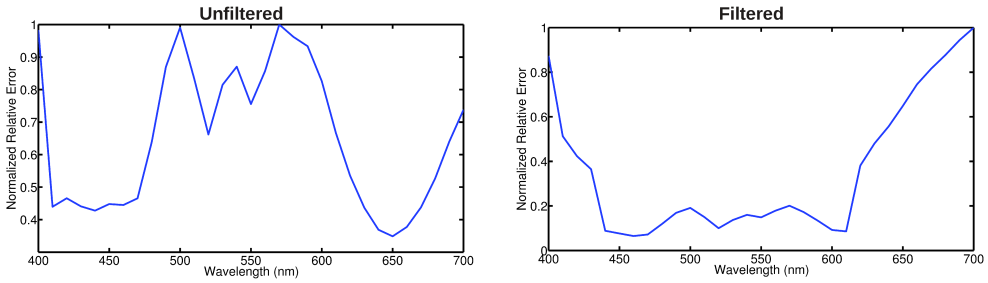

Figure 35: Relative error for unfiltered and filtered cases for each wavelength

is a measure of goodness, representing how well a set of filters (or responsivities weighted by illuminant and filter in our case) is able to recover the same reflectances that the set of Color Matching Functions. It is a relative error between the spaces spanned by both sets. The smaller the error is, the more similar the group of colors that both sets can reproduce is. In Table 6, we see the values of this error for TFD1 and TFD2 in the filtered and unfiltered cases.

Table 6: Measure of goodness proposed by Vora and Trussel for the filtered and unfiltered cases

Approach	TFD1	TDF2
Unfiltered	0.22211	0.21439
Filtered	0.03154	0.02995

We have found the same trend that is shown in Table 6 for Vora and Trussel's measurement in all sets of TFDs. The error found for the filtered sensors is lower than the error found without adding the filter. This reflects the fact that for the system, it is easier to recover the tristimulus values when the filter is used than when it is not.
We complete the analysis by showing in Figure 36 two instances of recovered reflectances.

The black continuous lines represent the original reflectances and the green and red dotted curves represent the recovered reflectances. We see how in both cases, the original reflectances presented have tails in the extremes of the spectrum (maybe due to some problems inherent to the capture procedure for the hyperspectral images database). These tails are present in the red extreme for the sample on top of the figure, and in the blue extreme for the sample presented on

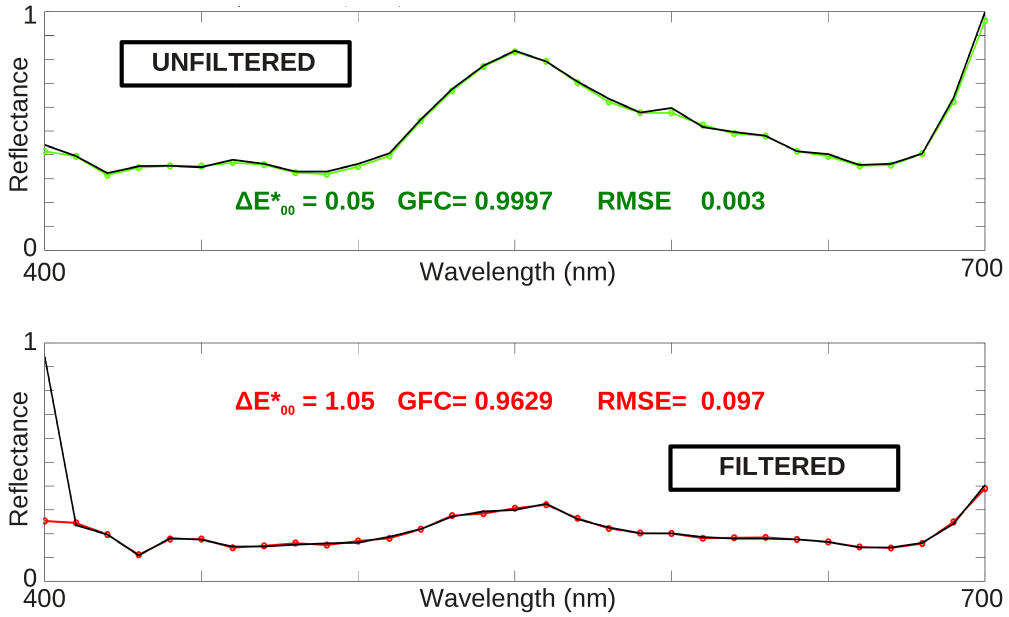

Figure 36: Examples of good and bad recoveries comparing unfiltered (top) and filtered (bottom) approaches

the bottom of the figure. Also the color and spectral metrics results are shown for each recovery. We can see how for the unfiltered case, the recovery is quite good even in the red extreme of the spectrum when the tail is present. On the other hand, for the filtered case, we see how the tail present in the blue extreme of the spectrum is not well recovered. Even though the rest of the spectrum is recovered good, and the color metric gives us a reasonable accurate value, the spectral metrics drop off giving bad values, specially for the GFC. Naturally, GFC would be more sensitive to this issue, because there is a rather noticeable change in the shape of the curve in the recovered reflectance when compared to the original. For the RMSE, most of the wavelengths are recovered very satisfactorily, and only one or two of them have high absolute error. Thus, RMSE is not so sensitive to this problem as GFC because the square error is averaged across wavelengths when computing this measure.

3.2.2 Experiment VII: Influence of data set used

As we explained in the methods chapter in Section 2.3, instead of building a pool by joining all the reflectances together, as we did for the previous application, we have used the four different sets independently for the spectral estimation application. We divided the data sets into training and test sets and trained all the models using a k-fold approach as explained in section 2.6. In Figure 37, we show the spectral estimation results for each data set corresponding to filtered TFDs and polynomial Kernels. A very similar trend was found for the rest of the models and for the unfiltered sensors.

We can see in the figure how for the color difference metric, the performance is quite similar

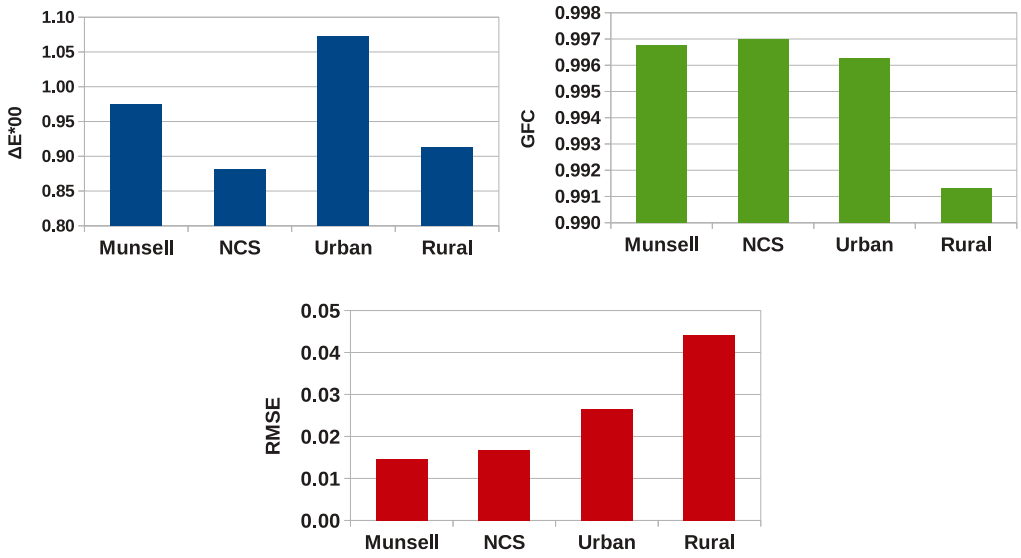

Figure 37: ΔE^*_{00} (blue), GFC (green) and RMSE (red) of Polynomial Kernels for the filtered case using different data sets

for all the data sets (note that the scale in the y-axis is quite large). If we have a look at the spectral metrics, we will clearly see how standard sets (Munsell and NCS) give better results than the sets coming from the spectral images. The rest of the data for the different models and the unfiltered case, including the ones plotted in the figure can be seen in Tables from 15 to 20 in the Appendix. In addition, in Table 7, we can see the results for the different data sets averaging the performance of every model for all the metrics and both filtered and unfiltered cases together.

Table 7: Average results comparing the four data sets averaging for all models and both filtered and unfiltered cases together

Data set	ΔE^*_{00}	GFC	RMSE
Munsell	1.58	0.996	0.016
NCS	1.50	0.996	0.019
Urban	2.06	0.995	0.029
Rural	2.63	0.988	0.050

We see how the rural set gives the lowest accuracy for all metrics, and Munsell and NCS the best for the spectral and color metrics respectively. In Table 8 we show the average results corresponding to the three metrics comparing standard sets with spectral images sets. The first row correspond to the average of the first two rows of Table 7 and the second row to the average of the third and fourth rows of the same table.

Table 8: Average results comparing standard versus spectral images data sets

Group of sets	ΔE^*_{00}	GFC	RMSE
Standard	1.54	0.996	0.017
Spectral images	2.34	0.991	0.040

The reason for the inferior performance of the data sets from the spectral images is that these spectral images were taken with a monochrome camera and a Liquid Crystal Tunable Filter (LCTF), to let pass only light wavelength by wavelength. This system has very low global responsivity in the blue extreme of the spectrum and thus, the SNR is lower for the short wavelength bands, yielding as artifact the typical tails which are difficult to be recovered specially for the filtered case (see Figure 36 bottom). This fact plus some issues concerning the assumption of uniform illumination, give as result less smooth reflectances than we can find in the standard databases. For a graphic proof of this see Figure 38, where some randomly chosen reflectances from rural set are shown in the right and some others from Munsell set are shown in the right.

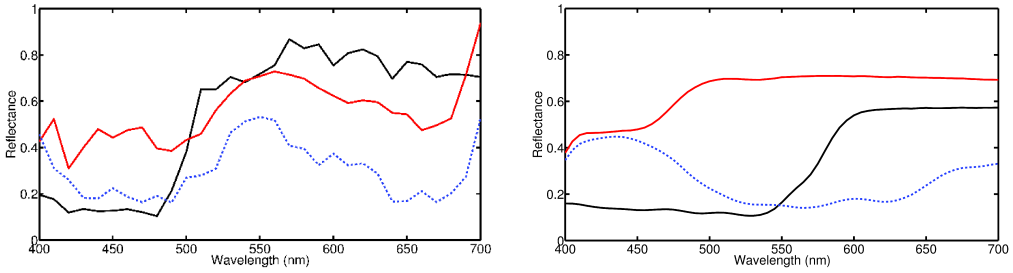

Figure 38: Some examples of reflectances extracted randomly from Rural data set (left) and Munsell data set (right)

Note the perceptible difference in smoothness of the reflectances from the different sets. Naturally, spiky reflectances are more difficult to be accurately recovered than smooth ones. If we would introduce artificially some noise in a smooth reflectance, we could get a similar effect.

3.2.3 Experiment VIII: Influence of model used

We saw how for the color calibration application, Kernel models perform better than others (see Subsection 3.1.4). In the spectral estimation application the same results were found. In Figure 39, we show the results of the color and spectral metrics for the different models in the filtered case only with Munsell data set. The behavior was the same for the unfiltered case. We refer the reader to the Tables 15 to 20 in the Appendix for a compete set of results including the ones shown in Figure 39.

We see how both Kernel models have almost the same performance for all the metrics studied and they are better than Wiener and Pseudoinverse. To have a global view of the effect of the model used, in Table 9 we show the average for all the data sets used and both filtered and unfiltered cases of each of the models in the three metrics studied.

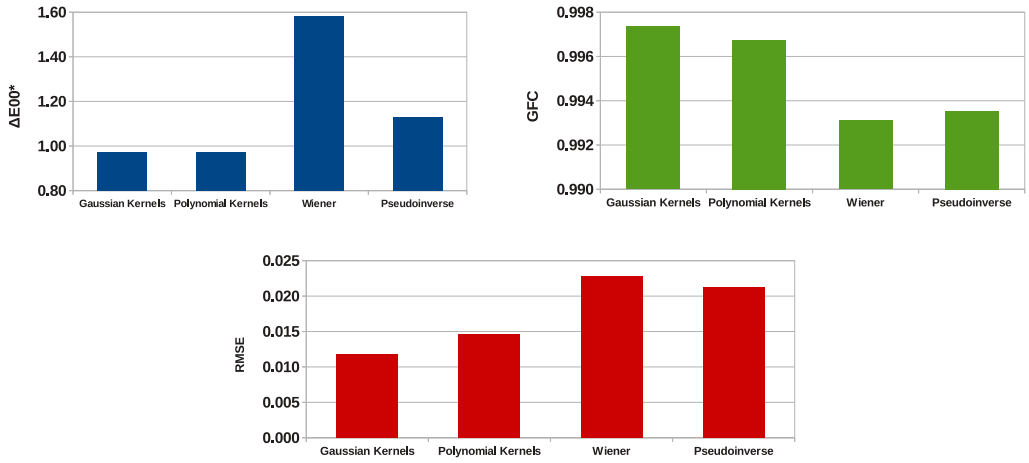

Figure 39: Model comparison for Munsell data set in the filtered case

Table 9: Comparison of different models averaging results

Model	ΔE^*_{00}	GFC	RMSE
Gaussian Kernels	1.53	0.996	0.022
Polynomial Kernels	1.59	0.996	0.023
Wiener	2.64	0.988	0.039
Pseudoinverse	2.01	0.994	0.029

Gaussian Kernels perform best together with Polynomial Kernels which give almost the same results. We get better color difference (around half ΔE^*_{00} unit in average and also better spectral quality of the recovered reflectances) using Kernel models. Comparing Wiener and Pseudoinverse models, we see how Pseudoinverse performs better in average for all metrics. Even though Wiener includes a parameter representing the noise of the system, since the noise in TFDs is not very high, and the optimization of this parameter was done somewhat coarsely by selecting among different orders of magnitude, the results are slightly worse than for Pseudoinverse. As Kernels performed best, we did not go further with the optimization of Wiener for comparing it against Pseudoinverse.

3.3 Direct vs indirect color calibration

In the first application studied, we were doing a direct mapping from sensors' responses to tristimulus values, using a system with three or six channels. In the second application the mapping was done to spectral reflectance and from there to tristimulus values the conversion is straight forward (indirect mapping), and we were using a 24 channels system.

The different conditions of both applications studied (regarding requirements of models used, number of channels, etc) make the comparison between direct and indirect mappings impossible

from the previous data. This is why we designed and performed a special experiment to compare them fairly under the same conditions.

We show in Subsection 3.3.1 the results of this experiment, and in Subsection 3.3.2 the results of the comparison also under the same conditions of Retiga and TFD for the indirect mapping.

3.3.1 Experiment IX: direct versus indirect mapping

In order to be able to compare fairly and under the same conditions, the performance of direct and indirect mapping, we included an experiment of mapping directly with the 24 channels from sensors' responses to tristimulus values and then compare results. We did it only for the Munsell data set (because there is not any *a priori* reason to think that results would be data-dependent) and the models giving best results in previous experiments, and the results found are shown in Figure 40.

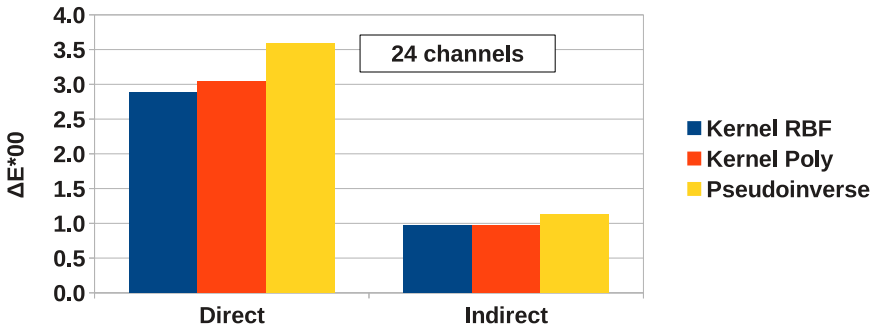

Figure 40: Comparison of indirect and direct color calibration for Munsell data set in the 24 channels filtered case

From the analysis of these results we conclude that the indirect mapping is performing better than the direct one. Apart from the relative comparison between approaches, we see also how for Kernel models, we are reaching average ΔE^*_{00} values below 1 unit, which is a very good accuracy taking into account that we are including noise in the camera responses.

3.3.2 Experiment X: Retiga versus the very best of TFDs

As a last experiment, since the first application of this work, we saw how the Retiga sensor, with three channels, was able to get better results than any of the TFD approaches used for color calibration due to its specially narrower responsivities. We wanted to compare the final best performance we have reached with the TFD after the proposed and implemented improvements (multi-shot filtered with indirect mapping) with those very good results the Retiga was giving. We did a last experiment which results are shown in Table 10. In this experiment we performed an indirect mapping for the Retiga, and compare the results with those of TFD, 24 channels, filtered case.

We see how finally, the TFD outperforms the Retiga. This comparison may not be fair from the point of view of the number of shots, since the Retiga has only three channels (one shot) and

Table 10: ΔE^*_{00} results for Retiga and 24 channels filtered TFD system using indirect mapping and Kernel models

Model	Retiga	TFD + filter
Gaussian Kernels	1.11	0.97
Polynomials Kernels	1.08	0.97

the TFD has 24 (eight shots). Still for fast applications where we can only afford short capturing times, Retiga sensor can give higher accuracy, but in other applications where we can have longer capture times, TFD gives not only better results but also full spatial resolution. In any case, whatever the application is, if we want to acquire a full spatial resolution color calibrated or spectral image, we have to use TFD sensors. For future work, we would like to study for which number of shots the performance of the TFD equals that of Retiga.

4 Conclusions and future work

In this chapter we briefly present the conclusions of this work and propose some future research lines.

We have analyzed the potential of TFD sensors to carry on two different tasks: color calibration and spectral reflectance estimation. The former was never studied for this kind of sensors, and the latest only with relatively simple methods. These two tasks are correlated, in the sense that we can perform color calibration either directly mapping from sensor responses to tristimulus values or doing it passing through spectral reflectance estimation. We have studied both tasks for general purpose, without focusing in a concrete application. We have shown that TFDs, can be used in a multispectral imaging device, to get full spatial resolution spectral images, or color corrected images.

For direct color calibration, we have show that only one shot taken with a TFD sensor is not enough to get accurate results. However, adding more shots to our capturing process and tuning the responsivities of the TFD between shots, we can improve results and still work with fast capturing times, allowing the use of this set-up for real-time applications. Only adding one extra shot, we can improve the accuracy in an average of 10%. Besides, we have demonstrated that adding a band-pass optical filter that is commercially available, we can improve the results even more, reducing the average error down to 20%. We have shown how filtering the responsivities of the TFD, they are more similar with the color matching functions (CMFs). As future work we would like to investigate the possibility of determining which is the optimum filter (physically implementable) to narrow down the responsivities of TFDs while preserving the full spatial resolution property and minimizing both the color correction error.

Regarding the models implemented for color calibration, we have used Kernel-based models for the first time in this context of direct color correction, and we have found that they perform best, compared with other typical models used for direct color calibration like Neural Networks and Polynomials. We have studied the behavior of these last two widely used models in the parameterization process, to be sure that their performance was the best we could get, and was not biased by overfitting. For regularized least squares fitting with Polynomials we have seen how the performance improves as we increase the degree of the polynomial up to seven. After that the model overfits the training data set and the performance decreases for higher polynomial degrees. For the case of neural networks, RBFNN reach the stability regime after adding around 100 neurons to the hidden layer, no matter the spread of the Gaussian function we are using. For smaller number of neurons, the higher the spread of the Gaussian function is, the less neurons are needed to reach the stable regime. In case of the FFBPNN, only one or two hidden layers are enough to have a good performance. The number of neurons per hidden layer is more important

when we increase the number of layers in the network. The optimal architecture was two layers and seven neurons per layer.

Summarizing, for direct color calibration we can use conventional RGB sensors, whose responsivities are narrower that those of TFDs and get satisfactory accuracy, but we need to take into account that the spatial resolution is one third of the one offered by TFD based sensors.

For the spectral reflectance estimation task, we have used a configuration of eight shots (24 channels), to push to the limit the capacity of the TFDs using the responsivities the developers at the Polytechnic University of Milan provided. Even though the eight shots can be taken in a very short time, this configuration is oriented to applications where we can afford longer capturing times and so we are under more controlled conditions. Using this set-up we can fairly estimate the reflectance of every pixel in an image, and if we include the same filter used for the direct color calibration, the colorimetric accuracy improves around 57% in average, although the spectral accuracy is worse. We have seen how the filter helps to recover the central area of the spectrum, but spoils the capacity of recovering the extremes of it because it is a cut-off filter with close to zero transmittance in these areas. Our future work will tackle also the optimization of filters for the specific task of spectral estimation, trying to find a trade-off between improving the color distance metrics and not worsening too much or even improving also the spectral metrics.

With respect to the models used for spectral reflectance estimation, Kernel-based models were performing best, in comparison to other widely used ones like Pseudoinverse and Wiener for all the different data sets used. This is a novel result since Kernel methods for spectral estimation were not used before for TFD sensors. We have also seen how the quality of the data set influences the performance of the models. If we have a data set which is noisy or has artifacts due to the acquisition process of the spectral data, the estimations may be worse than for other clean data sets composed of smooth reflectances.

As a final step, we conducted two experiments to show how, for color calibration, using 24 TFD filtered channels, the indirect mapping passing through spectral reflectance gives better results than the direct mapping (improving around 66% the accuracy in average). Besides, in order to compare the performance of the RGB, three-channels Retiga sensor with the filtered TFD taking eight shots, we calculated an indirect mapping for both approaches, and we found out that TFD performs 10% better and gives full spatial resolution. Thus, we conclude that if we can afford longer capturing times in our specific application, then TFDs plus filter are definitely the best option.

Comparing the very first performance we found for single-shot unfiltered TFD for direct color calibration, with the very best final performance reached for the last configuration with eight shots, filter and indirect mapping, we have reduced the colorimetric error in 70%, outperforming the good results Retiga was giving and getting values below 1 ΔE^*_{00} unit in average.

Since TFDs are still a prototype, we have no real image data available to feed the models. We

have used simulations of the capture mimicking the electronic noise processes present in TFDs. All the experiments were conducted under an average condition of 42dB of SNR, which is realistic for these kind of sensors. Nevertheless, we expect in the future to be able to obtain real sensor responses when the TFD sensor is enlarged and coupled to the appropriate optics and housing.

In [72], a local-area tuning is proposed for tunable sensors. The authors deal with theoretical ideal Gaussian responsivities curves that can be tuned freely to any wavelength. The system take a first shot with normal RGB configuration to study the color information present in each area of the image. Then, depending on the color found, different areas of the sensor can be tuned to different modes (red mode, green mode, blue mode, etc...), in such a way that the multispectral capture is optimal for each area of the image depending on the information in the scene. We would like to study the possibility of designing a system which is not only local-area tunable but pixel-wise using real TFD responsivity functions. In addition, we would like to combine it with adaptive training approaches. Once the color information of the image is studied pixel-wise, not only the responsivities of each pixel are tuned accordingly, but also the training set used to train the models adapts locally. Thus we train the models only with samples whose color information is close to that found in the RGB picture.

5 Appendix

Table 11: ΔE^*_{00} results for all set-ups, with a training set size of 2920 samples (66% of total set)

Model	Single TFD	Single filtered	Multishot	Multi filtered
Gaussian Kernels	3.08	3.21	2.76	2.47
Polynomials Kernels	3.08	3.16	2.73	2.47
RBFNN	3.08	3.17	2.74	2.43
FFBPNN	3.31	3.30	3.18	2.51

Table 12: ΔE^*_{00}, GFC and RMSE results comparing unfiltered and filtered approaches for 24 channels and NCS data set

	ΔE^*_{00}		GFC		RMSE	
Model	**Unfiltered**	**Filtered**	**Unfiltered**	**Filtered**	**Unfiltered**	**Filtered**
Gaussian Kernels	1.50	0.88	0.998	0.997	0.013	0.016
Polynomial Kernels	1.62	0.88	0.998	0.997	0.014	0.017
Wiener	2.26	1.55	0.995	0.993	0.021	0.025
Pseudoinverse	2.31	1.01	0.996	0.994	0.020	0.023

Table 13: ΔE^*_{00}, GFC and RMSE results comparing unfiltered and filtered approaches for 24 channels and Urban data set

	ΔE^*_{00}		GFC		RMSE	
Model	**Unfiltered**	**Filtered**	**Unfiltered**	**Filtered**	**Unfiltered**	**Filtered**
Gaussian Kernels	2.29	1.04	0.997	0.997	0.020	0.025
Polynomial Kernels	2.26	1.07	0.997	0.996	0.020	0.027
Wiener	3.67	1.74	0.991	0.987	0.038	0.047
Pseudoinverse	3.02	1.35	0.996	0.994	0.023	0.032

Table 14: ΔE^*_{00}, GFC and RMSE results comparing unfiltered and filtered approaches for 24 channels and Rural data set

	ΔE^*_{00}		GFC		RMSE	
Model	**Unfiltered**	**Filtered**	**Unfiltered**	**Filtered**	**Unfiltered**	**Filtered**
Gaussian Kernels	3.25	0.92	0.993	0.991	0.038	0.044
Polynomial Kernels	3.33	0.91	0.993	0.991	0.039	0.044
Wiener	3.82	0.95	0.980	0.972	0.064	0.078
Pseudoinverse	3.82	0.95	0.991	0.989	0.044	0.050

Table 15: ΔE^*_{00} results for all models and the four different data sets for the filtered case

Model	Munsell	NCS	Urban	Rural
Gaussian Kernels	0.97	0.88	1.04	0.91
Polynomials Kernels	0.97	0.88	1.07	0.92
Wiener	1.58	1.55	1.74	1.57
Pseudoinverse	1.13	1.01	1.35	0.95

Table 16: ΔE^*_{00} results for all models and the four different data sets for the unfiltered case

Model	Munsell	NCS	Urban	Rural
Gaussian Kernels	1.39	1.50	2.29	3.25
Polynomials Kernels	1.63	1.62	2.26	3.33
Wiener	2.46	2.26	3.67	6.29
Pseudoinverse	2.51	2.31	3.02	3.82

Table 17: GFC results for all models and the four different data sets for the filtered case

Model	Munsell	NCS	Urban	Rural
Gaussian Kernels	0.997	0.997	0.997	0.991
Polynomials Kernels	0.997	0.997	0.996	0.991
Wiener	0.993	0.993	0.987	0.972
Pseudoinverse	0.994	0.994	0.994	0.989

Table 18: GFC results for all models and the four different data sets for the unfiltered case

Model	Munsell	NCS	Urban	Rural
Gaussian Kernels	0.999	0.998	0.997	0.993
Polynomials Kernels	0.998	0.998	0.997	0.993
Wiener	0.995	0.995	0.991	0.980
Pseudoinverse	0.995	0.996	0.996	0.991

Table 19: RMSE results for all models and the four different data sets for the filtered case

Model	Munsell	NCS	Urban	Rural
Gaussian Kernels	0.012	0.016	0.025	0.044
Polynomials Kernels	0.015	0.017	0.027	0.044
Wiener	0.023	0.025	0.047	0.078
Pseudoinverse	0.021	0.023	0.032	0.050

Table 20: RMSE results for all models and the four different data sets for the unfiltered case

Model	Munsell	NCS	Urban	Rural
Gaussian Kernels	0.010	0.013	0.020	0.038
Polynomials Kernels	0.012	0.014	0.020	0.039
Wiener	0.020	0.021	0.038	0.064
Pseudoinverse	0.018	0.020	0.023	0.044

Bibliography

[1] Longoni, A., Zaraga, F., Langfelder, G., & Bombelli, L. 2008. The transverse field detector (tfd): a novel color-sensitive cmos device. *Electron Device Letters, IEEE*, 29(12), 1306–1308.

[2] Grahn, H. & Geladi, P. 2007. *Techniques and applications of hyperspectral image analysis*. Wiley.

[3] Kinnunen, J., Saarakkala, S., Hauta-Kasari, M., Vahimaa, P., & Jurvelin, J. 2011. Optical spectral reflectance of human articular cartilage–relationships with tissue structure, composition and mechanical properties. *Biomedical optics express*, 2(5), 1394–1402.

[4] Noordam, J., van den Broek, W., & Buydens, L. 2005. Detection and classification of latent defects and diseases on raw french fries with multispectral imaging. *Journal of the Science of Food and Agriculture*, 85(13), 2249–2259.

[5] Noordam, J., van den Broek, W., & Buydens, L. 2004. Perspective of inline control of latent defects and diseases on french fries with multispectral imaging. *Monitoring food safety, agriculture, and plant health: 29-30 October 2003, Providence, Rhode Island, USA*, 5271, 527185.

[6] Nischan, M., Joseph, R., Libby, J., & Kerekes, J. 2003. Active spectral imaging.

[7] Shaw, G. & Burke, H. 2003. Spectral imaging for remote sensing. *Lincoln Laboratory Journal*, 14(1), 3–28.

[8] Golub, M., Nathan, M., Averbuch, A., Lavi, E., Zheludev, V., & Schclar, A. 2009. Spectral multiplexing method for digital snapshot spectral imaging. *Applied optics*, 48(8), 1520–1526.

[9] Du, H., Tong, X., Cao, X., & Lin, S. 2009. A prism-based system for multispectral video acquisition. In *Computer Vision, 2009 IEEE 12th International Conference on*, 175–182. IEEE.

[10] Murakami, Y., Yamaguchi, M., & Ohyama, N. 2012. Hybrid-resolution multispectral imaging using color filter array. *Optics Express*, 20(7), 7173–7183.

[11] Langfelder, G., Zaraga, F., & Longoni, A. 2009. Tunable spectral responses in a color-sensitive cmos pixel for imaging applications. *Electron Devices, IEEE Transactions on*, 56(11), 2563–2569.

[12] Hubel, P. 2005. Foveon technology and the changing landscape of digital cameras, inthirteenth is&t color imaging conference'. In *Thirteenth IS&T Color Imaging Conference*, 314.

[13] Ramanath, R., Snyder, W., Bilbro, G., & Sander III, W. 2002. Demosaicking methods for bayer color arrays. *Journal of Electronic imaging*, 11, 306.

[14] Gunturk, B., Glotzbach, J., Altunbasak, Y., Schafer, R., & Mersereau, R. 2005. Demosaicking: color filter array interpolation. *Signal Processing Magazine, IEEE*, 22(1), 44–54.

[15] Lu, W. & Tan, Y. 2003. Color filter array demosaicking: new method and performance measures. *Image Processing, IEEE Transactions on*, 12(10), 1194–1210.

[16] Lyon, R. & Hubel, P. 2002. Eyeing the camera: Into the next century. In *Tenth Color Imaging Conference: Color Science and Engineering Systems, Technologies, Applications*, 349–335.

[17] Langfelder, G. 2012. Spectrally reconfigurable pixels for dual-color-mode imaging sensors. *Applied Optics*, 51(4), A91–A98.

[18] Langfelder, G., Malzbender, T., Longoni, A., & Zaraga, F. 2011. A device and an algorithm for the separation of visible and near infrared signals in a monolithic silicon sensor. In *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, volume 7882, 5.

[19] Martínez-Domingo, M., Valero, E., Hernández-Andrés, J., & Langfelder, G. 2012. Spectral reflectance estimation from transverse field detectors responses. In *CGIV 2012, SixthEuropean Conference on Colour in Graphics, Imaging, and Vision. 12th International Symposium on Multispectral Colour Science*.

[20] Zhao, Y. & Berns, R. 2007. Image-based spectral reflectance reconstruction using the matrix r method. *Color Research & Application*, 32(5), 343–351.

[21] Stark, H., Yang, Y., & Yang, Y. 1998. *Vector space projections: A numerical approach to signal and image processing, neural nets, and optics*, volume 2. Wiley.

[22] Heikkinen, V., Jetsu, T., Parkkinen, J., Hauta-Kasari, M., Jaaskelainen, T., & Lee, S. 2007. Regularized learning framework in the estimation of reflectance spectra from camera responses. *JOSA A*, 24(9), 2673–2683.

[23] Heikkinen, V., Lenz, R., Jetsu, T., Parkkinen, J., Hauta-Kasari, M., & Jääskeläinen, T. 2008. Evaluation and unification of some methods for estimating reflectance spectra from rgb images. *JOSA A*, 25(10), 2444–2458.

[24] Shimano, N. 2006. Recovery of spectral reflectances of objects being imaged without prior knowledge. *Image Processing, IEEE Transactions on*, 15(7), 1848–1856.

[25] Usui, S., Arai, Y., & Nakauchi, S. 2000. Neural networks for device-independent digital color imaging. *Information Sciences*, 123(1), 115–125.

[26] Hong, G., Luo, M., & Rhodes, P. 2001. A study of digital camera colorimetric characterisation based on polynomial modelling. *Color Research & Application*, 26, 76–84.

[27] Maloney, L. & Wandell, B. 1986. Color constancy: a method for recovering surface spectral reflectance. *JOSA A*, 3(1), 29–33.

[28] Imai, F. & Berns, R. 1999. Spectral estimation using trichromatic digital cameras. In *Proceedings of the International Symposium on Multispectral Imaging and Color Reproduction for Digital Archives*, volume 42.

[29] Shi, M. & Healey, G. 2002. Using reflectance models for color scanner calibration. *JOSA A*, 19(4), 645–656.

[30] Vrhel, M. & Trussell, H. 1999. Color device calibration: A mathematical formulation. *Image Processing, IEEE Transactions on*, 8(12), 1796–1806.

[31] Schettini, R., Barolo, B., & Boldrin, E. 1995. Colorimetric calibration of color scanners by back-propagation. *Pattern recognition letters*, 16(10), 1051–1056.

[32] Finlayson, G. & Morovic, P. 2000. Metamer constrained color correction. *Journal of Imaging Science and Technology*, 44(4), 295–300.

[33] Vrhel, M. & Trussell, H. 1995. Optimal color filters in the presence of noise. *Image Processing, IEEE Transactions on*, 4(6), 814–823.

[34] Finlayson, G., Drew, M., & Funt, B. 1994. Spectral sharpening: sensor transformations for improved color constancy. *JOSA A*, 11(5), 1553–1563.

[35] Barnard, K. & Funt, B. 1998. Experiments in sensor sharpening for color constancy. In *6th Color Imaging Conference: Color, Science, Systems and Applications*, 43–46. Citeseer.

[36] Piché, R. 2002. Nonnegative color spectrum analysis filters from principal component analysis characteristic spectra. *JOSA A*, 19(10), 1946–1950.

[37] Ng, D. & Allebach, J. 2006. A subspace matching color filter design methodology for a multispectral imaging system. *Image Processing, IEEE Transactions on*, 15(9), 2631–2643.

[38] Chatzis, I., Kappatos, V., & Dermatas, E. 2007. Filter selection for multi-spectral image acquisition using the feature vector analysis methods. In *Intelligent production machines and systems: 2nd I* PROMS Virtual Conference, 3-14 July 2006*, 283. Elsevier Science.

[39] Langfelder, G., Buffa, C., Longoni, A., Pelamatti, A., & Zaraga, F. 2012. Active pixels of transverse field detector based on a charge preamplifier. In *Proceedings of SPIE*, volume 8299, 829906.

[40] López-Álvarez, M., Hernández-Andrés, J., Valero, E., & Romero, J. 2007. Selecting algorithms, sensors, and linear bases for optimum spectral recovery of skylight. *JOSA A*, 24(4), 942–956.

[41] Finlayson, G. & Süsstrunk, S. 2000. Spectral sharpening and the bradford transform. *Proc. CIS2000*, -, 236–243.

[42] Vora, P. & Trussell, H. 1993. Measure of goodness of a set of color-scanning filters. *JOSA A*, 10(7), 1499–1508.

[43] Vrhel, M. & Trussell, H. 1994. Filter considerations in color correction. *Image Processing, IEEE Transactions on*, 3(2), 147–161.

[44] Shawe-Taylor, J. & Cristianini, N. 2004. *Kernel methods for pattern analysis*. Cambridge Univ Pr.

[45] Cheung, V., Westland, S., Connah, D., & Ripamonti, C. 2004. A comparative study of the characterisation of colour cameras by means of neural networks and polynomial transforms. *Coloration technology*, 120(1), 19–25.

[46] Yea, L., Hongfeia, Y., & Junshenga, S. 2008. Camera characterization using back-propagation artificial neutral network based on munsell system. In *Proc. of SPIE Vol*, volume 6621, 66210A–1.

[47] Shen, H., Xin, J., & Shao, S. 2007. Improved reflectance reconstruction for multispectral imaging by combining different techniques. *Optics express*, 15(9), 5531–5536.

[48] Shimano, N., Terai, K., & Hironaga, M. 2007. Recovery of spectral reflectances of objects being imaged by multispectral cameras. *JOSA A*, 24(10), 3211–3219.

[49] Shen, H., Wan, H., & Zhang, Z. 2010. Estimating reflectance from multispectral camera responses based on partial least-squares regression. *Journal of Electronic Imaging*, 19, 020501.

[50] Heikkinen, V. *Kernel methods for estimation and classification of data from spectral imaging*. PhD thesis, Faculty of Forestry and natural Sciences, University of Eastern Finland, 2011.

[51] Simon, H. 1999. *Neural networks: a comprehensive foundation*. Prentice Hall.

[52] Ng, A. 2011. Machine learning course. *Lecture 8. Neural networks*, http://www.ml-class.org/course/class/index.

[53] Benoudjit, N., Archambeau, C., Lendasse, A., Lee, J., Verleysen, M., et al. 2002. Width optimization of the gaussian kernels in radial basis function networks. In *Proc. of ESANN*, 425–432.

[54] Li, X. 2008. Scanner color management model based on improved back-propagation neural network. *Chinese Optics Letters*, 6(3), 231–234.

[55] Matlab. Levenberg-marquardt algorithm for neural networks training in matlab. *http://www.weizmann.ac.il/matlab/toolbox/nnet/trainlm.html*.

[56] Stigell, P., Miyata, K., & Hauta-Kasari, M. 2007. Wiener estimation method in estimating of spectral reflectance from rgb images. *Pattern Recognition and Image Analysis*, 17(2), 233–242.

[57] Kohonen, O., Parkkinen, J., & Jääskeläinen, T. 2006. Databases for spectral color science. *Color Research & Application*, 31(5), 381–390.

[58] Parkkinen, J., Hallikainen, J., & Jaaskelainen, T. 1989. Characteristic spectra of munsell colors. *JOSA A*, 6(2), 318–322.

[59] Nascimento, S., Ferreira, F., & Foster, D. 2002. Statistics of spatial cone-excitation ratios in natural scenes. *JOSA A*, 19(8), 1484–1490.

[60] Foster, D., NASCIMENTO, S., & Amano, K. 2004. Information limits on neural identification of colored surfaces in natural scenes. *Visual neuroscience*, 21(03), 331–336.

[61] Nascimento, S., Foster, D., & Amano, K. 2005. Psychophysical estimates of the number of spectral-reflectance basis functions needed to reproduce natural scenes. *JOSA A*, 22(6), 1017–1022.

[62] Noboru, O. & Robertson, A. 2005. *Colorimetry (Fundamentals and Applications)*. West Sussex, England, Wiley & Sons.

[63] Berns, R. et al. 2000. *Billmeyer and Saltzman's principles of color technology*. Wiley New York.

[64] Sharma, G., Wu, W., & Dalal, E. 2005. The ciede2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. *Color Research & Application*, 30(1), 21–30.

[65] Melgosa, M. 2008. Color-difference formulas. In *Proceeding of 4th Conference of Balkan Light*.

[66] Valero, E., Nieves, J., Nascimento, S., Amano, K., & Foster, D. 2007. Recovering spectral data from natural scenes with an rgb digital camera and colored filters. *Color Research & Application*, 32(5), 352–360.

[67] Mansouri, A., Sliwa, T., Hardeberg, J., & Voisin, Y. 2008. An adaptive-pca algorithm for reflectance estimation from color images. In *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, 1–4. IEEE.

[68] Zhang, W. & Dai, D. 2008. Spectral reflectance estimation from camera responses by support vector regression and a composite model. *JOSA A*, 25(9), 2286–2296.

[69] Hardeberg, J. 2001. *Acquisition and reproduction of color images: colorimetric and multispectral approaches*. Dissertation. com.

[70] López-Álvarez, M., Hernández-Andrés, J., Valero, E., & Nieves, J. 2005. Colorimetric and spectral combined metric for the optimization of multispectral systems. In *Proceedings of the 10th Congress of the International Colour Association (AIC'05)*, 1685–1688. Association Internationale de la Couleur.

[71] Michalsky, J. 1985. Estimation of continuous solar spectral distributions from discrete filter measurements: Ii. a demonstration of practicability. *Solar energy*, 34(6), 439–445.

[72] Lin, A. & Imai, F. 2011. Efficient spectral imaging based on imaging systems with scene adaptation using tunable color pixels. In *19th Color and Imaging Conference. Final Program and Proceedings*.