https://doi.org/10.2352/EI.2023.35.15.COLOR-199

Machine Learning Estimation of Camera Spectral Sensitivity Functions with non-RGB Color Filters

Abraham Sachs, Ramakrishna Kakarala; Omnivision Technologies Inc.; Santa Clara, CA/USA

Abstract

The spectral sensitivity functions of a digital image sensor determine the sensor's color response to scene-radiated light. Knowing these spectral sensitivity functions is very important for applications that require accurate color, such as computer vision. Traditional measurements of these functions are time consuming, and require expensive lab equipment to generate narrow-band monochromatic light. Previous works have shown that sensitivity curves can be estimated using images of a color checker chart with known spectral reflectances, using either numerical optimization, or machine learning. However, previous works in the literature have not considered sensitivity functions for CFAs (color filter arrays) other than RGB, such as RCCB (Red Clear Blue) or RYYCy (Red Yellow Cyan). Non-RGB CFAs have been shown to be useful for automotive and security camera applications, especially in low light situations. We propose a machine learning method to estimate the sensitivity curves of sensors with non-RGB filters, in addition to the RGB filters addressed previously in the literature, using a single image of a color chart under unknown illumination.

Introduction

Most digital image sensors today do not have their spectral sensitivity curves readily available. In order to access the sensitivity curves, one must either measure them directly or somehow estimate them. Direct measurement is a long process that requires expensive lab equipment inaccessible to many people. The curves play a main role in determining the color response of a digital image sensor. Knowing their shape, therefore, is crucial for accurate color classification and differentiation, which are both important topics in computer vision. Having access to the sensitivity curves of a sensor allows for simulating the color response to any known reflectance spectrum under any illumination, without having to travel to the scene for a real image capture. A simple estimation process also aids research because curves which are not supplied by the manufacturer can now be compared to proposed sensors. There have been published methods [2, 3, 4] for estimating the sensitivity curves of sensors with a standard RGB CFA, but we propose a method which includes spectral sensitivity curves for sensors with either an RGB or a non-RGB filter such as RCCB or RYYCy. A diagram of our process is shown in Figure 1.

An example of a study of the advantages of non-RGB color filters in automotive cameras was discussed this year by Funatsu et al [1]. They found that image quality was improved by changing the CFA, and they were able to solve a problem with traffic light detection. Their work was based on the manipulation of the spectral sensitivity functions. One example of previous work done in estimating sensitivity functions comes from Tominaga et al [4]. They use the first principal component of their curve dataset to estimate the curves of mobile phone cameras. Their work relies on the fact that all of the mobile phone cameras they used have very similar shapes and can all be closely approximated by just the first principal component. Another example of previous work done in curve estimation comes from Ji et al [2]. Their study uses a compressive sensing approach to solve for weights of Gaussian basis functions to approximate the shape of the curves. An example of a neural network approach to estimating sensitivity curves is provided by He et al [3]. Both [2, 3] cover only RGB filters, and in this paper we expand on this problem to include non-RGB filters. Like [3], we use a neural network, but one which is much simpler than their six convolutional layer network.

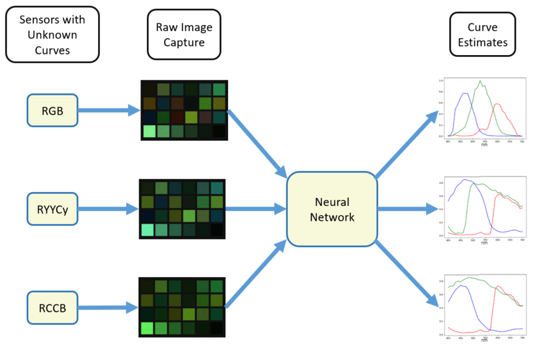

Figure 1: The steps of our estimation process. A sensor with unknown sensitivity curves captures a raw image of a color chart. The tri-chromatic values of chart tiles collected from the demosaiced image are fed into the neural network which then outputs an estimate of the sensor's curves

In this paper, we investigate machine learning methods to accurately estimate the sensitivity curves of various color filters, including both RGB filters that have been previously studied, as well as the non-RGB filters that are important for automotive and security camera applications. We develop techniques that work from a single image of a color checker chart taken under unknown illumination. We also compare our methods to previous publications, which to date have not considered non-RGB filters. Including non-RGB filters makes the estimation problem much more challenging, since the resulting space of color filters is no longer modelled by simple shapes, such as Gaussians. Allowing for unknown illumination also provides challenges, as we cannot simply assume that the image has been properly white balanced. Currently, finding the curves for non-RGB filters requires the slow, traditional measurement techniques and expensive equipment. By including non-RGB filters in our estimation, we can improve the accessibility of curves for filters used in more diverse camera applications.

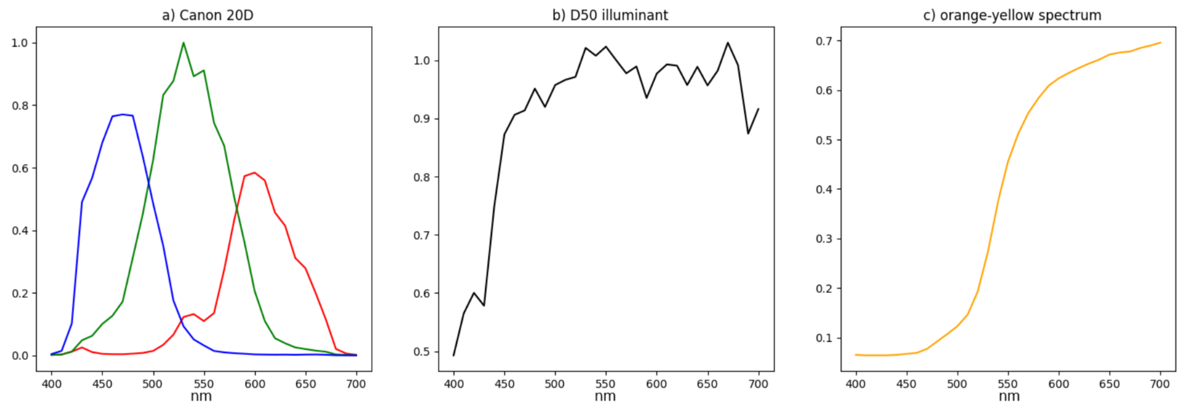

Figure 2: Example sensitivity curves, illuminant spectrum, and reflectance spectrum.

Background

Sensitivity Curves

The spectral sensitivity curve of a channel of an image sensor represents that channel's response to incident light at each wavelength in the visible spectrum, approximately 400-700nm. An example of the sensitivity curves of a Canon 20D image sensor is shown in Figure 2a. We will denote the three curves as S_1, S_2, and S_3, and the channel responses as C_1, C_2, and C_3. The incident light to the sensor at some point depends on both the illumination spectrum of the scene, denoted by ℓ, as well as the reflectance spectrum of the color at that point, denoted by R. An example of the D50 illumination spectrum and the reflectance spectrum of the orange-yellow tile of a color chart are shown in Figure 2b and 2c respectively. The response of one channel, C_i, to some color is given by:

$$C_i = \int_{400nm}^{700nm} S_i(\lambda)\ell(\lambda)R(\lambda)d\lambda \tag{1}$$

Applying this with each tile of a color chart results in a sensor's trichromatic response to that tile. For the purposes of this paper, we sampled all of these spectra at 31 points, from 400 to 700nm with 10nm spacing between samples. We could then represent integration by matrix multiplication. The three channel response to a single tile becomes:

$$\begin{bmatrix} C_1 & C_2 & C_3 \end{bmatrix}^T = S * L * R \tag{2}$$

Where S is a 3×31 matrix corresponding to the sampled spectra of each channel's sensitivity curve, L is a 31×31 diagonal matrix which has the sampled illumination spectrum along its diagonal, and R is a 31×1 vector representing the sampled reflectance spectrum. Although it seems like this inverse problem could be solved by a simple least squares solution, previous work has shown that the dimensionality of the problem is too high for a least squares approach [2, 4].

Principal Component Analysis and Intuition

At first glance, this is an underdetermined problem, we are trying to estimate 93 unknowns using only 72 inputs. When we carry out a principal component analysis of our dataset, however, we find that the dimensionality of our set of curves is just 9 to

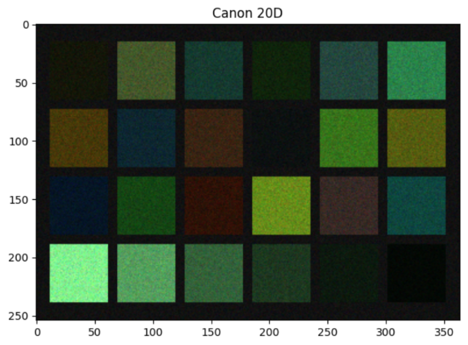

Figure 3: Colors generated by the curves shown in Figure 2a

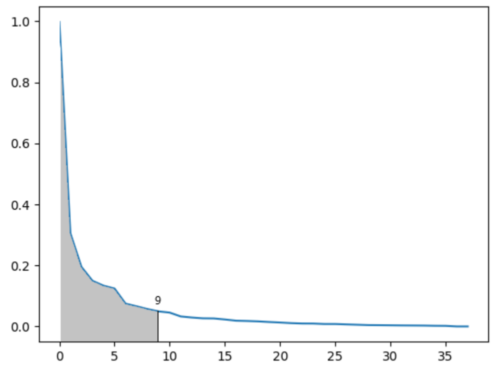

Figure 4: Principal component contribution graph. Converges at the 9th component

capture 99% of variations, meaning we should be able to make good estimates with our 72 inputs. The graph of the principal component contributions is shown in Figure 4. As for estimations under unknown illumination, we believe that the network is able to estimate and account for the contribution of the illuminant over ranges of the spectrum. Many of the tiles of a color chart have overlap in regions of their reflectance spectra, such as the blue sky and foliage tiles shown in Figure 5. Because of this overlap, the channel responses to these tiles are affected very similarly by the illuminant in these ranges, and the ratio of the channel response in those two tiles can be used to approximate the illuminant contribution in that wavelength range.

The first nine principal components are shown in Figure 6,

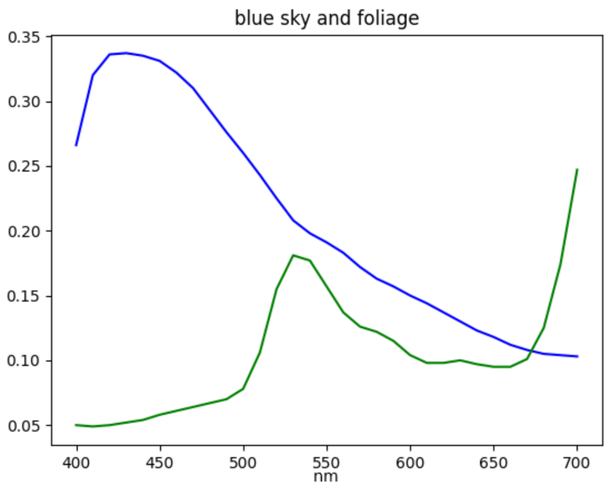

Figure 5: The blue sky and foliage tiles are close in the 550-650nm range

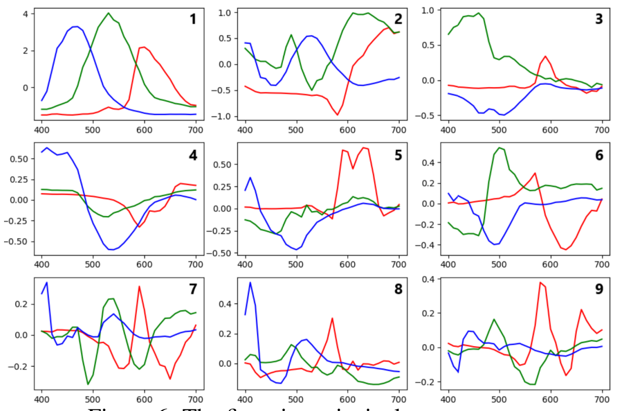

Figure 6: The first nine principal components

and are labeled in their order of importance. The first component has a distinctly RGB shape to it, which accounts for most of the simple RGB sensors. The features of non-RGB curves are distributed throughout the other components. For example, in components 2 and 3, we clearly see the right and left sides of a clear channel; in component 6, we see a spike that corresponds to a yellow channel. Our principal component analysis shows that the features of these curves can be represented by much fewer than 93 dimensions. A principal component method, however, would not be able to handle the added complexity of an unknown illumination which can be overcome by a neural network. We are also better able to enfore non-negativity, which is not a given in principal components, with our neural network.

Method

We built a fully connected neural network in Pytorch which would take in the 24 trichromatic values of the tiles of a color chart and output the estimated shape of the curves which generated the colors.

Training Data

To create training data, we needed spectral sensitivity curves, illuminant data, and reflectance data of a color chart. In order to make the network more robust to imperfect illumination, either from a non-standard illuminant or from non-uniform illumination, we needed to apply some illuminant error to the color generating process. To simulate this effect, we used the illumination spectra of some LED lighting and multiplied it with the standard illuminants to create some illuminant imperfections.

We generated 15808 sets of 24 tri-chromatic values using 38 known sensor sensitivity curves, 28 of the curves collected from [7] and 10 from [1], 8 LED illuminant spectra used as sources of error collected from a lamp database [6], with 52 different illuminants and the 24 reflectance spectra of color charts both collected from the colour-science python package [5]. The trichromatic responses were calculated then stored in a matrix, C, to be used by the network as shown in equation 3.

$$C = S * L * L_{err} * R_{chart} \tag{3}$$

Where S and L are the same as before, C is a 3×24 matrix storing the 24 trichromatic values of the chart tiles, L_{err} is a 31×31 diagonal matrix representing an LED illuminant spectrum meant to simulate imperfect illumination, and R_{chart} is a 31×24 matrix storing the 24 different reflectance spectra of the color chart. The C matrices were saved to a text file and labeled with the curve which generated the colors to be used as the ground truth while training. 10000 sets of values were labeled for training, 3000 for a test set, and the remaining 2808 were set aside for validation. An example of the colors generated by the Canon 20D under D50 illumination with no LED error is shown in Figure 3. When displaying colors, we add a dark grey border as well as Gaussian noise to make it look like an actual captured image of a color chart. During the generation process, 15% of the trichromatic triplets were randomly set to (0,0,0) to discourage the network from relying too heavily on any particular set of tiles.

Network Structure

A visualization of the fully connected network structure is shown in Figure 7 [9]. The network has a 72-node input layer corresponding to the 24 average tri-chromatic values of the tiles in the input image. The network has a single 100-node hidden layer. The output layer has 93 nodes, 31 for each of the three curves. Because we expect only positive values at the output, we used a leaky relu activation function which passes positive values

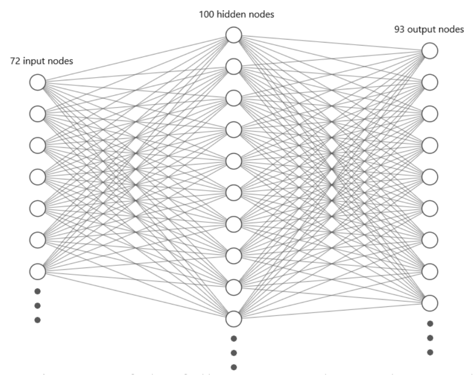

Figure 7: Diagram of the fully connected neural network structure. 72 node input layer, 100 node hidden layer, and a 93 node output layer

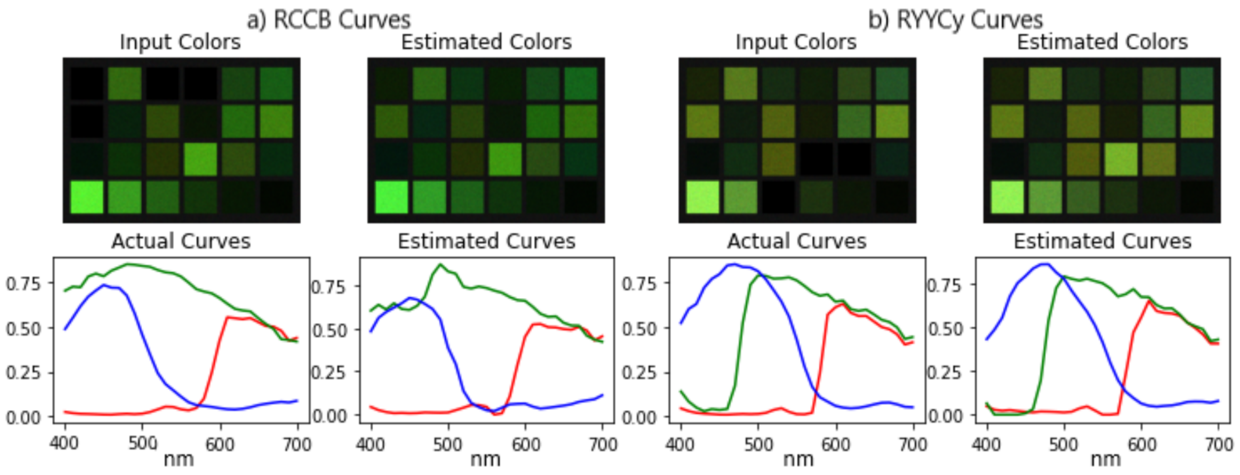

Figure 8: Input colors, ground truth curves, estimated colors, and estimated curves for RCCB and RYYCy color filters

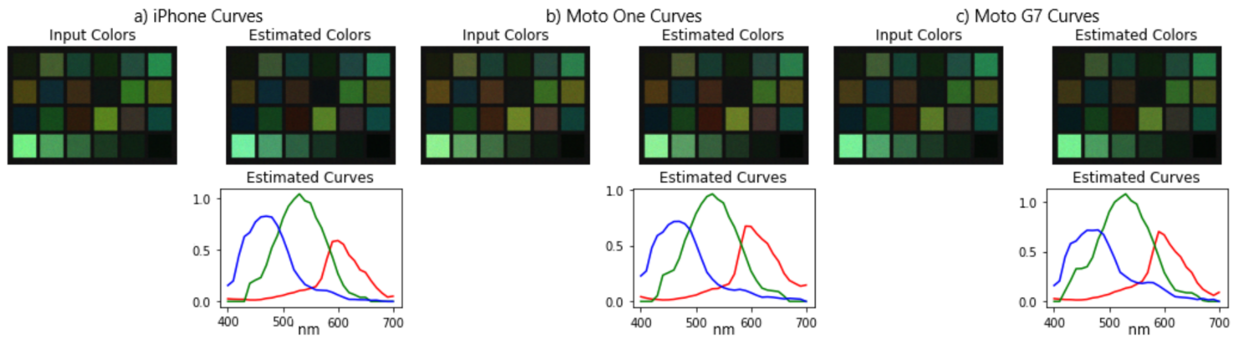

Figure 9: Actual RGB sensor curve estimates

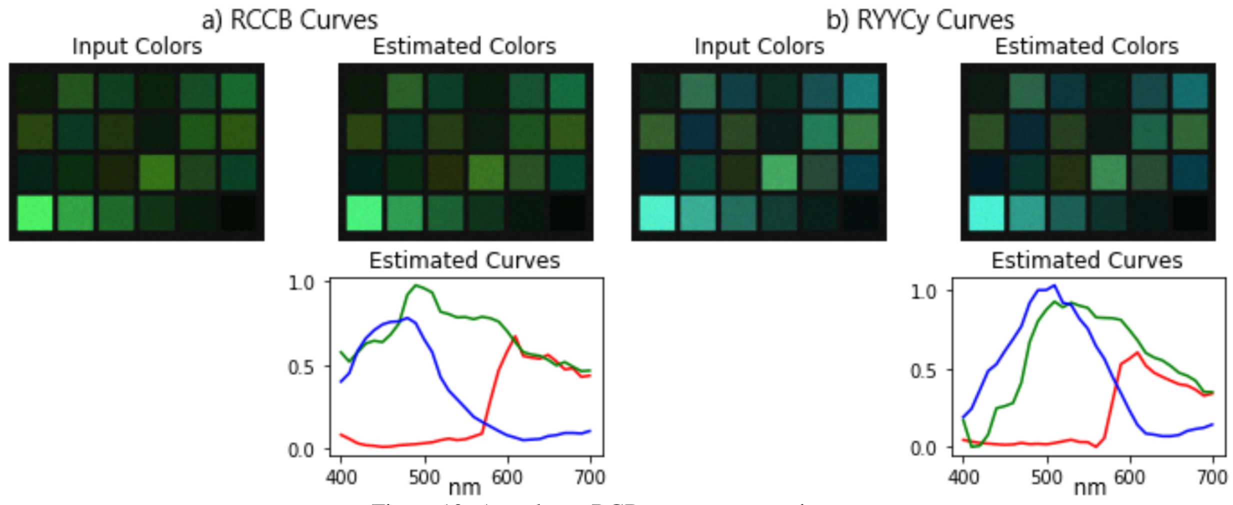

Figure 10: Actual non-RGB sensor curve estimates

and reduces the impact of negative values. In our case, we used a slope of 0.01 for the negative range. We used the leaky relu because, when using a standard relu, there was a tendency for neurons to "die", or stop updating with their output stuck at 0. The loss function, shown in Equation 4, was defined as the mean squared error between the 93 points of the output curves, *Out*, and the ground truth curves which generated the input values, *GT*. The network was trained for 100 epochs at a learning rate of 1E-4.

$$loss = \frac{\sum_{i=1}^{93}(GT_i - Out_i)^2}{93} \tag{4}$$

Results

Test Data

At the end of training, the loss function converged to around 1E-3 for most of the test data, 1E-2 for some of the larger error samples. The network achieved a good fit for most samples, and in some cases there was little to no visual difference between the estimated and the ground truth curves. The loss was on the same level regardless of the type of color filter. Some examples of a RCCB and a RYYCy estimate are shown in Figure 8a and 8b respectively. We see that the shapes of the curves are followed well by this relatively simple neural network. The colors generated by the estimated curves are also shown as another visual indication of the accuracy of the estimation.

Actual Sensor Estimates

With the network fully trained, we wished to estimate some real sensors with unknown sensitivity curves. For standard RGB sensors, we supplied the trichromatic data of demosaiced images from an iPhone X, a Motorola One Vision, and a Motorola G7, all under D50 illumination. For non-RGB sensors, we supplied the trichromatic data from one RCCB sensor and one RYYCy sensor both under D65 illumination. The estimated RGB curves are shown in Figure 9, and the estimated non-RGB curves are in Figure 10. Because the ground truth curves are not available to us, we validate the estimations by using the estimated curves to reconstruct the chart colors using the same method as generating the training data colors. The images were taken on a test bench, so we assume the illuminant is accurate, and we applied the proper illuminants without any LED error. The actual and estimated colors are shown in the same figures as the estimated curves. To calculate the color error between the actual and estimated colors, we used DeltaE 2000, a standard measurement of color error described in [8]. The relatively low DeltaE values of the reconstructed images, shown in Table 1, lead us to believe that these are good approximations of the actual sensitivity curves. We also calculated DeltaC for each estimate, which is the same as DeltaE, but neglects the luminance value in LAB color space. A low DeltaC means that there is good chromatic accuracy even if the two colors are off in terms of luminance. The DeltaC values are also shown in Table 1. We see that most of the DeltaC values are slightly smaller than the corresponding DeltaEs, but in the RYYCy case we see a large increase in accuracy. This means that, in this case, our estimated colors are accurate in terms of chromaticity but are generally a little high or low in luminance.

Sensor	Maximum DeltaE/C	Average DeltaE/C
iPhone	6.14 / 5.92	4.40 / 3.72
Moto One	6.26 / 5.65	2.72 / 1.93
Moto G7	5.68 / 5.49	2.56 / 1.79
RCCB	4.52 / 3.63	2.80 / 2.29
RYYCy	9.75 / 3.77	4.19 / 2.17

Table 1: DeltaE and DeltaC 2000 values from estimated curves

Comparison

The study by Tominaga et al [4] used a principal components technique for estimating the sensitivity curves of mobile phone cameras. They relied on the fact that all of the mobile phone cameras they measured had very similar shaped curves. Because the shapes of all curves were so similar, they were able to make good estimates using only the first principal component of the average curves. This estimation method only works for curves which have a similar shape to Figure 11. Our method expands on this by including RGB sensors with various curve shapes including the wide variety of curves represented in DSLR image sensors.

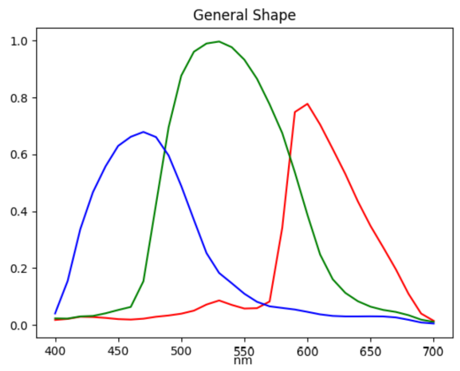

Figure 11: Average shape of all curves measured in [4]

Ji et al [2] also expanded the problem to a more general RGB curve shape, but using a compressive sensing approach. Rather than estimating the curves directly, they use a set of Gaussian basis functions and attempt to solve the equation:

$$C = R_{chart}^{T} * \Psi * s \tag{5}$$

Where C and R_{chart} are the same as above, Ψ is the basis functions, and s is a matrix of weights which, together with Ψ, generate the sensitivity curves. They achieved this by minimizing the following loss function using the CVX convex optimization algorithm in MATLAB:

$$\frac{1}{2}||C - \Phi * s||_2^2 + \gamma||s||_1 \tag{6}$$

Where $\Phi = R_{chart}^{T} * \Psi$, and γ is a constant weight. The l_2 norm represents a least squares minimization used to solve Equation 5, and the added l_1 norm encourages sparsity which helps the estimate follow a Gaussian shape by using as few basis functions as possible. This approach works well for estimating RGB curves that have a Gaussian shape to them, but when we attempt an estimation of the RYYCy filter in Figure 12a we get the curve in

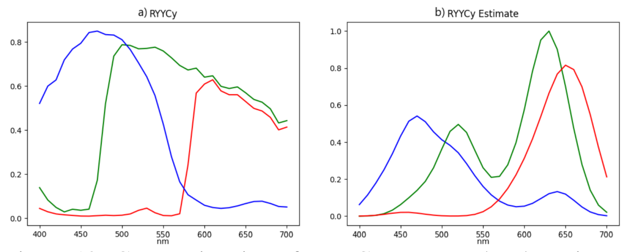

Figure 12: CVX estimation of RYYCy curves. The shape is not followed closely because it is not Gaussian, which is expected of most RGB filters.

Figure 12b. The non-gaussian shape of the curve makes the minimization problem much harder because we are not able to rely on Gaussian basis functions. Instead, we use the power of the large number of weights and biases in even our small neural network to estimate the curves directly.

Another machine learning approach to this estimation problem was proposed by He et al [3]. They used a confidence voting convolutional neural network which included 6 convolutional layers, 5 pooling layers, and a confidence voting layer. The network would take in an entire image and output estimates of the sensor curves using various basis functions. Their network is much more complex than ours and again only deals with standard RGB color filters. In comparison, we simplified the network to our relatively small fully connected neural network that only takes in 24 tri-chromatic values rather than an entire image. At the same time we expanded the problem by including non-RGB filters in our training data to allow estimation on a broader range of sensors.

Conclusion

This paper describes a method to estimate the sensitivity curves of a camera using a single demosaiced image of a color chart with known reflectances. Our method directly estimates the curves without relying on basis functions, and does not require knowledge of the scene illumination. Our method expands on previous work by being able to estimate the sensitivity curves of a camera without assuming a RGB color filter array. We introduce examples of RCCB and RYYCy color filters which have been shown to have benefits in the automotive and security camera fields. The addition of new types of color filters made this a more complicated problem because there are very large variations in the shapes of curves when comparing two different filters. This means we could not rely on principal components with simple shapes as in [4] and we had to make sure the network was not biased toward any particular shape. The results of this project are promising for the development of a simple estimation method that generalizes to any color filter, and requires nothing more than a raw image of a color chart under some unknown illumination.

References

[1] Eiichi Funatsu, Steve Wang, Jken Vui Kok, Lou Lu, Fred Chen, Mario Heid. “Non-RGB Color Filter Options and Traffic Signal Detection Capabilities.” Electronic Imaging, vol. 34, no. 16, 2022, pp. 215–1. Crossref, https://doi.org/10.2352/ei.2022.34.16.avm-215.

[2] Yuhyun Ji, Yunsang Kwak, Sang Mok Park, Young L. Kim. “Compressive Recovery of Smartphone RGB Spectral Sensitivity Functions.” Optics Express, vol. 29, no. 8, 2021, p. 11947. Crossref, https://doi.org/10.1364/oe.420069.

[3] Tianyue He, Qican Zhang, Mingwei Zhou, Junfei Shen. “CVNet: Confidence Voting Convolutional Neural Network for Camera Spectral Sensitivity Estimation.” Optics Express, vol. 29, no. 13, 2021, p. 19655. Crossref, https://doi.org/10.1364/oe.425988.

[4] Shoji Tominaga, Shogo Nishi, Ryo Ohtera. “Measurement and Estimation of Spectral Sensitivity Functions for Mobile Phone Cameras.” Sensors, vol. 21, no. 15, 2021, p. 4985. Crossref, https://doi.org/10.3390/s21154985.

[5] Thomas Mansencal, Michael Mauderer, colour-science python package. https://github.com/colour-science/colour

[6] Johanne Roby, LSPDD: Lamp Spectral Power Distribution Database. https://lspdd.org/app/en/home

[7] Jinwei Gu, Camera Spectral Sensitivity Database. https://www.gujinwei.org/research/camspec/db.html

[8] Bruce Justin Lindbloom, Useful Color Information, Studies and Files. http://www.brucelindbloom.com/Eqn_DeltaE_CIE2000.html

[9] Alex Lenail, NN-SVG. http://alexlenail.me/NN-SVG/index.html

Author Biography

Abraham Sachs is a senior at UC Davis completing a BS in electrical engineering with minors in music and mathematics, graduating in June of 2023. He plans to complete a PhD in electrical engineering. He was an intern at Omnivision during the summer of 2022 where he researched color processing.

Ramakrishna Kakarala is Director of Algorithm Development at Omnivision. He received his PhD in Mathematics from the University of California, Irvine, in 1992.