477 HW2

Aditya Jadhav

September 7, 2023

1 Abstract

In this paper, we study camera spectral sensitivities and their calibration. Utilizing a 101x3 matrix that represents the spectral sensitivities of a particular camera, we begin by plotting these sensitivities in MATLAB. We then generate 1600 random light spectra and compute their (R, G, B) responses using these sensitivities. Moreover, we simulate measurement errors by adding Gaussian noise to the RGB values, thereby mimicking a realistic camera calibration experiment. The paper also explores the estimation of these sensitivities through the least squares method and evaluates its performance using Root Mean Square (RMS) error. The results are further tested under varying noise levels and the introduction of data clipping into [0,255]. Finally, we touch upon the concept of Gamma Calibration, where the gamma value is calculated based on the gray value observed

2 Introduction

This paper aims to delve deeply into slightly specialized functionalities of MATLAB, specifically focusing on aspects of image processing. The primary subject of our investigation is a 101 by 3 matrix, representing spectral sensitivities of a camera sensor. We investigate how the camera responds to different light wavelengths, ranging from 380 to 780 nanometers at 4nm intervals.

Initially, we plot the given sensor data for each of the RGB sensitivity values. Following this, we will generate 1600 random light spectra which we will use to generate a hypothetical image. We will use these generated spectra to compute the RGB responses, while also paying close attention to various scale factors and units involved in the process. To visualize these findings, we will construct a 400x400 color image, made up of 1600 10x10 blocks of uniform RGB for further visual findings.

Additionally, we will estimate the sensor sensitivities through least squares methods, and evaluate the accuracy of these estimations using RMS error calculations. Moreover, the effects of Gaussian noise and scaling factors will be considered to assess the accuracy/error of our sensor sensitivity calculation techniques.

We lastly take a look at gamma calibration in displays using a hypothetical experiment and compute the gamma for the hypothetical screen.

We will use MATLAB to execute all mathematical computations and data visualizations. This report is compiled using LaTeX in OverLeaf, employing packages such as amsmath for mathematical equations, graphicx for images, subcaption for image captioning, and tabularx for tables. It took approximately 5 hours to conduct the experiments and a further 3 hours to write the report making it a whopping 8 hours of work.

3 Experiments

3.1 Playing with Camera Spectral Sensitivities

Thanks to the instructor, we have a sample txt file containing 101 by 3 matrix representing estimates for the spectral sensitivities of a camera. We start by plotting the values of these senstivities so we have a visual reference of the values.

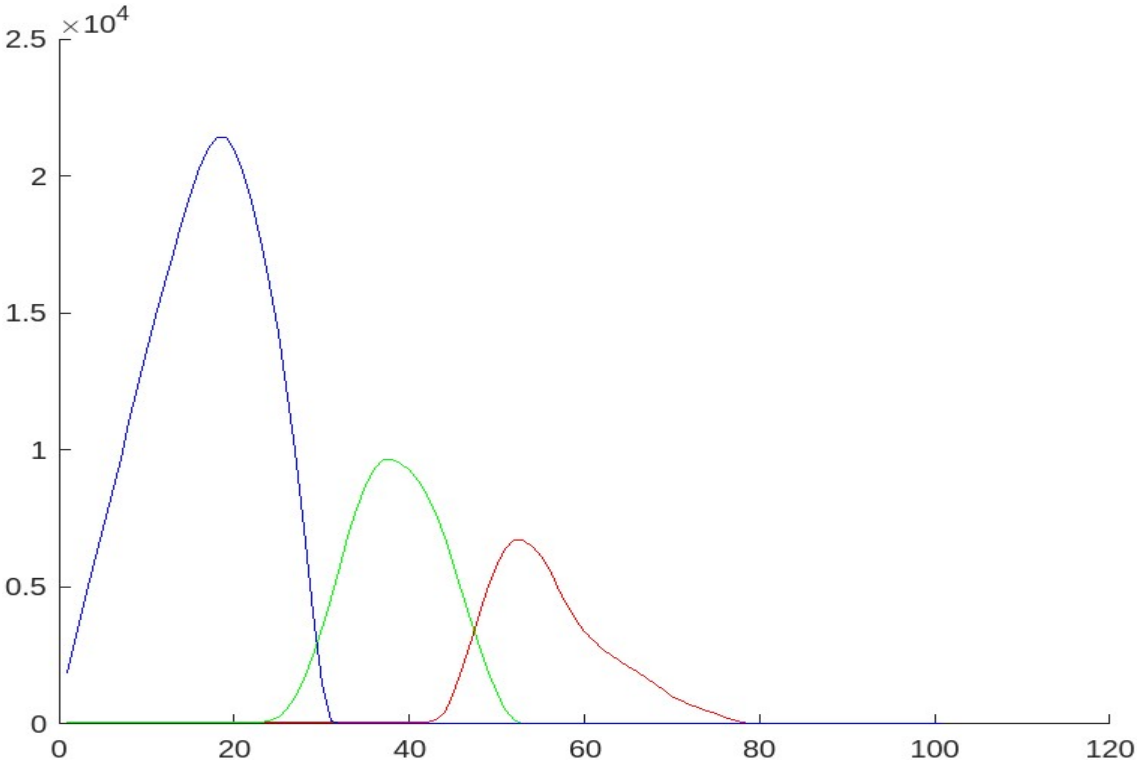

Figure 1: Plot of the spectral sensitivity values for RGB. The line for the sensitivities for R, G and B is color coded.

Then we generate 1600 random light spectras as 101 element row vectors by setting rng as 477 and then calling rand to get the values. We then compute the output image using the generated spectra and the given light sensitivites. We scale the output values by dividing by 1061 to get values in the 8 bit range. To make it visually easier to discern, we create a 400 by 400 color image made from 1600 10x10 blocks of uniform RGB. The image we get is shown below.

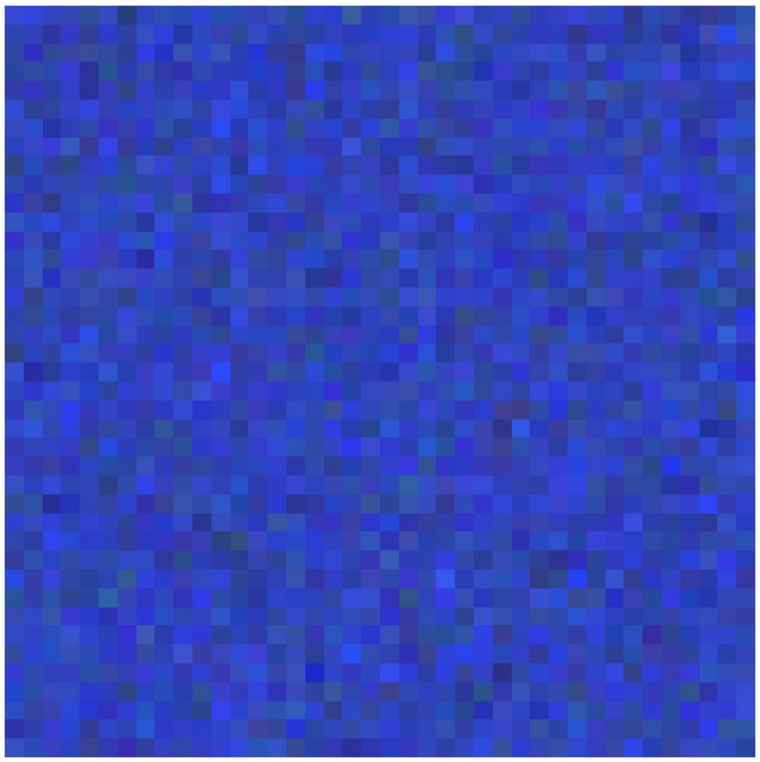

Figure 2: 400 by 400 color image made from 1600 10x10 blocks containing values we get from the generated spectra and given spectral sensitivity values for a camera. Note the blue shift of the image. The bias is because the camera balance is for light that is biased away from blue, as is the case for indoor lighting.

3.2 Estimating spectral sensitivities from output image and error checking

We have the output value from the previous section. Now we try to do the inverse of what we did. We try to estimate the sensitivities from the spectral values and output RGB values using the formula

$$x = (U^T U)^{-1} U^T y$$

We get the following plot of the sensitivity values after estimation using the above formula.

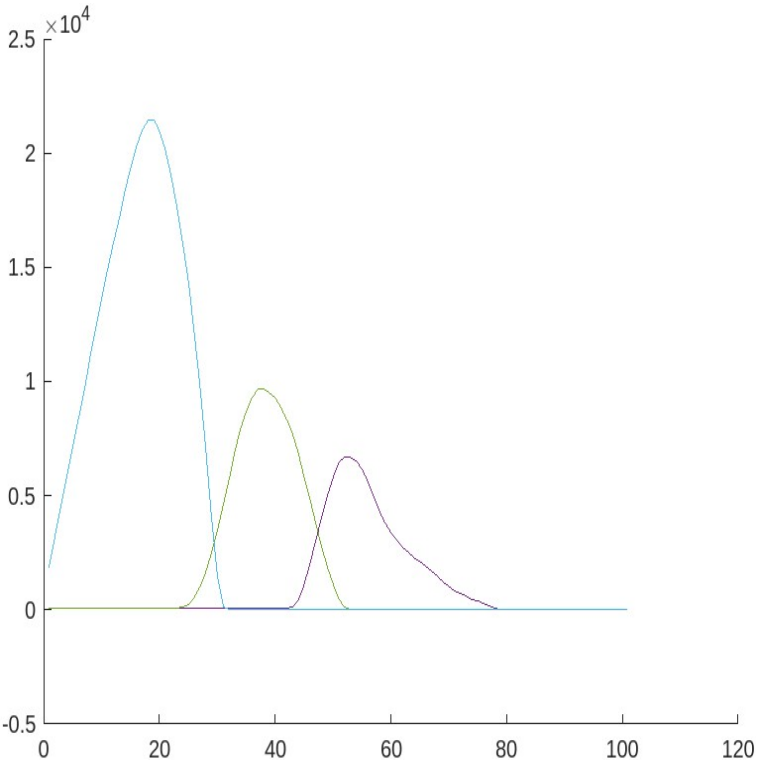

Figure 3: Plot of the estimated spectral sensitivity values for RGB. The line for Red values is the largest in blue, the one for Green is in green and red is in purple colour (decreasing peaks respectively).

We calculate the sensitivity values, then check the accuracy of the returned value, we get RMS error between the calculated R, G and B sensitivity values and actual R, G and B values. The table gives the error for each color.

Color	RMS Error ($\times 10^{-10}$)
R	1.2135
G	1.1903
B	4.9363

Table 1: RMS Error between Red, Green, and Blue Actual Sensitivity Values and Calculated Values.

It is also prudent to get the error between the output image from the actual sensitivity values and the calculated sensitivity values. We find out RMS error

between each of the output image values from both. This error is a measure of how well our estimated sensors can reconstruct the RGB.

Color	RMS Error ($\times 10^{-13}$)
R	3.0925
G	3.0333
B	1.2299

Table 2: RMS Errors for R, G, and B for Output Image from Calculated and Actual Sensitivity Values

3.3 Impact of Noise on the experiment

We now measure the impact of noise (which can be very common during practical scenarios) on our experiments till now. We check the impact on the output image, how it affects estimating sensitivity values from spectral values and the output image from the sensors. We begin by adding gaussian noise to each of the R, G and B values we get from the original sensitivity values and the spectral images. We set the standard deviation to be 10. We try to estimate the sensitivity values again from the noisy output image we have generated. We plot them below to get a visual understanding again.

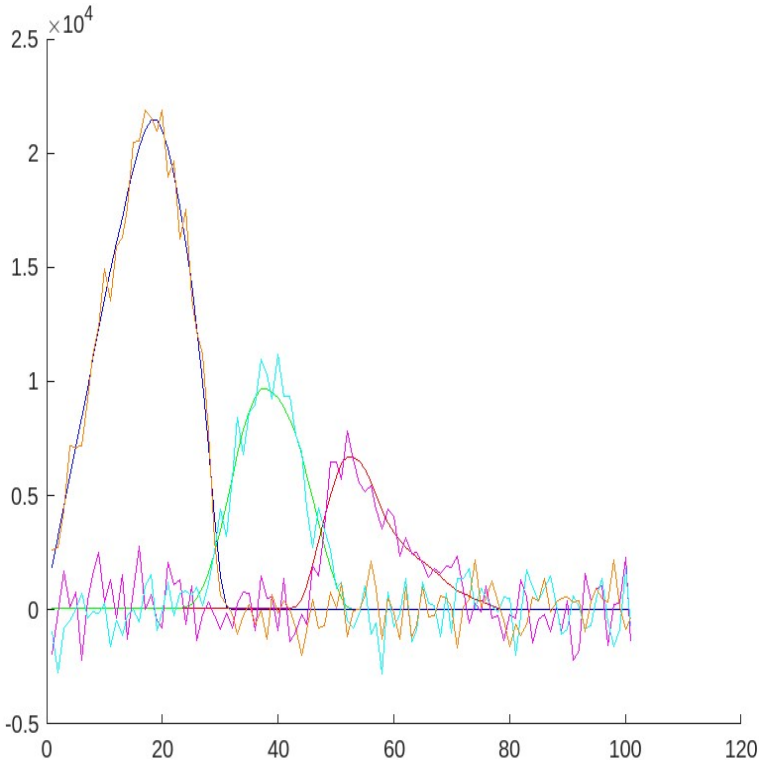

Figure 4: Comparison of Estimated Spectral Sensitivities: This plot shows the actual spectral sensitivities of the camera for Red (R), Green (G), and Blue (B) as curves in red, green, and blue respectively. Alongside, the estimated sensitivities with noise are displayed in magenta (R'), cyan (G'), and orange (B').

As we add more noise to the data, we will start to get lots of negative values and values beyond 255. This might not be the simulation we are interested in, as real data from sensors will be "clipped" to be in the range of [0,255]. So we estimate the spectral sensitivity values again after clipping the values of the output image to be between [0,255]. We use the min and max functions to achieve this. We get the following plot for the estimate.

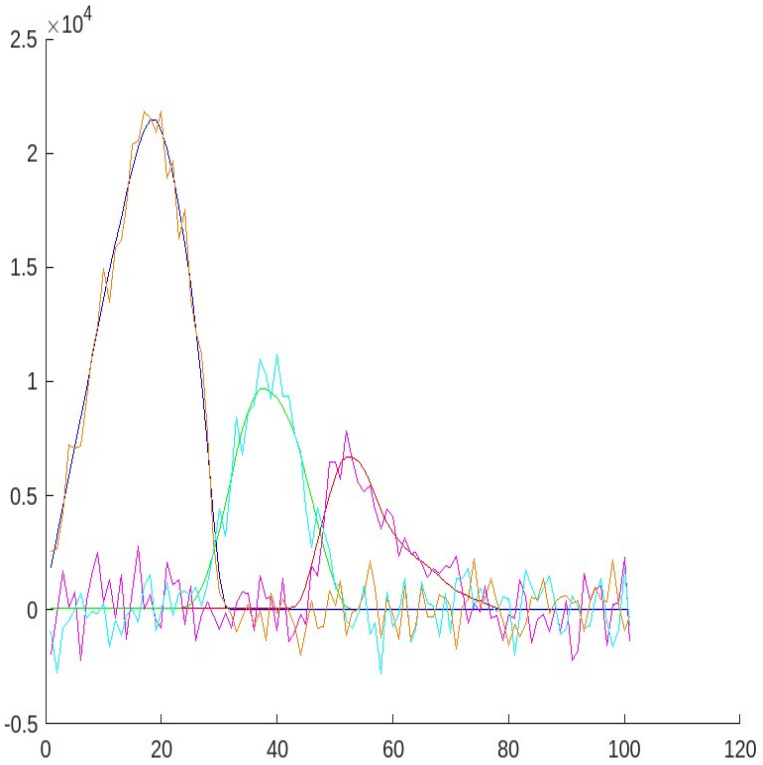

Figure 5: Comparison of Estimated Spectral Sensitivities after clipping the output image we use for estimation in the formula to be between [0,255]: This plot shows the actual spectral sensitivities of the camera for Red (R), Green (G), and Blue (B) as curves in red, green, and blue respectively. Alongside, the estimated sensitivities with noise are displayed in magenta (R'), cyan (G'), and orange (B').

We note that much hasn't changed after clipping the output image suggesting the output image contained only values between [0,255]. With this caution behind, we again compute the RMS error between each of the three estimated sensors and the actual ones.

Table 3: Comparison of RGB and Sensor Errors at Different Noise Levels. We make a division between clipping the output image values between [0,255] and not clipping them.

NoiseStdDev	RGB_Error_NoClip	RGB_Error_Clip	Sensor_Error_NoClip	Sensor_Error_Clip
10	9.9633	9.9549	902.82	900.71
20	20.023	19.899	1898.4	1893.5
30	30.085	29.233	2833.4	2749.8
40	39.325	37.017	3910.9	3711.3
50	49.568	45.215	4499.5	4128.6
60	59.766	52.227	5626.9	4906.6
70	70.095	58.812	6390.9	5409.3
80	81.098	66.2	7696.4	6146.2
90	89.314	70.148	8519.5	6642.9
100	100.72	77.288	8813	6632.3
120	119.57	85.658	10566	8117.5
140	140.89	93.534	12868	8368.2
160	158.02	98.124	14774	8861.9
180	181.41	105.51	18085	9882.7
200	196.62	105.59	19391	10593
250	248.45	114.71	23957	10857
300	299.96	119.68	26764	10651
350	349.15	123.59	34839	11913
400	399.27	124.76	36791	11660

We see as the noise level increases, so does the overall RMS error for both RGB and sensor estimation values. We find that clipping the values greatly reduces the RMS error as the noise levels increase. This suggests that their is a greater tendency for values to exceed bounds when noise level is high, which is to be expected. Clipping them keeps them under bounds and leads to much lower RMS error (almost 1/3 error in the sensor estimation at 400 stdev). At higher noise levels, the RMS error must lead to almost incomprehensible image values. Drawing the plots for every noise value becomes unfeasible so we draw estimated sensor values for the 5th and 10th iteration which is standard deviation 50 and 100 respectively.

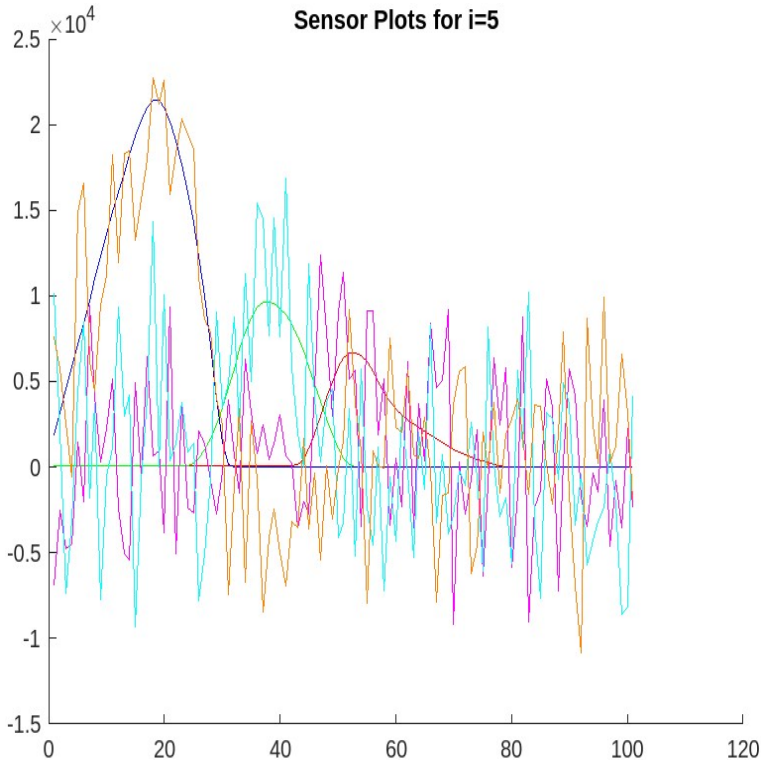

Figure 6: Comparison of Estimated Spectral Sensitivities with stdev as 50 after clipping the output image we use for estimation in the formula to be between [0,255]: This plot shows the actual spectral sensitivities of the camera for Red (R), Green (G), and Blue (B) as curves in red, green, and blue respectively. Alongside, the estimated sensitivities with noise are displayed in magenta (R'), cyan (G'), and orange (B').

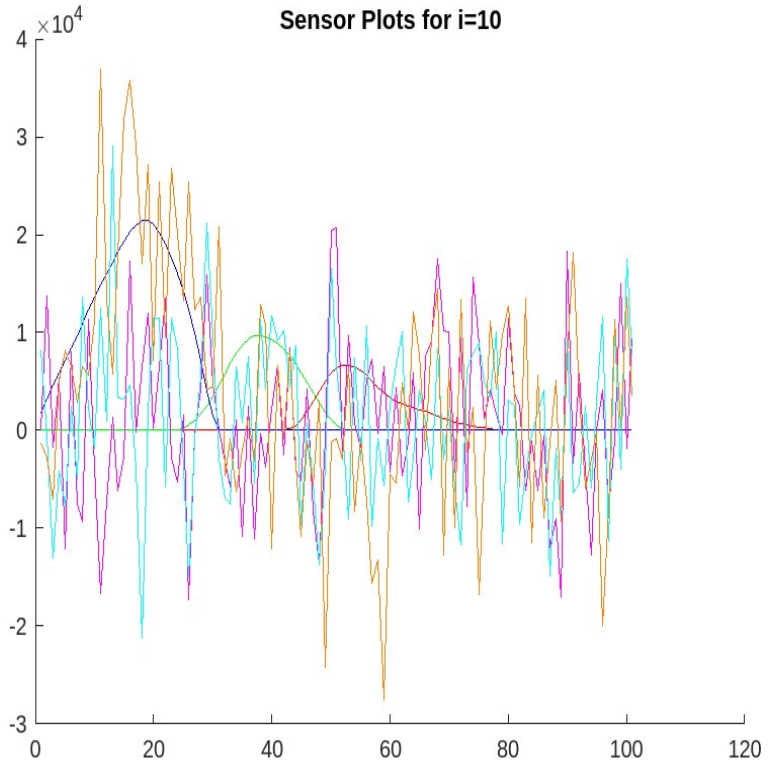

Figure 7: Comparison of Estimated Spectral Sensitivities with stdev as 100 after clipping the output image we use for estimation in the formula to be between [0,255]: This plot shows the actual spectral sensitivities of the camera for Red (R), Green (G), and Blue (B) as curves in red, green, and blue respectively. Alongside, the estimated sensitivities with noise are displayed in magenta (R'), cyan (G'), and orange (B').

We see that the sensor values go very far from the actual values in the plot as the stdev value increases. These are unusable in practical applications but these noise levels are seldom noticed in real life.

3.4 Gamma Shifting

Now we setup a hypothetical experiment to learn about gamma calibration in modern displays. The idea is that when the gray apple matches the stripes viewed at a distance, this is when the raw intensity of the apple is $\frac{1}{2}$ between black and white. Suppose that when the user matched the gray apple with the stripes viewed at distance, the gray value was 80 on a scale from 0 to 255. Assuming that this non-linearity can be fixed with a gamma correction, we can compute the value of gamma. We are interested in finding the value of the gamma (γ) which can correct this non-linearity.

The standard gamma correction formula is:

$$I_{\text{out}} = 255 \times \left(\frac{I_{\text{in}}}{255}\right)^{\gamma}$$

where I_{out} is the output intensity and I_{in} is the input intensity.

In our case, $I_{\text{in}} = 127.5$ and $I_{\text{out}} = 80$ (since it should be halfway between 0 and 255 for a perfect gray).

Rearranging the formula to solve for γ, we get:

$$\gamma = \log_{\left(\frac{I_{\text{in}}}{255}\right)} \left(\frac{I_{\text{out}}}{255}\right)$$

Substituting $I_{\text{in}} = 127.5$ and $I_{\text{out}} = 80$, we find that:

$$\gamma = \log_{\left(\frac{127.5}{255}\right)} \left(\frac{80}{255}\right)$$

After calculating, we find that $\gamma \approx 1.67$.

4 Conclusion

We now have succesfully discussed and experimented many MATLAB functions for image processing, used linear algebra to compute output values from spectral values. Estimated sensor sensitivity values from output image using inversion. We have learnt about how colors are important in an image's representation. We calculate errors as noise is introduced to data and how clipping works combating it a bit. We also have verified how gamma correction works.

5 Bibliography

- https://www.overleaf.com/learn/latex/Matrices
- https://www.overleaf.com/learn/latex/Inserting_Images
- `https://www.overleaf.com/learn/latex/How_to_Write_a_Thesis_in_LaTeX_(Part_3)`
- https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes
- matlab.mathworks.com