Google Pixel 4a Color Calibration

(**Teacher's Note:** The dimensionality of the spectral radiance used as inputs is lower than we would like for a great calibration. The MCC, even with the additional illuminants, doesn't produce enough of a range to get a great estimate of the sensor QE. The mathematical methods used here help. We might consider whether we could improve the test chart to obtain a better sensor QE estimate. Or a test chart plus some more variations in the illumination.)

Team members: Claire Zhang, Megan Zhang

Contents:

I. Introduction

An imaging device takes incident photons as input and converts them to electrons. In reality, not all photons received can be converted to electrons, resulting in the loss of information. Spectral quantum efficiency (spectral QE) measures the sensitivity of an imaging device in the percentage of photons landed versus photons converted to electrons. Since spectral QE is a ratio, it is dimensionless. The higher the spectral QE value, the more effective the imaging device is in retaining and converting the light information downstream. Spectral QE is measured over a range of wavelengths to characterize the device's efficiency for photons of different energy levels.

Spectral QE is highly dependent on the sensor structure and materials. But if we have a model of an imaging device's spectral QE, we will be able to approximate the device's digital output given light radiance input. For a sensor with RGB channels, the relationship between RGB digital output and light radiance input can be characterized by the equation below:

$$RGB = spectralQE * radiance$$

The objective of this project is to create a spectral QE model for the sensors in Google Pixel 4 camera from the spectral radiance input values and digital camera output values.

The camera sensor at the back of Google Pixel 4a is Sony IMX363 [1]. Since the specifications of this particular type of sensor are available to us, it can be used as the golden model. Figure 1 shows Sony's approximation of the Google Pixel 4a camera's spectral QE curves for RGB channels.

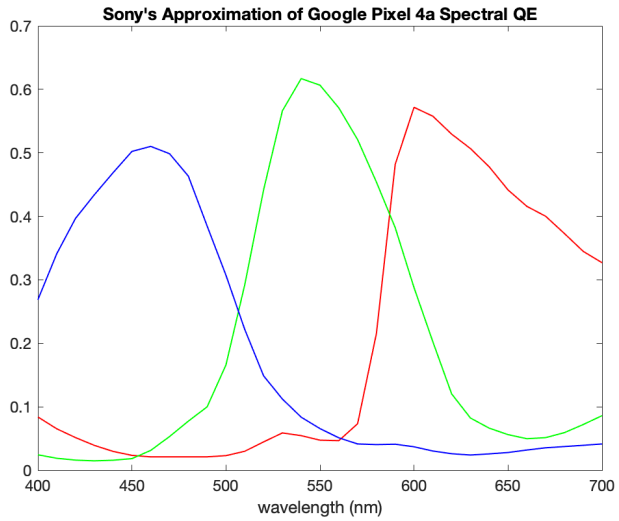

Figure 1. Approximation of Sony IMX363 (sensors of the rear-facing camera on Google Pixel 4a) Spectral QE

II. Data

For the project, we are provided with input (spectral radiance measurements) and output (sensor images) for three illuminants on the 24 patch Macbeth Color Checker (MCC). The three illuminants are tungsten light (denoted as A), cool white fluorescent (denoted as CWF), and daylight (denoted as Day).

1. Radiance (Input)

The radiance measurements are made from Photoresearch spectrophotometer model PR670. We are provided with these data in .mat files, which contain 24 lines that correspond to the 24 color patches on MCC for each illuminant. The radiance curves are shown in Figure 2.

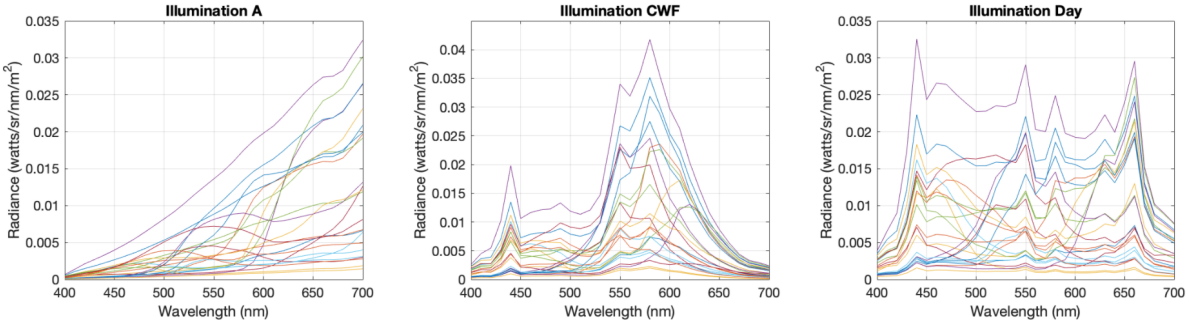

Figure 2. Radiance for three illuminants.

2. Sensor Response (Output)

The sensor images are provided in digital negative files (DNG) from the Google Pixel 4a camera. The photos are taken when MCC is in the middle of the frame and illuminated by each of the three spectral lights. The three files are set to the same ISO speed.

III. Methods

In this section, we described the methods that we use throughout this project. The subsections are organized based on the order of the project stages.

1. Problem Formulation

After loading and pre-processing the input/output data, we want to find the spectral QE that best connects them. We translate this into a linear system:

$$predictedRGB = radiance^T * spectralQE$$

where the dimension of the three matrices are RGB: 24x3, radiance: 31x24, and spectral QE: 31x3. Here 24 comes from the 24 color patches of MCC. 31 is the dimension of the radiance data provided. Our goal is to minimize the difference between predicted RGB and ground truth RGB.

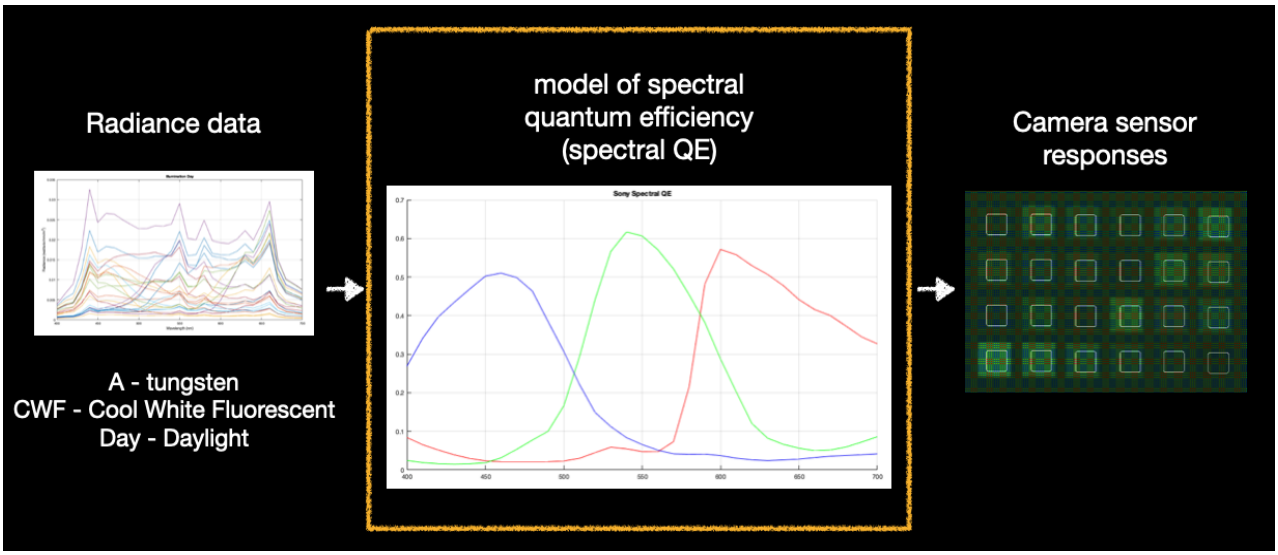

Figure 3. input and output of the spectral QE model

2. Data Pre-processing

Before we could get the sensor RGB values, we needed to go through several preprocessing steps, with the given file and final sensor reading illustration shown in Figure 4.

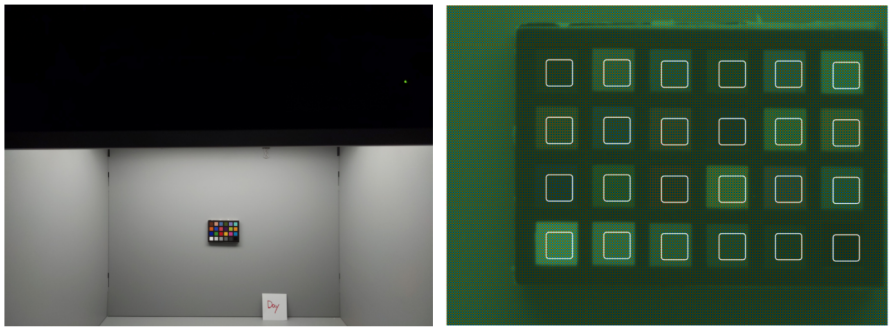

Figure 4. Left: original camera photo. Right: sensor reading.

2.1 Cropping

After creating a sensor in ISETCam and reading in the DNG files, we first cropped out the central region where the MCC is and selected the four corner points to locate the 24 patches.

2.2 Black Level and Exposure Time Correction

Also, we got the values for the black border. After extracting the sensor values, we needed to correct those RGB values for black level and exposure time. The reasoning for the need for black level correction is that we have to make sure the black borders in the images have R, G, and B equal to zero. Therefore, we subtracted the black level of the corresponding illuminant to achieve that. The varying exposure time is also something we had to take account into. Since our goal is to generate the linear values of RGB in the units of digital value per second, the way to correct for exposure is to divide the original values by exposure time.

$$converted RGB = \frac{raw RGB - black\ border}{exposure\ time}$$

Signal-to-noise ratio (SNR) is a common metric that is used to compare the level of the desired signal to that of the background noise. They are often expressed in the units of dB with the formula $SNR = 20\log_{10}(\text{Mean}/\text{Variance})$. We looked at the SNR for the RGB values. For all three illuminants, the peaks are all around 20dB or above, suggesting good signal qualities.

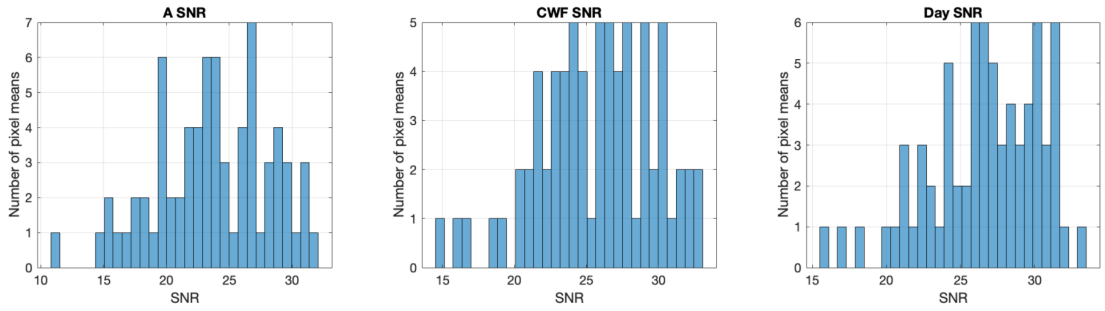

Figure 5. SNR for sensor data.

2.3 Cross-Validation

We apply cross validation in two ways: 1) use illuminant A data only as training set, and the rest data as testing set 2) combine all illuminant's data and randomly select 70% as training set, and the rest 30% as testing set. Using cross validation allows us to evaluate our model's performance on unseen data.

3. Fitting Methods

There are two main mathematical concepts involved in the fitting methods that we used to find the model, least-squared and singular value decomposition.

2.1 Least-Square and Pseudo-Inverse

Formally, our problem is

$$\begin{aligned} &\boldsymbol{minimize}\ ||\boldsymbol{A} * \boldsymbol{x} - \boldsymbol{y}|| \\ &\boldsymbol{subject\ to}\ \boldsymbol{x_i} \geq \boldsymbol{0} \end{aligned}$$

, with A being the transpose of radiance matrix, x is spectral QE for R, G, or B channel, and y is the ground truth R, G, or B values. In practice, we simplified the problem by leaving out the non-negativity constraint. The reasoning behind is that we did implement with the constraint using MATLAB function lsnonneg(), supported by [2], and the result was not good for full-rank A and was not converging for non-full-rank $\tilde{\boldsymbol{A}}$ after SVD (described below). So we decided to focus on the simple least square problem of minimizing the norm of the difference between predicted and ground truth RGB.

For the simplified least-square problem $\boldsymbol{minimize}||\boldsymbol{A} * \boldsymbol{x} - \boldsymbol{y}||$, if $\boldsymbol{A} \in \mathbb{R}^{\boldsymbol{mxn}}$ is full-rank and $\boldsymbol{m} > \boldsymbol{n}$, we could formulate a pseudo-inverse $\boldsymbol{A}^{\dagger}$ such that $\hat{\boldsymbol{x}} = \boldsymbol{A}^{\dagger} * \boldsymbol{y} \approx \boldsymbol{x}$. The formula for $\boldsymbol{A}^{\dagger}$ is

$$\boldsymbol{A}^{\dagger} = (\boldsymbol{A}^{\boldsymbol{T}} * \boldsymbol{A})^{-1} * \boldsymbol{A}^{\boldsymbol{T}}$$

, or we can use MATLAB built-in function, $\boldsymbol{pinv()}$.

2.2 Low-Rank Approximation via Singular Value Decomposition

When we naively solved the least-square problem to approximate the spectral QE matrix, we observed that the resulting model is overfitting due to high dimensionality. Therefore, we decided to tackle the overfitting issue by restricting the degree of freedom in the system matrix. Concretely, we used a low rank approximation of A matrix (radiance') via singular value decomposition (SVD) instead of the full-rank A in the least-square solver.

$$\boldsymbol{A} = \boldsymbol{U} * \boldsymbol{\Sigma} * \boldsymbol{V}^{\boldsymbol{T}} = [\boldsymbol{u_1} \dots \boldsymbol{u_r}] * \boldsymbol{diag}(\boldsymbol{\sigma_1} \dots \boldsymbol{\sigma_r}) * [\boldsymbol{v_1} \dots \boldsymbol{v_r}]^{\boldsymbol{T}}$$

Mathematically, when we apply system A to an input x, we are essentially doing three steps: 1) map x to the direction of right singular vector $\boldsymbol{v_i}$, 2) scale it with corresponding singular value $\boldsymbol{\sigma_i}$, 3) reconstitute along left singular vector $\boldsymbol{u_i}$. SVD breaks down this procedure. The sorted singular values provide information on how many and which singular value and vector pairs capture the most important information about the original system A. We can select the k < r (r is rank) first singular value and singular vector pairs to reconstitute a new system matrix $\tilde{\boldsymbol{A}}$ that approximates the original A.

When we used A radiance and RGB as the training set, the singular values for the radiance matrix are shown in Figure 5. The magnitude of the singular values dropped quickly at the beginning, with an 80% decrease at the fourth singular value. The later singular values are really small. So we picked the first four pairs of singular value and singular vectors to reconstitute the radiance A matrix. We then feed this newly formed matrix to pseudo-inverse to solve for spectral QE.

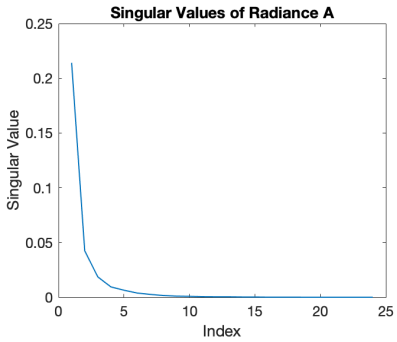

Figure 5. Singular Values of Radiance A.

When we used 70% of all radiance and RGB as the training set, the singular values for the radiance matrix are shown in Figure 6. We picked the first four pairs of singular value and singular vectors to reconstitute the training radiance matrix to feed into pseudo-inverse to solve for spectral QE.

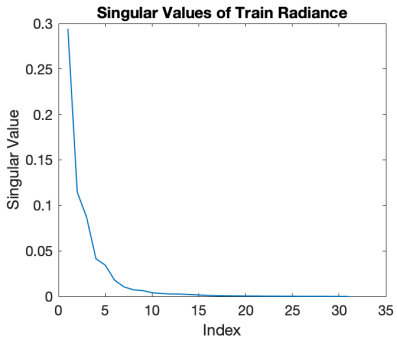

Figure 6. Singular Values of Train Radiance.

4. Data Post-Processing

To ensure that we evaluate the metrics at the right scale, there are two important post-processing steps we need to go through.

First, in order to calculate the correct ΔE value, we need to convert the predicted RGB back to the scale of the original sensor RGB. Due to exposure time being really small and on the scale of 10^{-3}, the RGB values converted from sensor RGB using the pre-processing steps are now on the order of 10^{4}. Therefore, we revert the pre-processing step to calculate the RGB to be evaluated (RGB2evaluate) as follows:

$$RGB2evaluate = predictedRGB * exposure\,time + black\,border$$

Second, in order to have a fair comparison between our spectral QE model and Sony's model on the same scale, we need to normalize our spectral QE values. For the spectral QE curve of each channel, we normalize it against the maximum value of Sony's model for the same channel. So the three spectral QE curves have the same peak amplitudes as those of Sony's model.

5. Model Evaluation Methods

To thoroughly analyze the performance of our model, we applied three different methods - (1) plotting the spectral QE curves, (2) directly comparing the predicted RGB values with measured RGB values, and (3) calculating the ΔE between predictions and measurements using the color metrics (CIELAB). Each of the evaluation metrics has its own advantages and focus when it comes to model evaluation. Therefore, applying those three methods in our model evaluation stage gave us a more comprehensive understanding of the performance of each of the models.

The incentives behind applying each of the evaluation methods are as follows:

5.1 Spectral QE Curves

We used the IMX363 sensor specifications from Sony as a reference. Comparing our spectral QE curves with the ones from Sony gives us a clear picture of whether our model deviates from the golden model.

5.2 Measured vs. Predicted RGBs

In this method, we made a direct comparison between the model predictions and the actual measurements. This method offers an intuitive way to understand how our models perform, in terms of the numerical difference between predictions and measurements.

5.3 CIELAB ΔE

The ΔE term in the CIELAB metric offers a better way to quantify color difference visibility than the Euclidean distance. Therefore, by calculating the ΔE between the predicted RGB (after post processing) and measured RGB, we can have a more solid understanding of how the predicted MCC chart is perceptually different from or similar to the actual MCC chart. The higher the value, the easier it is to visually distinguish two colors. After obtaining the RGB2evaluate values, we converted RGB to XYZ, and then converted XYZ to La*b*. We calculated the ΔE between predicted and ground truth La*b*. We used MATLAB functions $rgb2xyz()$ and $xyz2lab()$ to conduct the two conversion steps. For the second conversion step, we used MATLAB's default reference white light (CIE standard illuminant D65).

$$[L, a^*, b^*] = xyz2lab\left(rgb2xyz\left(RGB\right)\right)$$

$$\Delta E = \sqrt{(L1 - L2)^2 + (a_1^* - a_2^*)^2 + (b_1^* - b_2^*)^2}$$

Assuming that ΔE is within the range [0,100], we can interpret the ΔE values in the following way [2]:

ΔE	Perception
<= 1.0	Not perceptible by human eyes
1-2	Perceptible through close observation
2-10	Perceptible at a glance
11-49	Colors are more similar than opposite
100	Colors are the exact opposite

IV. Results

Next, let's evaluate the results with the three methods discussed in the previous section:

1. Pure Pseudo-inverse

Generally speaking, pseudo-inverse generates an inverse mapping from RGB to radiance by trying to fit all available sample points without considering the model dimensionality. Therefore, naive pseudo-inverse can lead to a high-dimensional model.

1.1 Training Set: A only

1.1.1 Spectral QE Curves: The QE curves for R, G, and B do not have a clear pattern and are very far from the Sony specs data. Each of the R, G, and B curves looks noisy (i.e., contains high-frequency components). This is caused by the model overfitting on high-dimensional data.

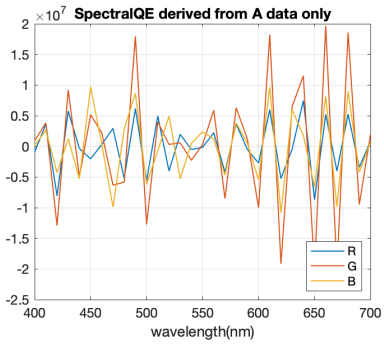

Figure 7. Spectral QE derived from A data only using pure pseudo-inverse.

1.1.2 Measured vs. Predicted RGBs: Since we only used A data for training, the model fits really well for A. However, for the two illuminants that did not play a role in training (i.e., CWF and Day), the model did not show good results. Overall, the model has high variance because it only pays attention to dataset A and does not generalize well for the other two sets.

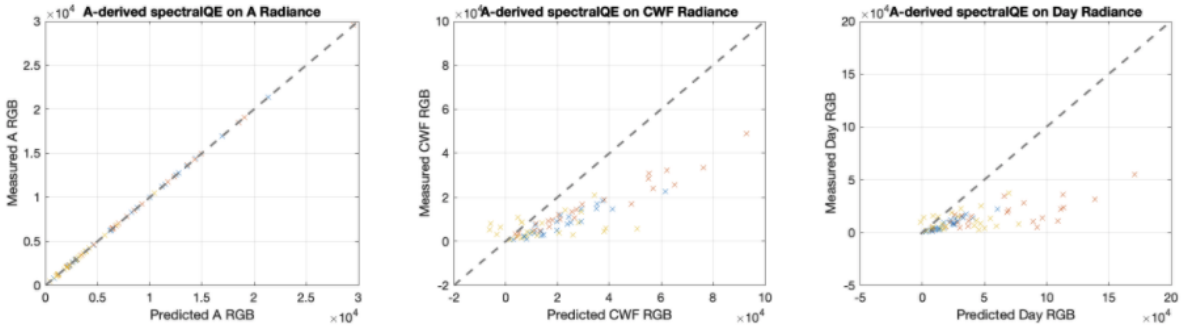

Figure 8. Measured vs predicted RGB with A-derived spectral QE.

1.2 Training Set: A, CWF, and Day (with Cross Validation)

1.2.1 Spectral QE Curves: The QE curves for R, G, and B do not have a clear pattern and are very far from the Sony specs data. This is caused by the model overfitting on high-dimensional data.

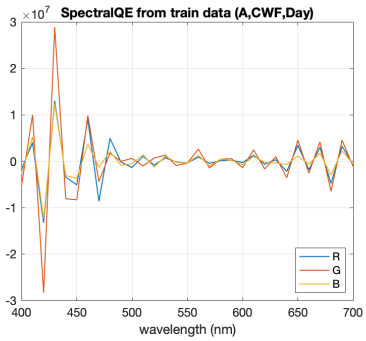

Figure 9. Spectral QE derived from train data using pure pseudo-inverse.

1.2.2 Measured vs. Predicted RGBs: The leftmost plot shows the measured and predicted RGB values in the test set. It shows that the model has good performance on the test set in terms of numerical differences between measured and predicted RGB values. Also, we can tell from the other three plots that the model generalizes well for all three illuminants.

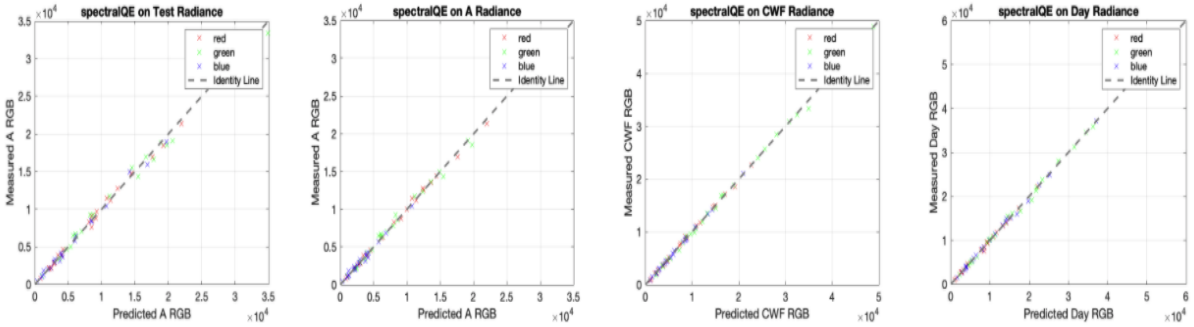

Figure 10. Measured vs predicted RGB with train data derived spectral QE.

1.2.3 ΔE of CIELAB: With the interpretation of the ΔE values provided in [2], the predicted color values are not perceptually differentiable for radiance in set A, CWF, and Day. The predicted RGB values of the test set are perceptible through close observation, but the colors are very similar to the measured ones.

Test	A	CWF	Day
2.2979	1.5384	0.82624	0.76533

2. SVD + Pseudoinverse

2.1 Training Set: A only

2.1.1 Spectral QE Curves: The model (dashed lines) has a relatively good approximation of the Sony QE curves (solid lines) - the R, G, and B curves have the same shapes and their peaks are centered at approximately the same wavelengths as the Sony QE curves. Note that the crossings between the R and B/G are also very close to the golden model. One problem with this model is that it is not physically plausible due to the negative QE values of R, G, and B at some wavelength bands.

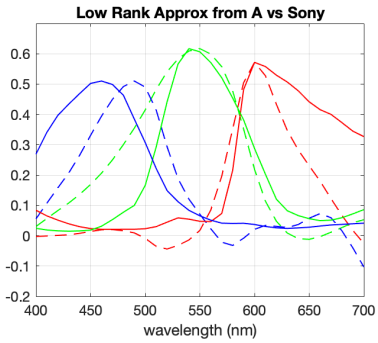

Figure 11. Spectral QE derived from low-rank approximation of A data.

2.1.2 Measured vs. Predicted RGBs: This model also has good performance if we use numerical differences between measured and predicted RGB values as our main evaluation metric. It also generalizes well for all three illuminants.

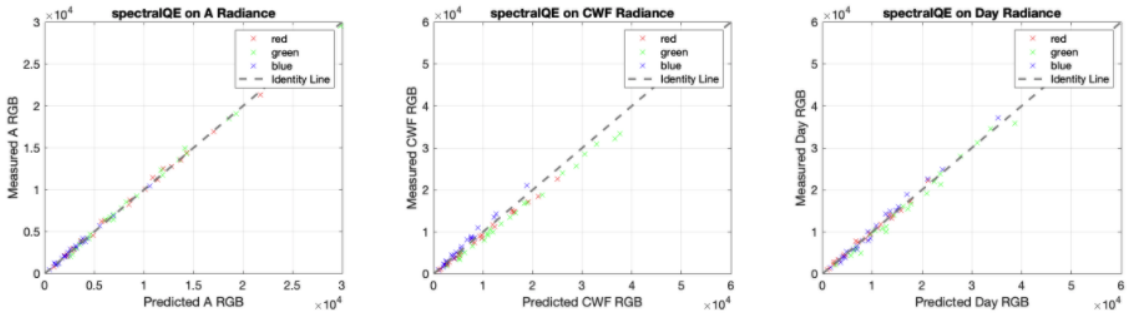

Figure 12. Measured vs predicted RGB with train data derived spectral QE.

2.2 Training Set: A, CWF, and Day (with Cross Validation)

2.2.1 Spectral QE Curves: After incorporating the cross-validation method into this method, our model has a decent approximation of the Sony QE curves. The issue with the negative R, G, and B QEs remains in this model.

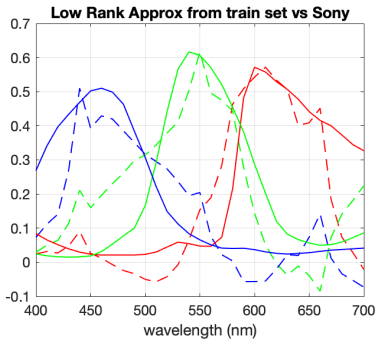

Figure 13. Spectral QE derived from low-rank approximation of all data (cross-validation).

2.2.2 Measured vs. Predicted RGBs: The first plot shows the measured and predicted RGB values in the test set (30% of the data randomly selected from the whole dataset with A, CWF, and Day combined). The model has good performance on the test set in terms of numerical differences between measured and predicted RGB values. Also, we can tell from the other three plots that the model generalizes well for all three illuminants.

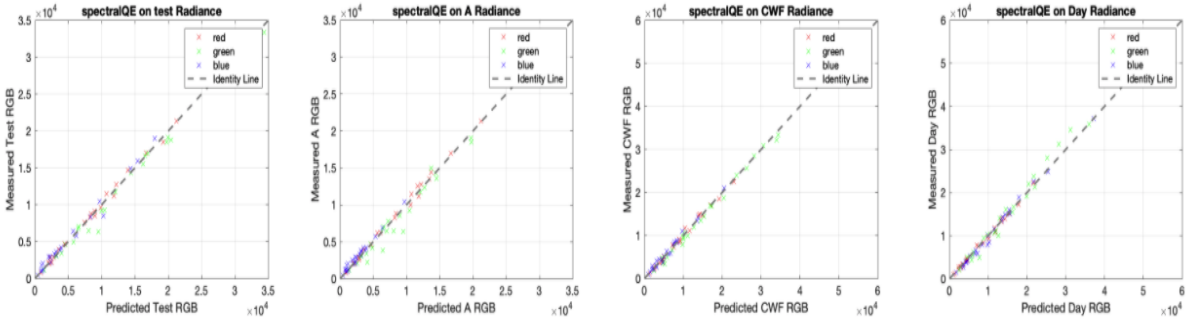

Figure 14. Measured vs predicted RGB with train data derived spectral QE.

2.2.3 ΔE of CIELAB: With the interpretation of the ΔE values provided in [2], the predicted color values are perceptible through close observation for radiance in sets test, A, CWF, and Day.

Test	A	CWF	Day
4.6271	5.043	3.1201	4.6688

3. MCC Visualization

We visualize the measured versus predicted MCC to examine their perceptual difference. Each row shows MCC under a illuminant. The first column shows the sensor measured MCC. The second column shows the MCC predicted by naive pseudo-inverse on train data. The third column shows the MCC predicted by SVD + pseudo-inverse on train data. The ΔE values for the predictions are also labeled in the captions for Figure 15, 16, 17. We observe that the predictions are generally perceptually close to the ground truth, an observation that is supported by the low ΔE values as well. The middle column has the lower ΔE values than the right column, possibly due to the fact that naive pseudo-inverse overfits on the train data and do not purposely drop model complexity like SVD does.

It is also noticeable that, compared with the reference MCC in Figure 18, all of our visualizations including measurement and prediction are very green. This is possibly due to the Bayer pattern in the sensor having twice amount of green channels than red or blue channels. The RGB we are visualizing are still sensor RGB, and not RGB values that are suitable for a display that we look at. However, the underlying values we are visualizing are most likely correct in the sense that the general pattern and trends in our visualization match those in the reference MCC. For example, the red color patches in reference MCC still look red in our visualizations, likewise for the blue patches. In addition, we also observe a grey-scale progression on the bottom rows of our visualizations.

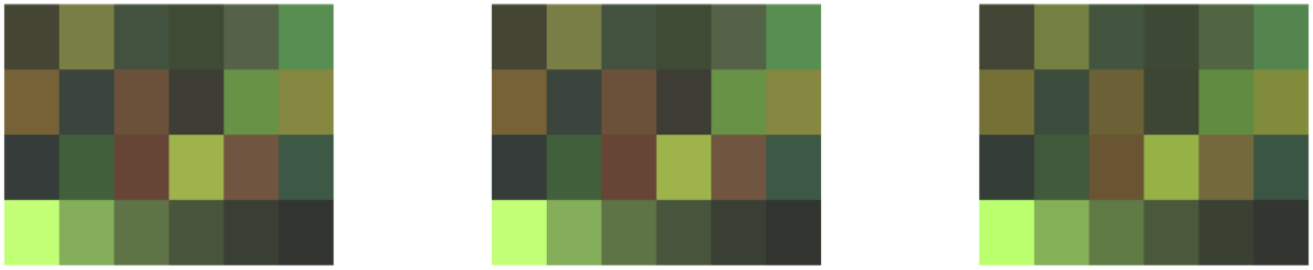

Figure 15. Illuminant A. Left: measured. Middle: pure pseudo-inverse on train data (ΔE=1.5384). Right: SVD + pseudo-inverse on train data (ΔE=5.043).

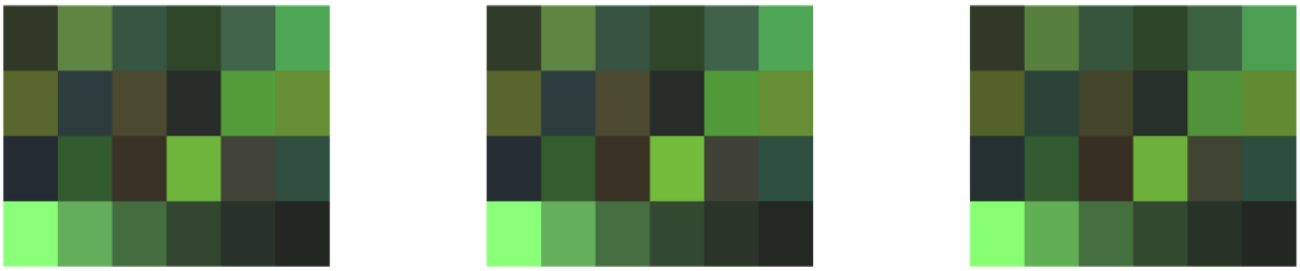

Figure 16. Illuminant CWF. Left: measured. Middle: pure pseudo-inverse on train data (ΔE=0.82624). Right: SVD + pseudo-inverse on train data (ΔE=3.1201).

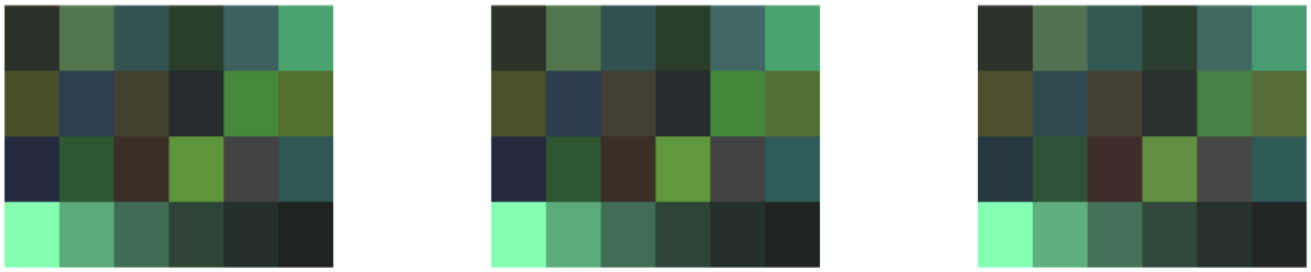

Figure 17. Illuminant Day. Left: measured. Middle: pure pseudo-inverse on train data (ΔE=0.76533). Right: SVD + pseudo-inverse on train data (ΔE=4.6688).

Figure 18. Reference MCC [4].

V. Conclusions

For this project, we tried two methods to build the spectral QE matrix - (1) pure pseudo-inverse and (2) pseudo-inverse with low-rank approximation via SVD. Both methods gave good quantitative results, in terms of numerical difference between predicted RGBs and measured RGBs and ΔE values. However, direct pseudo-inverse exhibits overfitting issues, which are later solved by SVD. Qualitatively, both methods show minimal perceptual differences from the ground truth MCC chart. Overall speaking, taking pseudo-inverse after applying SVD turns out to be the method that generates the best results across all three evaluation metrics. For future work, we can try re-mosaic our final RGB and swap in as sensor data to rebuild the scene.

VI. References

[1] Z. Lyu, K. Kripakaran, M. Furth, E. Tang, B. Wandell, and J. Farrell “Validation of image systems simulation technology using a Cornell Box,” arXiv, p. 2105.04106, May. 2021.

[2] Lawson, C. L. and R. J. Hanson. *Solving Least-Squares Problems*. Upper Saddle River, NJ: Prentice Hall. 1974. Chapter 23, p. 161.

[3] http://zschuessler.github.io/DeltaE/learn/

[4] https://hollynorth.com/product/macbeth-chart/

VII. Appendix

- MATLAB code: https://github.com/clairezb/Pixel4a_Color_Calibration