Image Color calibration using Evolutionary Optimization Algorithms

Juan Pablo Enríquez Pedroza
Facultad de Ingeniería
Universidad Panamericana
Aguascalientes, México
0228903@up.edu.mx

Ulises Gallardo Rodríguez
Facultad de Ingeniería
Universidad Panamericana
Aguascalientes, México
0229261@up.edu.mx

***Abstract*—Image-based color calibration is used to compensate for changes in the image due to illumination conditions. There are several proposals to solve the problem of color calibration using different techniques like Machine Learning and Histogram Matching. We use Evolutionary Algorithms such as Differential Evolution for accurate Color Calibration Matrix (CCM) estimation. The experimental results showed that the absolute difference between main chart colors is relatively high, so further improvements can be made to this method.**

Keywords—color calibration, optimization, color correction matrix, ArUco makers, evolutionary algorithms.

I. Introduction

Color is something of great relevance in various aspects of life. Some tools rely on color to give us information about an illness or the quality of something, to give us the best experience with, for example, a movie or a videogame or to save memories with the highest fidelity of reality in photos.

When taking a photo, many factors influence the quality of the final colors compared to the originals: light, reflections, perspective, etc. And sometimes, these photos contain very important information about, for example, a medical test that gives its results in a certain range of colors. That is why we have to be very careful with the conditions in which we take the photo: to get the colors as real as possible.

To achieve this, some tools for color correction were created. One example is the color calibration chart (or ColorChecker), a color range of 24 scientifically prepared natural, chromatic, primary, and grayscale frames. Many of the frames represent natural objects, such as human skin, foliage, and blue skies. Because they exemplify the color of their counterparts and reflect light equally in all parts of the visible spectrum, the frames will match the colors of representative samples of natural objects in any kind of lighting and with any color reproduction process. Each solid patch is individually formulated to produce pure, simple, and vivid color [1].

II. Our proposal

We used some Computer Vision techniques to find the color calibration chart in photos with different conditions and some Evolutionary Optimization Algorithms (something we did not see used elsewhere) to calibrate the colors in that photo based on the colors obtained from the original chart using a Color Correction Matrix (CCM): a tool that provides a means of subtracting out the overlap in the color channels caused by the fact that Blue light is seen by the Red and Green pixels on the imager, Red light is seen by the Blue and Green pixels, and Green light is seen by the Red and Blue pixels [11].

III. Related work

We selected two related articles that try to solve this problem:

- *Automatic color correction with OpenCV and Python by Adrian Rosebrock on February 15, 2021* [2].

  – *Summary:* after extracting actual colors from the image with the color checker chart taken under certain light conditions using ArUco makers, histogram matching was used to transfer the color intensity distribution from the reference image to the source or input image, to resolve this lighting discrepancy.

- *Using Machine Learning for Color Calibration with a Color Checker by Navarasu Oct 21, 2019* [3].

  – *Summary:* To extract the colors from the photo with the color checker, segmentation and search for the most dominant color were used. After this, the problem was solved using different techniques such as Machine Learning with PCA, PLSRegression and Root-Polynomial Regression methods to reduce dimensions of the problem and to find the relationship between color matrices. As well as Thin-Plate Spline Technique, which is an n-dimensional interpolation technique that can estimate the deformation between two images. In this case the technique is used to warp the input color matrix with the reference matrix and the unknown color region is transformed using the warping.

- *Camera Array Calibration with Color Rendition Charts* [4].

  – *Summary:* In this case, the problem to solve is the color correction between shots from different cameras. To handle this problem it was necessary to discover the color chart within images, by finding some interesting areas from the image and removing non-relevant areas, to finally construct the chart with axis orientation guessing with affine coordinate transform and chart orientation correction using the next idea: generate all possible (flipped, rotated, shifted) coordinate transforms and to choose the one with least color error while compensating for the illumination. In the end, the color transformation was done using a generic optimization algorithm called fmin_l_bfgs_b from scipy.optimize module, a variant of the BFGS algorithm.

- *When Color Constancy Goes Wrong: Correcting Improperly White-Balanced Images* [5].

  – *Summary:* This paper proposes to use a dataset of over 65,000 pairs of incorrectly white-balanced images and their corresponding correctly white-balanced images, to implement the k-nearest neighbor strategy that is able to compute a nonlinear color mapping function to correct the image's colors.

  As an overview, for the input sRGB image and the training data, they first extract the histogram feature of the input image, followed by generating a compact PCA feature to find the most similar k nearest neighbors to the input image in terms of colors. Based on the retrieved similar images, a color transform M is computed to correct the input image.

- *Color Calibration of Proximal Sensing RGB Images of Oilseed Rape Canopy via Deep Learning Combined with K-Means Algorithm* [6].

  – *Summary:* In this study it is not mentioned how to obtain the color calibration chart from the images but they focused in stimate the CCM efficiently. In this approach, they firstly manually derived CCMs by mapping RGB color values of each patch of a color chart obtained in an image to standard RGB (sRGB) color values of that chart. Then they grouped the images into clusters according to the CCM assigned to each image using the unsupervised k-means algorithm. Thirdly, the images with the new cluster labels were used to train and

validate the deep learning convolutional neural network (CNN) algorithm for an automatic CCM estimation. Finally, the estimated CCM was applied to the input image to obtain an image with a calibrated color. The performance of the model for estimating CCM was evaluated using the Euclidean distance between the standard and the estimated color values of the test dataset.

IV. Methodology

A. Step by step

Color Guide: We used a modified ColorChecker with ArUco markers (Figure A.I) in the corners to get the color reference we needed to calibrate our photos.

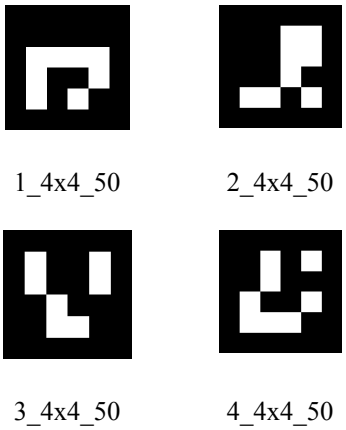


Figure A.I.Our ArUco Markers.

Color extraction: For our digital ColorChecker (our *control image*), we defined a grid and iterated over its intersections; every intersection was in a central area of a color frame from the chart (Figure B.II), so we defined a "window" to get a sample of the color and we obtained the average of the pixels in it to fill the color matrix that we would calibrate. Then, we used the ArUco markers to find our ColorChecker in the photo we were calibrating (our *input image*) and to make some adjustments with affine transformations to get a top-down view of our chart in order to obtain the colors easier. Once we got the top-down view of our chart, we applied the same method to get the colors as we did in our *control image*.

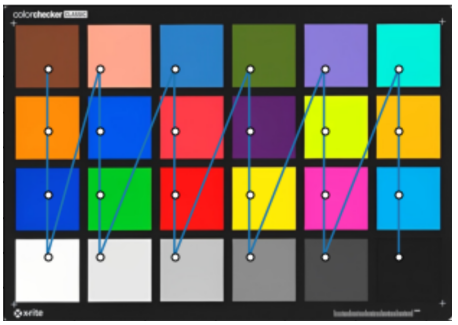


Figure A.II. Intersections of the grid in our *control image*.

Color calibration: We used three different Optimization algorithms (Differential Evolution, Particle Swarm Optimization and Evolutionary Programming) to get a Color Correction Matrix (CCM) that minimizes the difference between the color matrix of the *control image* and the color matrix of the *input image*. Finally, we multiplied the *input image* with the color correction matrix to get our resulting image.

B. Techniques

ArUco Markers

An ArUco marker is a synthetic square marker composed by a wide black border and an inner binary matrix which determines its identifier (id). The black border facilitates its fast detection in the image and the binary codification allows its identification and the application of error detection and correction techniques. The marker size determines the size of the internal matrix. For instance a marker size of 4x4 is composed of 16 bits [7].

Differential Evolution (DE)

It is a population-based metaheuristic search algorithm that optimizes a problem by iteratively improving a candidate solution based on an evolutionary process. Such algorithms make few or no assumptions about the underlying optimization problem and can quickly explore very large design spaces. DE is arguably one of the most versatile and stable population-based search algorithms that exhibits robustness to multi-modal problems [8].

Particle Swarm Optimization (PSO)

It is one of the bio-inspired algorithms and it is a simple one to search for an optimal solution in the

solution space and it also has very few hyperparameters.

It is inspired by the movement of a flock of birds when searching for food, all birds in the flock can share their discovery and help the entire flock get the best hunt.

This is a metaheuristic solution because we can never prove the real global optimal solution can be found and it is usually not. However, we often find that the solution found by PSO is quite close to the global optimal [9].

Evolutionary Programming (EP)

It is a method of optimization for combinatorial and real-valued functions, in which the optimization surface or fitness landscape is "rugged," possessing many locally optimal solutions.

It places emphasis on the behavioral linkage between parents and their offspring, rather than seeking to emulate specific genetic operators as observed in nature.

EP uses Gaussian distributed mutations and the self-adaptation paradigm. In EP, the parent selection mechanism is deterministic, and the survivor selection process (replacement) is probabilistic and based on a stochastic tournament selection [10].

V. Results

After some experimentation with different error measures, we decided to use a simple sum of absolute differences (SAD). Given the colors matrices C1, our *control image* color matrix, and C2, our *input image* color matrix multiplied by our color correction matrix, the distance between them is defined by:

$$\Delta(C_1, C_2) = \sum_i abs(r_i - r_i') + abs(g_i - g_i') + abs(b_i - b_i')$$

Where r, g, and b are the respective channel intensities for each color point in the matrices.

Using the selected Evolutionary Optimization Algorithms with a population size of 250 and 500 iterations, we got the results shown in the Table V.I.

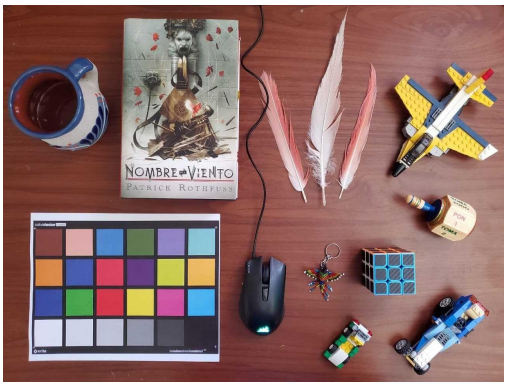
	Best fitness	execution time
Differential evolution	902.988345	17.140625
Evolutionary programming	934.066367	5.500000
Particle swarm optimization	909.242699	4.171875

Table V.I: Performance for each model (own implementation).

As we can see in Table V.I, we have two candidates for better performance: Differential Evolution (DE) returned the best result based on the fitness, but it takes four times longer than Particle Swarm Optimization (PSO) to return a value; on the other hand, PSO has the fastest execution time and also return an excellent fitness value, only 7 points worse than DE. Based on this, we can say that PSO is the best algorithm for our color calibration.

When we applied the returned color correction matrix to our *input image* we could observe a drastic change compared with the original version of it (Figure V.I). The color calibration chart has colors closer to our control image and we can see how this affects all the elements in the image. The calibrated image was similar with the three returned color correction matrices of the Optimization Algorithms.

Before



After

Figure V.I. Color correction matrix applied to our *input image.*

Something important to note after several executions is that there exists a local optimal solution where after some iterations the changes are minimal. This could have multiple factors but the algorithms gave a good result.

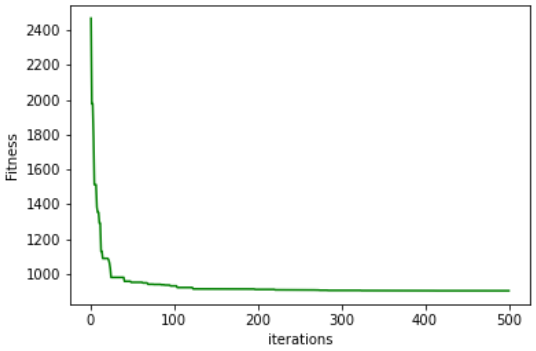


Figure A.IV. DE performance

IV. Conclusion

This problem is widely studied and different methods have been proposed to solve the color discrepancy between images in distinct conditions, as we could see in the Related Work section. Some of them required more advanced techniques and data to handle it.

Even though we got good results with our proposal, we observed that the best fitness apparently stayed at a local minimum, because the fitness result was still far from 0, so we think that there has to be a way to improve our implementation, mainly in the objective function.

Therefore, for future work it would be important to take into consideration comparison between other models results and even with programs like Adobe Photoshop.

V. References

[1] ColorChecker® Classic. (s/f). X-Rite.Retrieved on November 28, 2022, from https://www.xrite.com/es/categories/calibration-profiling/colorchecker-classic

[2] Rosebrock, A. (2021, febrero 15). Automatic color correction with OpenCV and Python. PyImageSearch. https://pyimagesearch.com/2021/02/15/automatic-color-correction-with-opencv-and-python/

[3] Navarasu. (2019, octubre 21). Using machine learning for color calibration with a color checker. FranciumTech.https://blog.francium.tech/using-machine-learning-for-color-calibration-with-a-color-checker-d9f0895eafdb

[4]Behringer, A., & Eisert, P. (s/f). Camera Array Calibration with Color Rendition Charts Studienarbeit HUMBOLDT-UNIVERSITÄT ZU BERLIN MATHEMATISCH-NATURWISSENSCHAFTLICHE FAKULTÄT II INSTITUT FÜR INFORMATIK. Hu-berlin.de.Retrieved on November 28, 2022, from https://www.informatik.hu-berlin.de/de/forschung/gebiete/viscom/thesis/final/Studienarbeit_Behringer_201308.pdf

[5]Afifi, M., Price, B., Cohen, S., & Brown, M. S. (2019). When color constancy goes wrong: Correcting improperly white-balanced images. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Retrieved on November 28, 2022, from https://cvil.eecs.yorku.ca/projects/public_html/sRGB_WB_correction/files/Afifi_When_Color_Constancy_Goes_Wrong.pdf

[6]Abdalla, A., Cen, H., Abdel-Rahman, E., Wan, L., & He, Y. (2019). Color calibration of proximal sensing RGB images of Oilseed rape canopy via deep learning combined with K-means algorithm. Remote Sensing, 11(24), 3001. https://doi.org/10.3390/rs11243001

[7]OpenCV: Detection of ArUco Markers. (s/f). Opencv.org. Retrieved on November 28, 2022, from https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html

[8]Georgioudakis, M., & Plevris, V. (2020). A comparative study of differential evolution variants in constrained structural optimization. Frontiers in built environment, 6. https://doi.org/10.3389/fbuil.2020.00102

[9]Tam, A. (2021, September, 15). A gentle introduction to particle swarm optimization. Machinelearningmastery.com; Machine Learning Mastery. https://machinelearningmastery.com/a-gentle-introduction-to-particle-swarm-optimization/

[10]Jacob, C. (2001). Evolutionary Programming. En Illustrating Evolutionary Computation with Mathematica (pp. 297–344). Elsevier. https://www.sciencedirect.com/topics/computer-science/evolutionary-programming

[11]Color correction matrix (CCM). (s/f). Imatest.com. Retrieved on November 28, 2022, from https://www.imatest.com/docs/colormatrix/

[12]Color Correction Matrix Application Note. (s/f). Lumenera.com. Retrieved on November 28, 2022, fromhttps://www.lumenera.com/media/wysiwyg/support/pdf/LA-2100-CustomCorrectionMatrixAppNote.pdf

[13]Murty, N. (2021, julio 9). Why every artist needs a Color Calibration Chart - color tool for artists. Nancy Murty. https://www.nancymurty.com/meet-your-new-friend-the-color-calibration-chart/