Color Calibration with Machine Learning Methods

Pang Liu
Github source codes: jeffliulab/Color_Calibration

March 14, 2025

Contents

1 Introduction **2**

2 Project Pipeline and Data Preparation **2**
2.1 Color Prediction System 2
2.2 ETL Processing 2
2.3 Data Augmentation 3
2.4 Feature Extraction (ETV Processing) 3

3 Modeling Approach and Results **4**
3.1 Mathematical Formulation and Feature Construction 4
3.2 Model Selection and Performance Comparison 5
3.3 Model Deployment 5

4 Conclusion **6**

A Appendix: Color Calibration Card **6**

B Appendix: Model Results of Predicted vs. True Colors **7**

C Appendix: Generalization Test **8**

1 Introduction

Color calibration plays a vital role in ensuring accurate color reproduction in photography, industrial printing, and other imaging applications. This project develops an automated system for detecting and standardizing color calibration cards in images. The main objective is to design a robust calibration method that can detect colors under various conditions, offering a generalizable solution across different lighting scenarios.

2 Project Pipeline and Data Preparation

This project adopts an end-to-end workflow, covering data processing, feature extraction, and machine learning model training. The overall pipeline is depicted in Figure 1.

2.1 Color Prediction System

Based on the trained models, an automated Color Prediction System was developed. It can be viewed as part of the pipeline, integrating detection, extraction, and prediction in the following main steps:

1. **Color Region Detection:** Utilize the YOLO object detection model (PatternDetector) to recognize reference objects on the calibration card (e.g., `red_circle`, `green_triangle`, `blue_pentagon`, `black_box`).
2. **Color Extraction and Deviation Calculation:** Extract RGB values from the detected regions and compute deviation features (i.e., the difference between captured reference colors and standard colors).
3. **Color Prediction:** Use the trained model (Random Forest in our best case) for color correction and predict the true color of the target patch.
4. **Result Visualization:** Display the captured color (C_p), predicted true color ($C_{s,\text{pred}}$), and reference colors (R_p, G_p, B_p) for comparison.

2.2 ETL Processing

The initial step in the pipeline involves pattern recognition using annotations and the YOLOv8 object detection model. YOLO is employed to detect color patches and other patterns on the calibration card. Figure 1 provides an overview of the entire pipeline.

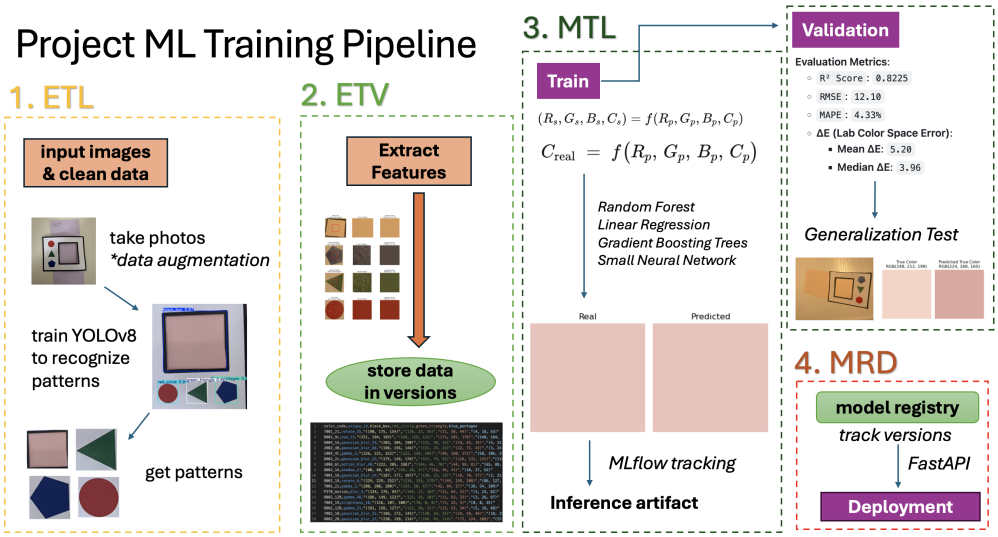


Figure 1: Project Pipeline Overview

2.3 Data Augmentation

To improve model robustness and enlarge the dataset, a series of data augmentation techniques were applied. The original 250 photos were augmented to over 2000 images. The main augmentations include:

- **Brightness:** `A.RandomBrightnessContrast(p=1.0)`
- **Hue:** `A.HueSaturationValue(p=1.0)`
- **Gamma:** `A.RandomGamma(p=1.0)`
- **Motion Blur:** `A.MotionBlur(blur_limit=5, p=1.0)`
- **Gaussian Blur:** `A.GaussianBlur(blur_limit=5, p=1.0)`
- **CLAHE:** `A.CLAHE(clip_limit=4.0, p=1.0)`
- **Noise:** `A.ISONoise(p=1.0)`
- **Rotation:** `A.Rotate(limit=10, p=1.0, border_mode=cv2.BORDER_REFLECT)`

2.4 Feature Extraction (ETV Processing)

In the data preprocessing stage, color information is extracted from the calibration card to obtain the following features:

- Extract the Red, Green, Blue, and Contrast values (denoted as R_p, G_p, B_p, and C_p, where p indicates the photo).
- Label each detected color patch with its corresponding reference color.
- Store the extracted features in structured datasets for further processing.

After running the extraction process, the features from 9000 photos are stored in a CSV file. Given that the standard reference colors R_s, G_s, and B_s are fixed (e.g., red as $(255, 0, 0)$, green as $(0, 255, 0)$, blue as $(0, 0, 255)$), only the standard target color C_s (labeled as "real_rgb") needs to be obtained for calibration.

3 Modeling Approach and Results

3.1 Mathematical Formulation and Feature Construction

We have four primary color values of interest:

- $\mathbf{R_p}, \mathbf{G_p}, \mathbf{B_p}$: The camera-captured RGB values for the three **reference** patches (red, green, blue).
- $\mathbf{C_p}$: The camera-captured RGB value for the **target** patch.
- $\mathbf{R_s}, \mathbf{G_s}, \mathbf{B_s}$: The known **standard** RGB values of the reference patches, for example:

$$\mathbf{R_s} = (255,\ 0,\ 0), \quad \mathbf{G_s} = (0,\ 255,\ 0), \quad \mathbf{B_s} = (0,\ 0,\ 255).$$

- $\mathbf{C_s}$: The **true** (or canonical) RGB value of the target patch.

Matrix-Based Approach. If we treat the calibration as a linear mapping from captured colors to standard colors, we can write:

$$\begin{pmatrix} R_s \\ G_s \\ B_s \\ C_s \end{pmatrix} = \mathbf{M} \cdot \begin{pmatrix} R_p \\ G_p \\ B_p \\ C_p \end{pmatrix},$$

where the left-hand vector represents the standard values of the reference patches plus the target patch, and the right-hand vector represents the camera-captured values. The 4×4 matrix $\mathbf{M}$ is learned from training data.

General Function. Color distortion may be nonlinear. Hence, one may seek a function $f(\cdot)$ such that

$$(R_s,\ G_s,\ B_s,\ C_s) \ = \ f(R_p,\ G_p,\ B_p,\ C_p),$$

or equivalently focus on the target patch:

$$C_{\text{real}} \ = \ f(R_p,\ G_p,\ B_p,\ C_p).$$

Color Differences (Δ) as Features. In practice, we compute the difference between the camera-captured reference patches and their known standard values. For instance, for the red patch:

$$\Delta_{RR,\text{red}} = R_{p,R} - 255, \quad \Delta_{RG,\text{red}} = R_{p,G} - 0, \quad \Delta_{RB,\text{red}} = R_{p,B} - 0,$$

and similarly for green and blue. We then combine these nine difference values with the target patch's captured RGB $(C_{p,R}, C_{p,G}, C_{p,B})$ to form a 12-dimensional feature vector, which serves as input X. The output is $\mathbf{C_s}$, the standard (true) color of the target patch.

3.2 Model Selection and Performance Comparison

We experimented with four models to predict the true color from the extracted features: **Linear Regression**, **Random Forest Regression**, **Gradient Boosting Trees (XGBoost)**, and a **Small Neural Network (MLP)**. A non-model baseline method (predicting via the mean of the training set) was also tested.

A single train/test split (70%/30%) was performed, and each model was evaluated using R^2, RMSE, MAPE, and ΔE (in Lab space). Table 1 summarizes the results:

Model	R^2	**RMSE**	**MAPE**	**Mean** ΔE	**Median** ΔE
Baseline	-0.0037	31.55	14.98%	14.61	13.84
Linear Regression	0.7113	14.98	6.41%	20.87	6.63
Random Forest	0.8225	12.10	4.33%	5.20	3.96
XGBoost	0.8280	11.76	4.09%	5.14	3.95
Small Neural Net	0.7068	14.22	5.92%	7.39	6.70

Table 1: Performance Comparison of Different Models (70% training, 30% testing).

From Table 1, **Random Forest** and **XGBoost** achieve the best performance in most metrics. The mean ΔE of around 5 indicates a decent commercial-grade performance.

Hyperparameter Tuning: To optimize model performance, Bayesian optimization with K-Fold Cross-Validation ($K = 3$) is employed for XGBoost. The tuning process efficiently searches the optimal hyperparameters, leveraging GPU acceleration for faster computation. The final optimized model, trained on March 7, achieved the highest accuracy with Lab Mean $\Delta E = 4.59$ and Lab Median $\Delta E = 3.61$, meeting general commercial printing standards. The key parameters optimized include:

- **XGBoost:** learning rate, number of estimators, maximum depth, minimum child weight, gamma, subsample ratio, column sample ratio, L_1 and L_2 regularization.

3.3 Model Deployment

Finally, the model was deployed for end-to-end usage, as shown in Figure 2. This system can run on cloud services (e.g., Google Cloud) for real-time color calibration, enabling wider accessibility and scalability.

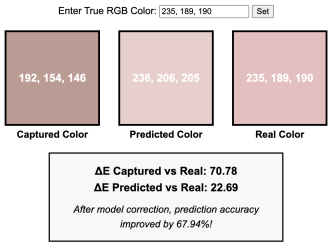


Figure 2: End-to-end model deployment on Google Cloud

4 Conclusion

This project presents a comprehensive color calibration workflow, covering data preprocessing, feature extraction, and machine learning model training and comparison. Experimental results indicate that models based on Random Forest and XGBoost significantly outperform simple linear regression and baseline methods, achieving normal commercial-grade performance (with a mean ΔE around 5) in training sets, and can significantly improve the output accuracy in generalizing tests.

A Appendix: Color Calibration Card

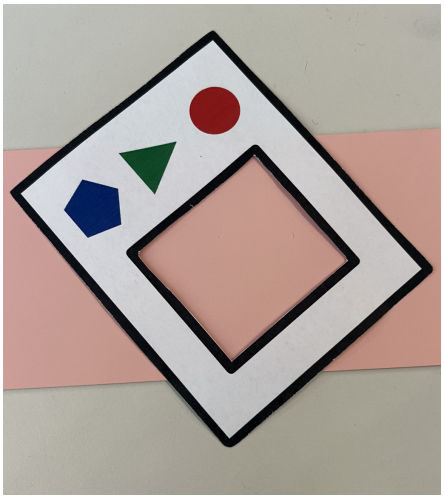
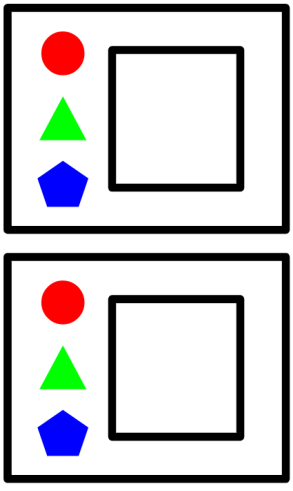
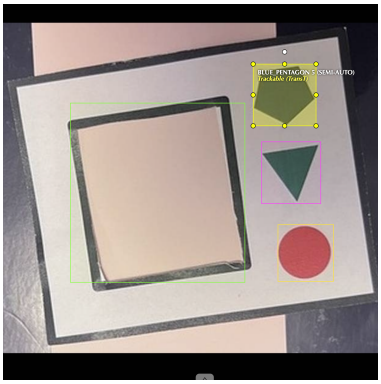


Figure 3: Calibration Card Image 1 (printed)



(a) Edition Prepared For Printing



(b) YOLO Detection of Color Patterns

Figure 4: Color Calibration Cards

B Appendix: Model Results of Predicted vs. True Colors

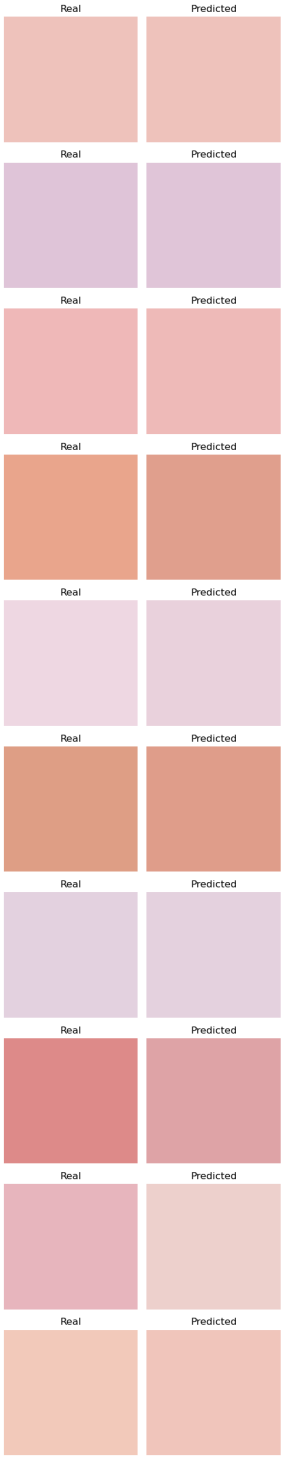


Figure 5: Model Training Results: Predicted vs. True Colors.

C Appendix: Generalization Test

To test the generalization performance, an independent photo was taken under extremely yellow lighting conditions (Figure 6). This calibration card image was not included in any of the collected data, making it a strong test of the model's ability to generalize.

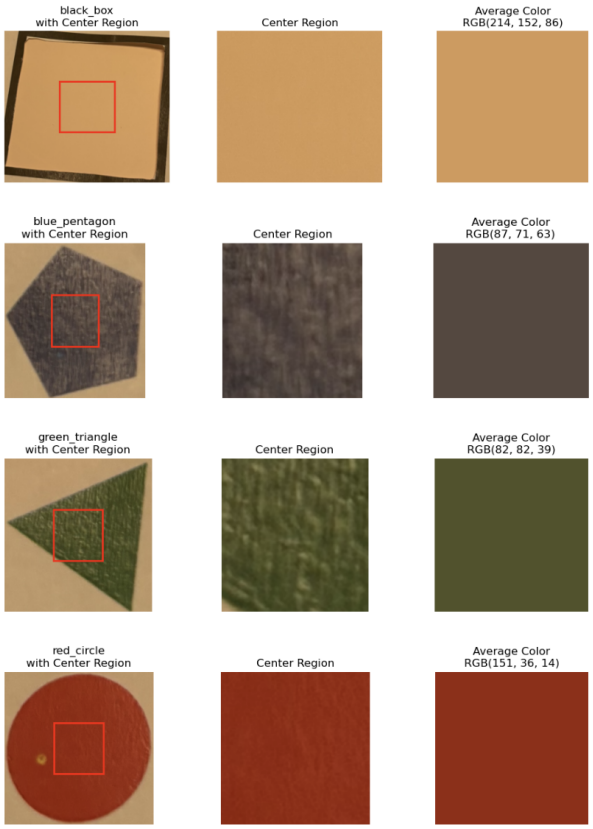


Figure 6: Sampling under extreme yellow lighting for generalization testing

Results:

- Computation of ΔE in Lab color space:

$$\Delta E(\, C_p,\ \text{True Color}) = 38.07 \quad \text{(Significant error)}$$

$$\Delta E(\, C_{s,\text{pred}},\ \text{True Color}) = 16.46 \quad \text{(Significant improvement)}$$

- Although the error is still larger than the ΔE observed in the training and test sets, the model-predicted color ($C_{s,\text{pred}}$) is noticeably closer to the true target color compared to the directly captured color (C_p).

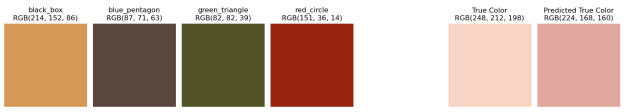


Figure 7: Generalization Test Results: Comparison between C_p and $C_{s,\text{pred}}$

This result serve as a baseline for color calibration card in generalization test.

References

[1] Finlayson, G. D., & Drew, M. S. (1997). *Constrained least-squares regression in color spaces.* Journal of Electronic Imaging, 6(4), 484–493.
This paper explores the application of constrained least-squares regression in color transformations, addressing issues related to noise and illumination variations.

[2] Hong, G., Luo, M. R., & Rhodes, P. A. (2001). *A study of digital camera colorimetric characterisation based on polynomial modeling.* Color Research & Application, 26(1), 76–84.
This study discusses in detail how polynomial models can represent the mapping from camera RGB signals to true colors, providing insights into the effectiveness of various polynomial degrees in characterizing digital cameras.

[3] Gatta, C., et al. (2007). *A LUT-based approach for color correction in an end-to-end color imaging system.* CIC15: Fifteenth Color Imaging Conference, 327–330.
This paper introduces the use of 3D Look-Up Tables (LUTs) for end-to-end color correction, covering aspects such as data acquisition and LUT interpolation techniques.

[4] Wei, X., Luo, M. R., & Pointer, M. R. (2019). *Evaluation of some non-linear methods for camera color characterisation.* Color Research & Application, 44(2), 291–303.
This research evaluates various non-linear methods—including neural networks, polynomial models, and Look-Up Tables—for camera color characterization, comparing their performance and applicability.

[5] Shi, L., & Healey, G. (2002). *Using reflectance spectra to recover device-independent color.* Color Research & Application, 27(1), 50–59.
This paper delves into the relationship between camera spectral responses and true surface reflectance, discussing methods to recover device-independent color representations from reflectance spectra.