CCMNet: Leveraging Calibrated Color Correction Matrices for Cross-Camera Color Constancy

Dongyoung Kim[1] Mahmoud Afifi[2] Dongyun Kim[1] Michael S. Brown[2] Seon Joo Kim[1]

[1]Yonsei University [2]AI Center - Toronto, Samsung Electronics

{dongyoung.kim,dongyunkim,seonjookim}@yonsei.ac.kr {m.afifi1,michael.b1}@samsung.com

Project page: https://www.dykim.me/projects/ccmnet

Figure 1. This paper introduces CCMNet, a framework for cross-camera color constancy. CCMNet uses pre-calibrated color correction matrices (CCMs) from camera ISP hardware to train an encoder that generates a *camera fingerprint embedding* (CFE), capturing the testing camera's color space. In (A), we show a raw image from a Canon 550D. In (B), we present C5 [6], which generalizes using randomly selected unlabeled images from the test camera—C5's performance varies depending on the image set. In (C), we show our results, relying only on fixed CCMs in the ISP. Neither method used Canon 550D data during training. Gamma correction was applied for visualization.

Abstract

Computational color constancy, or white balancing, is a key module in a camera's image signal processor (ISP) that corrects color casts from scene lighting. Because this operation occurs in the camera-specific raw color space, white balance algorithms must adapt to different cameras. This paper introduces a learning-based method for cross-camera color constancy that generalizes to new cameras without retraining. Our method leverages pre-calibrated color correction matrices (CCMs) available on ISPs that map the camera's raw color space to a standard space (e.g., CIE XYZ). Our method uses these CCMs to transform predefined illumination colors (i.e., along the Planckian locus) into the test camera's raw space. The mapped illuminants are encoded into a compact camera fingerprint embedding (CFE) that enables the network to adapt to unseen cameras. To prevent overfitting due to limited cameras and CCMs during training, we introduce a data augmentation technique that interpolates between cameras and their CCMs. Experimental results across multiple datasets and backbones show that our method achieves state-of-the-art cross-camera color constancy while remaining lightweight and relying only on data readily available in camera ISPs.

1. Introduction

Computational color constancy ensures that object colors remain consistent under varying lighting conditions [10]. In digital cameras, this is achieved through white balancing, which adjusts raw image colors to simulate neutral lighting [11, 20, 31]. This involves two main steps: illuminant estimation and linear white-balance correction [21].

Illuminant estimation predicts the color of the scene's light source under the assumption of single-source illumination [31]. The estimated illuminant color is then used in linear white-balance correction to counteract the effects of lighting and camera response biases [17, 26]. These steps are applied early in the image signal processor (ISP) pipeline to raw images [4] and are influenced by the camera's sensor-specific characteristics, such as response functions and lens properties [2, 52]. These factors complicate the generalization of illuminant estimation algorithms across cameras with varying characteristics [3].

Recent work on illuminant estimation achieves promising results using learning-based models [7, 30, 48, 65]. These models learn a mapping between input image colors

and scene illuminant colors using pairs of images and corresponding ground-truth illuminant colors, typically captured by the same camera used in testing [53]. Consequently, most learning-based methods struggle to generalize to new cameras with different characteristics than those used during training [3]. This limitation hinders their practical applicability in manufacturing, as fine-tuning or retraining is necessary for each new camera introduced. Some recent attempts have proposed solutions for improved adaptation through few-shot learning [63], or by using additional unlabeled images captured by the testing camera at inference time to facilitate generalization to the testing camera's color space [6]. While these methods show promising results, they require capturing new images with the testing camera for adaptation [46], making their performance inherently dependent on the characteristics of those images [6].

While camera ISPs rely on pre-calibrated, camera-specific information to assist in color processing after white balance has been applied, to the best of our knowledge, no prior work has leveraged this calibrated information for the cross-camera color constancy task. Specifically, consumer-grade ISPs rely on calibrated color correction matrices (CCMs) to transform the camera's raw color space to a device-independent standard color space (e.g., CIE XYZ). These CCMs are carefully calibrated during ISP manufacturing [41], are easily accessible within the ISP's firmware [17, 21, 33], and are also available in DNG files for post-capture raw rendering [39]. The availability of this information motivated us to utilize this calibrated data to improve cross-camera generalization.

Contribution In this paper, we present CCMNet, a learning-based method for cross-camera color constancy built on the convolutional color constancy (CCC) framework [6, 12, 13, 37]. Our method leverages pre-calibrated color correction matrices (CCMs) available from camera ISPs to transform predefined illuminant colors along the Planckian locus from the device-independent CIE XYZ color space into the raw space of the test camera. These transformed illuminations encode the unique characteristics of the camera's response function and serve as reference points. The transformed illuminations are compressed into an 8-dimensional embedding, allowing a learnable hypernetwork to adapt to the test camera's raw color space and generate a camera-specific CCC model tailored to the input image. Additionally, we introduce an augmentation technique that maps training images from a limited set of cameras to imaginary raw spaces, improving generalization. Consequently, CCMNet, which combines a design that dynamically adapts to the raw space of various cameras with a robust data augmentation strategy, accurately estimates illuminant colors for cameras unseen during training (see Fig. 1-C). Our approach is lightweight, accurate, and requires no additional test camera images, unlike prior work [6].

2. Related Work

A camera's ISP includes several color processing modules applied in a pipeline fashion. One of the early-stage modules corrects the colors of the captured raw image through two key steps [17, 26, 40]: (1) image white balancing (Sec. 2.1) and (2) transferring the camera raw colors to a standard color space via CCMs (Sec. 2.2).

2.1. Auto White Balance

As discussed in Sec. 1, auto white balance modules consist of two steps: illuminant estimation and correction. The correction is applied to the linear raw image colors, often using a diagonal correction matrix [10]. Most research focuses on illuminant estimation, which determines the scene's illuminant color in the camera's raw space. This can be categorized into learning-free (statistical) methods (e.g., [18, 19, 23, 32, 45, 55, 56, 60, 61]) and learning-based methods (e.g., [7, 12, 36, 48, 59]).

Statistical-based methods rely on specific hypotheses (e.g., gray-world [18], gray-edges [61], etc.) and use statistics derived from the input raw image colors to estimate the illuminant color. As a result, they inherently generalize across different cameras. However, these methods often have limited accuracy and may fail in scenarios where the scene's illuminant color cannot be reliably inferred from the captured image.

Learning-based methods (e.g., [12, 13, 16, 29, 30, 36, 47, 48, 53, 54, 58, 64, 65, 67]) improve accuracy by training models to map raw colors to illuminant colors. However, most fail to generalize to unseen cameras [3]. Some approaches attempt adaptation via meta-learning and few-shot learning [50], assuming access to a range of illuminant colors from the testing camera [34, 68], or creating generic methods that require fine-tuning on the new camera [14].

Among these efforts, our work falls into a category of methods designed to achieve adaptation without requiring a paired set of images from the test camera, even if that set is small. To this end, the work in [3] (termed SIIE) maps input raw images from different cameras to a learnable *working space*, reducing disparities in raw color spaces before illuminant estimation. However, this method assumes access to a diverse range of training cameras to effectively learn this mapping, making its accuracy dependent on the variability of the training data. More recently, C5 [6] utilizes additional images captured by the test camera during inference to dynamically generate a CCC model [12, 13]. While this method achieves promising results, its accuracy heavily depends on the characteristics of the additional images provided (see Fig. 1-B).

In contrast, our method leverages a static set of predefined guidance colors along with pre-calibrated data from the test camera, enabling consistently high accuracy without requiring additional images from the test camera.

2.2. Color Space Transfer via CCMs

Camera sensors exhibit unique spectral sensitivity and bias, resulting in each camera having its own native RGB color space. Camera ISP manufacturers calibrate and apply color correction matrices (CCMs) to facilitate image processing, ensuring an appropriate transformation between the native RGB space and a device-independent standard color space (e.g., CIE XYZ) within the imaging pipeline [15, 40].

Although the transformation between the camera's raw space and a standard color space is often nonlinear [24, 25, 35, 38], cameras primarily rely on linear transformation matrices due to their simplicity and practical benefits [24]. CCMs are typically calibrated by fitting a 3×3 matrix that maps the raw RGB values of a calibration object (e.g., a color chart) to their corresponding values in a standard color space under an illuminant with a specific correlated color temperature (CCT) [39, 41]. To accommodate diverse lighting conditions, CCMs are precomputed for at least two illuminants (typically for low and high CCTs [51]) and interpolated for intermediate conditions (see Fig. 2).

CCMs serve as the critical link between a camera's unique color characteristics and a standard color space. While most CCMs are designed to transform white-balanced camera raw-RGB to CIE XYZ, some types of CCMs within the ISP operate in the reverse direction, mapping observed CIE XYZ under a specific illuminant back to the camera's native raw-RGB space, as shown in Fig. 2-A. This inverse transformation, in particular, provides insight into how various illuminants are represented in the native raw-RGB space. By leveraging this transformation, we can approximate the color trajectories of illuminants in the raw-RGB domain across a range of CCTs.

We leverage this property of CCMs as a *bridge* to introduce a novel illuminant estimation method that adapts to the color space of unseen cameras. While previous studies have leveraged CCMs for data augmentation [6], none have explored their use during inference to improve illuminant estimation across different cameras. Additionally, we introduce a data augmentation strategy that exploits the linearity of CCMs to enhance generalization. Specifically, we propose a technique to generate *imaginary* camera images with corresponding CCMs, further improving the robustness and adaptability of our model.

3. Method

3.1. Preliminary

Auto White Balance Formulation. Assuming a single global illumination, a given linear raw image, I, is formed as the element-wise product of its white-balanced counterpart, W, and the global illuminant RGB color vector, ℓ, at every pixel location x. This can be mathematically described as follows:

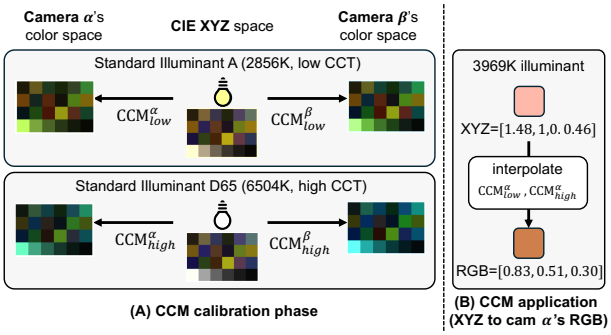

Figure 2. Example of CCM calibration (A) and application (B). CCMs are calibrated to transform between CIE XYZ chromaticity and camera-specific raw-RGB values under standard illuminants with predefined color temperatures (e.g., 2856K, 6504K). For other illuminants, the calibrated CCMs are interpolated. As a result, CCMs reflect the camera's unique color characteristics, capturing how the camera *perceives* illuminants along the color temperature trajectory.

$$I^x = W^x \circ \ell \quad \forall_x. \tag{1}$$

The conventional goal of the auto white-balance task is to optimize a model f that estimates the illumination RGB from a given raw image I:

$$[\hat{\ell}_R, \hat{\ell}_G, \hat{\ell}_B]^T = f(I). \tag{2}$$

Convolutional Color Constancy (CCC). As shown in the upper flow of Fig. 3-A, our method is fundamentally based on the CCC framework [12, 13], which transforms the image histogram N into an illuminant heatmap P, using a filter F and a bias B. CCC reformulates the illumination estimation problem as a coordinate localization task on a log-chroma histogram [22], commonly termed a uv-histogram. Specifically, for an image's RGB pixel $[I_R, I_G, I_B]$, the log-chroma values, u and v, are calculated as follows (pixel coordinate x is omitted for simplicity):

$$I_u = \log(I_G/I_R), \; I_v = \log(I_G/I_B). \tag{3}$$

After that, a uv-histogram can be generated as follows:

$$N(u,v) = \sum_x \|I^x\|_2 \, [|I^x_u - u| \leq \epsilon \wedge |I^x_v - v| \leq \epsilon], \tag{4}$$

where ϵ is the width of the histogram bin and $\|I^x\|_2$ is the weighting factor for each pixel, defined as the L2 norm of the raw RGB values of the pixel. In other words, the value of $N(u,v)$ represents the weighted count of pixels in image I that fall within a certain range (ϵ) around the point (u,v).

The goal of CCC is to optimize a global filter F and bias B, to predict a probability map P of the illumination within the histogram space using the following equation:

$$P = \sigma(B + \sum_i (N_i * F_i)), \tag{5}$$

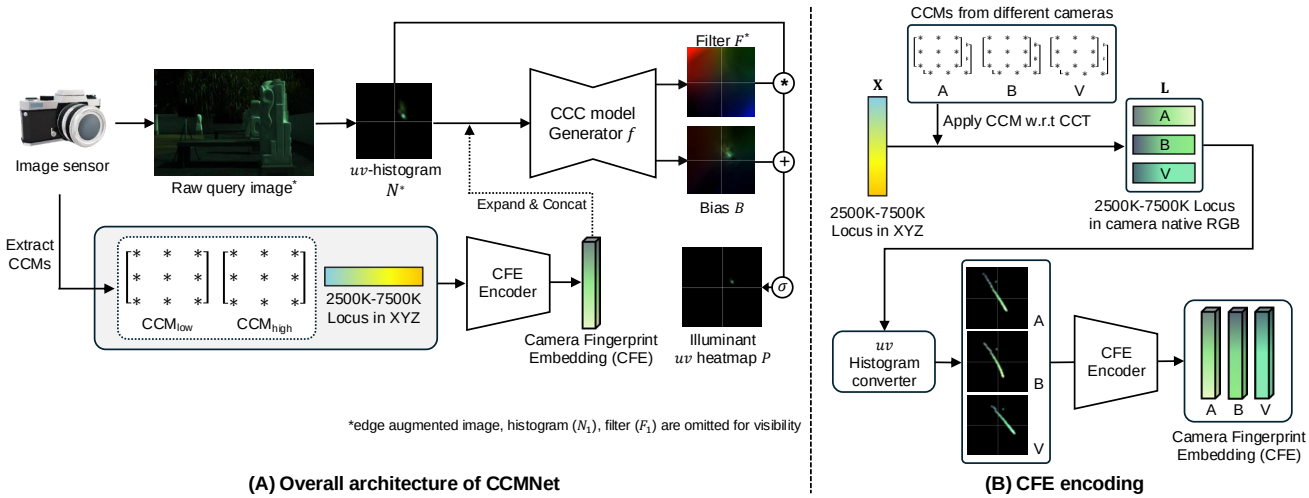

Figure 3. Overview of the CCMNet architecture. (A) Based on CCC [12] and C5 [6], CCMNet includes a network f that generates filters and bias from the uv-histograms of the input image. To process query images from diverse camera domains with varying spectral sensitivities, CCMNet uses a camera fingerprint embedding (CFE) as guidance. (B) The CFE for three example cameras (A, B, V)—two real (A, B) and one imaginary (V)—is constructed by mapping predefined illuminants (2500K–7500K along the Planckian locus) from the CIE XYZ space to each camera's native raw RGB space using calibrated CCMs. These values are converted into a 64×64 histogram and encoded into a 1D vector via a lightweight encoder.

where F and B have the same shape as the uv-histogram N, $*$ represents the convolution operation (accelerated using fast Fourier transforms), σ represents the softmax operation over the uv-coordinate space, and the subscript i denotes the index corresponding to the filter and histogram. Here, $i = 0$ refers to the original raw image, while $i \geq 1$ corresponds to augmented images (e.g., texture, edge). We can interpret P as a heatmap of confidence for each u, v coordinate, so the final prediction $(\hat{\ell}_u, \hat{\ell}_v)$ is expressed as a weighted sum of the coordinates using P:

$$\hat{\ell}_u = \sum_{u,v} uP(u,v), \quad \hat{\ell}_v = \sum_{u,v} vP(u,v). \tag{6}$$

The final RGB illumination estimate, $\hat{\ell}$, is obtained by inverting the transformation in Eq. (3):

$$\hat{\ell} = \left[\exp\left(-\hat{\ell}_u\right), 1, \exp\left(-\hat{\ell}_v\right)\right], \tag{7}$$

where the green channel is assumed to be G=1. Alternatively, $\hat{\ell}$ can be normalized to ensure that the vector has unit length.

The training objective of CCC is to optimize the filter and bias to minimize the angular error between the predicted illumination RGB and the ground truth illumination in the training dataset. For cross-camera color constancy, C5 [6] proposes a hypernetwork version of CCC that dynamically generates F and B for the test image after analyzing histograms of additional images taken by the same camera.

3.2. CCMNet

The proposed CCMNet framework is built upon C5 [6]. As mentioned above, C5 is a hypernetwork version of CCC [12, 13], where the network dynamically generates filters and bias. However, unlike C5, CCMNet does not require additional images from the target camera (typically 6–8). Instead, our method leverages two pre-calibrated Color Correction Matrices (CCMs) for low and high correlated color temperatures (CCTs) embedded within the ISP (see Fig. 3-A). These CCMs provide stable guidance via Camera Fingerprint Embedding (CFE) (see Fig. 3-B), ensuring consistent performance across diverse camera domains without extra test images. The core formulation of CCMNet is as follows:

$$\{F_0, F_1, B\} = \text{CCMNet}(N_0, N_1, \text{CCM}_{low}, \text{CCM}_{high}), \tag{8}$$

where N_0 and N_1 denote the original raw image and its edge-augmented counterpart, CCM_{low} and CCM_{high} are pre-calibrated matrices corresponding to low and high CCTs (typically 2500K and 6500K). The outputs F_0, F_1, and B, generated by CCMNet, are used to estimate the final uv coordinate of the illumination through Eqs. (5)–(7).

In Sec. 3.3, we introduce Camera Fingerprint Embedding (CFE), a method for extracting device-specific guidance features using CCMs. Additionally, in Sec. 3.4, we propose an imaginary camera augmentation technique to mitigate overfitting to the limited number of cameras and CCMs used during training. These strategies (CFE and augmentation) enhance CCMNet's ability to generalize across diverse spectral sensitivities, achieving state-of-the-art performance in cross-camera color constancy tasks.

3.3. Camera Fingerprint Embedding

Cross-camera color constancy aims to estimate the chromaticity of the light source while adapting to unseen sensor domains. To this end, we introduce a device-aware guidance feature called Camera Fingerprint Embedding (CFE). CFE encodes the color trajectory of light sources *observed* by each camera within a specific color temperature range into an 8-dimensional vector. As a result, it inherently captures each camera's unique color characteristics, enabling the model to adapt to the color space of previously unseen cameras. As shown in Fig. 3-B, CFE is generated through a two-step process. First, a set of illuminants along the Planckian locus (covering color temperatures from 2500K to 7500K) is converted into the specific camera's native raw RGB space using its pre-calibrated CCMs. Second, the resulting RGB illuminant colors are transformed into uv-histogram, which is then processed by a CNN-based encoder to extract device-specific feature.

Camera-Native Guidance Illuminants. Our goal is to obtain $\mathbf{L}$, the chromaticity set of light sources within a specific color temperature range as observed by a given camera. To achieve this, we use calibration matrices that transform the CIE XYZ coordinates of standard illuminants A or D65 into the camera's raw space. These calibration data are typically provided during camera manufacturing and can be extracted from DNG files produced by most cameras. For simplicity, we refer to these matrices as CCM_{low} and CCM_{high} throughout this paper, as used in Eq. (8). For details on CCM properties and extraction methods, please refer to the supplementary materials.

First, illuminant colors along the Planckian locus in the device-independent CIE XYZ color space are sampled at 100K intervals within the 2500K to 7500K color temperature range. Each sampled XYZ point, $\mathbf{X}_t$, corresponding to an illuminant with color temperature t, is then transformed into a camera-native RGB color, $\mathbf{L}_t$, using the following equation:

$$\mathbf{L}_t = \text{CCM}_t \mathbf{X}_t, \tag{9}$$

where CCM_t is a transformation matrix for color temperature t that maps the CIE XYZ values of an illuminant with color temperature t to the target camera's raw space. Since CCM_{low} and CCM_{high} are calibrated at specific color temperatures, interpolation is used to compute CCM_t for an arbitrary color temperature t. The interpolation of CCM_t is defined as:

$$\begin{aligned} \text{CCM}_t &= g\text{CCM}_{low} + (1-g)\text{CCM}_{high}, \\ \text{where } g &= \frac{t^{-1} - \text{CCT}_{high}^{-1}}{\text{CCT}_{low}^{-1} - \text{CCT}_{high}^{-1}}, \end{aligned} \tag{10}$$

where CCT_{low} and CCT_{high} denote the color temperatures of the standard illuminants for which CCM_{low} and CCM_{high} are calibrated, typically around 2500K and 6500K, respectively. The resulting set of camera-native RGB colors, $\mathbf{L}_t \mid t \in \{2500, 2600, \ldots, 7500\}$, represents the illumination colors along the Planckian locus in the camera's raw RGB space, sampled within the 2500K—7500K range as observed by a specific image sensor.

Histogram Conversion & Encoding. The camera-specific guidance illuminant set, $\mathbf{L}$, is transformed into a uv-histogram using Eq. (3) and Eq. (4). As shown in Fig. 3-B, the guidance illumination set follows distinct trajectories for each device in the uv-histogram space. To convert these trajectory differences into a device-aware embedding, we employ a lightweight CNN, the CFE encoder, consisting of four convolutional layers (with max pooling) followed by a two-layer MLP. This network encodes each device's locus histogram into an 8-dimensional CFE feature. The encoded CFE feature is then repeated along the u and v axes to match the resolution of the input histograms. Finally, it is concatenated with the input histograms (N_0, N_1) along the channel dimension and provided as input to the CCC generator network f, as shown in Fig. 3-A.

3.4. Imaginary Camera Augmentation

In prior illuminant estimation research, most data augmentation techniques (e.g., [1, 5, 27, 49]) rely on transferring ground-truth illuminant colors—randomly sampled from a given dataset—to other images within the same dataset (captured by the same camera) using chromatic adaptation. However, this approach is incompatible with our method, which is trained on raw images from different cameras, each with a distinct raw color space. Another augmentation approach [6] leverages camera-specific information and CCMs to perform raw-to-raw augmentation by transferring images from a source camera to a target camera. While promising, this method remains constrained by the limited diversity of training camera raw spaces.

To address these limitations, we propose a novel augmentation strategy that increases the diversity of camera characteristics, even with a limited set of training cameras. Specifically, we synthesize *imaginary* cameras by leveraging the CCMs of available training cameras. This expands the range of camera raw spaces encountered during training, significantly enhancing generalization.

Imaginary Camera Image Synthesis. Under the assumption of a single illuminant in the scene, the value of channel $c \in \{R, G, B\}$ at pixel x in the camera's raw space can be expressed as:

$$I_c(x) = \int S(\lambda, x) R(\lambda) Q_c(\lambda)\, d\lambda, \tag{11}$$

where $S(\cdot)$ and $R(\cdot)$ represent the spectral power distribution of the scene and illuminant at pixel x, respectively, and Q_c is the camera's spectral sensitivity for color channel c.

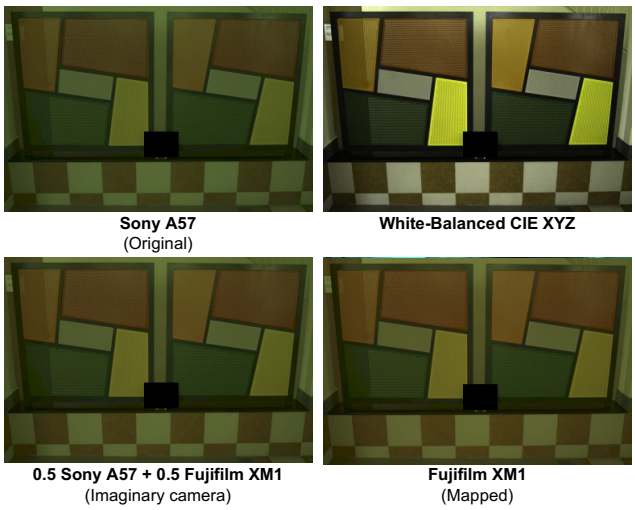

Figure 4. Visualization of our imaginary camera augmentation process. An image from the Sony A57 is white-balanced using the ground-truth illuminant, converted to CIE XYZ space, and mapped to the target camera's raw space. We illustrate two cases: mapping to the raw space of a real camera (Fujifilm XM1) and an imaginary camera. Brightness is adjusted for clarity.

The integral is computed over λ, corresponding to wavelengths in the visible light spectrum.

Since a camera's characteristics are defined by its spectral sensitivity function Q, an image captured by an *imaginary* camera, denoted as V, can be approximated by linearly combining the characteristics of cameras A and B with a ratio α, defined as:

$$\begin{aligned} I_c^V &= \int S(\lambda)R(\lambda)(\alpha Q_c^A + (1-\alpha)Q_c^B)(\lambda)\, d\lambda \\ &= \alpha I_c^A + (1-\alpha)I_c^B \ , \end{aligned} \tag{12}$$

where superscripts A, B, and V denote different cameras, including the imaginary camera, and $\alpha \in [0, 1]$ controls the contribution of each camera to the synthesized imaginary camera (omitting x for simplicity). As illustrated in Fig. 4, this approach allows mapping raw images to a specific target camera (e.g., Sony A57 with $\alpha = 1$) or to an imaginary camera (e.g., blending the Sony A57 and Fujifilm XM1 with a 0.5 to 0.5 ratio). Additionally, the ground truth illumination for the imaginary camera V can be synthesized as a linear combination of the ground truth illuminations of cameras A and B, weighted by α. For additional details on augmentation methods, please refer to the supplementary materials.

Derivation of the Imaginary Camera's CCM. Since CCMNet requires CCMs to encode CFE, it is also necessary to derive CCMs for the imaginary camera. Let us assume that cameras A, B, and the imaginary camera V observe a light source with a correlated color temperature CCT_{low}. Based on Eq. (9) and Eq. (12), the observed raw RGB values for camera V can be obtained as follows:

$$\begin{aligned} \text{CCM}_{low}^V\mathbf{X} = \mathbf{L}^V &= \alpha\mathbf{L}^A + (1-\alpha)\mathbf{L}^B \\ &= \alpha(\text{CCM}_{low}^A\mathbf{X}) + (1-\alpha)(\text{CCM}_{low}^B\mathbf{X}) \\ &= [\alpha\text{CCM}_{low}^A + (1-\alpha)\text{CCM}_{low}^B]\mathbf{X}, \end{aligned} \tag{13}$$

where the superscript V, A, B denotes the type of camera (omitted the subscript CCT_{low} for $\mathbf{L}$ and $\mathbf{X}$ for simplicity). As a result, the CCM_{low} for the imaginary camera V can be defined as:

$$\text{CCM}_{low}^V = \alpha\text{CCM}_{low}^A + (1-\alpha)\text{CCM}_{low}^B. \tag{14}$$

This relationship also holds for CCT_{high} and any arbitrary color temperature t within the range of low and high CCTs in the calibrated CCMs, as described in Eq. (10). We randomly select two cameras from the training dataset to generate an augmented set using the method outlined above. The augmented images and CCMs approximate the spectral sensitivity of the imaginary camera, enabling the CFE encoder to generalize to a wider range of cameras despite the limited number of training cameras.

4. Experiments

4.1. Experimental Setup

Training. The input image and camera-specific raw RGB illuminants (51 colors ranging from 2500K to 7500K, sampled at 100K intervals) are represented as 64×64 uv-histograms. The uv-ranges are empirically set to [-2.85, 2.85] for the input query image and [-0.5, 1.5] for CFE encoding.

We use the Intel-TAU [44], Gehler-Shi [57], NUS-8 [19], and Cube+ [9] datasets for training and testing. Each dataset includes images captured by distinct cameras, with no overlap between datasets. The number of cameras varies between one (Cube+) and eight (NUS-8).

Following the protocol in C5 [6], we adopt a leave-one-out cross-dataset evaluation approach, where the network is trained on all datasets except the test dataset. For instance, when validating on Gehler-Shi, the network is trained using Intel-TAU, NUS-8, and Cube+, ensuring no camera overlap between training and test datasets. We exclude the Sony-IMX subset of Intel-TAU due to the absence of CCM information, so Intel-TAU is used solely for training.

The mean angular error serves as the loss function during training. Additional details on batch size, epochs, other training hyperparameters, and model architecture are provided in the supplementary materials.

Data Augmentation. We augment the training data by selecting two cameras from the training datasets and applying

camera-to-camera mapping with random ratio interpolation to generate images and CCMs for imaginary cameras, as described in Sec. 3.4. The total number of augmented images matches the size of the original training set. Further details are provided in the supplementary materials.

Testing. For evaluation, we report commonly used error statistics: the mean, median, and tri-mean angular errors, along with the arithmetic mean of the top and bottom 25% angular errors between the predicted and ground truth illuminations.

4.2. Results

Results are presented in Table 1, where the first three tables show the main experimental results for three test datasets: Cube+, Gehler-Shi, and NUS-8. These results demonstrate that CCMNet achieves state-of-the-art performance across all datasets and metrics (see Fig. 5).

Unlike other learning-based models that report zero-shot results, DMCC retrains a target camera-specific network using calibrated matrices to transform training data (the Sony IMX-135 subset from the Intel-TAU dataset) into the test camera's color space. Similarly, C5 requires additional images from the test camera for guidance, making it difficult to determine the optimal number and content of these images.

In contrast, CCMNet achieves superior and more consistent results by leveraging CFE features from two pre-calibrated CCMs. CCMNet is simpler, more robust and does not require retraining for each test camera or the use of additional images. Notably, no data or CCMs from test cameras are used during training, ensuring true zero-shot generalization. Additional visual results are provided in the supplementary materials.

We also report results on the cross-sensor (CS) validation setup [3]. In this protocol, the network is trained on data from seven cameras in the NUS-8 dataset, excluding one as the test camera. This process is repeated for each of the eight cameras, and the results are averaged.

Following this protocol, we train CCMNet using data from Intel-TAU, Cube+, Gehler-Shi, and seven cameras from NUS-8 (excluding the test camera), aggregating results over eight iterations. As shown at the bottom of Table 1, CCMNet outperforms other methods under this evaluation protocol.

Another advantage of CCMNet is its lightweight design. Since the CFE feature is fixed once the camera device is determined, it only needs to be extracted once for a new camera and can be reused thereafter. As a result, CCMNet's size and computational cost depend solely on the backbone f, making it significantly more efficient than the C5 model, which requires 6–8 additional histogram encoders.

As shown in the first table of Table 1, the C5 model (with an additional 8 histograms, $m = 9$) requires approximately 2.09 MB of storage, whereas CCMNet, excluding the CFE encoder, occupies only 1.05 MB—almost half the size. This highlights CCMNet's compactness, making it particularly well-suited for integration into ISP modules, where efficient resource utilization is crucial.

Gehler-Shi [57]	**Mean**	**Med.**	**Tri.**	**B.25%**	**W.25%**	**Size(MB)**
2nd-order Gray-Edge [61]	5.13	4.44	4.62	2.11	9.26	-
Shades-of-Gray [23]	4.93	4.01	4.23	1.14	10.20	-
PCA-based B/W Colors [19]	3.52	2.14	2.47	0.50	8.74	-
ASM [8]	3.80	2.40	2.70	-	-	-
Woo *et al.* [62]	4.30	2.86	3.31	0.71	10.14	-
Grayness Index [56]	3.07	1.87	2.16	0.43	7.62	-
Cross-dataset CC [43]	2.87	2.21	-	-	-	-
Quasi-Unsupervised CC [14]	3.46	2.23	-	-	-	622
SIIE [3]	2.77	1.93	-	0.55	6.53	10.3
FFCC [13]	2.95	2.19	2.35	0.57	6.75	0.22
C5 ($m = 7$) [6]	2.36	1.61	1.74	0.44	5.60	1.74
C5 ($m = 9$) [6]	2.50	1.99	2.03	0.53	5.46	2.09
CCMNet (Ours)	**2.23**	**1.53**	**1.62**	**0.36**	**5.46**	1.05

Cube+ [9]	**Mean**	**Med.**	**Tri.**	**B.25%**	**W.25%**
Gray-world [18]	3.52	2.55	2.82	0.60	7.98
1st-order Gray-Edge [61]	3.06	2.05	2.32	0.55	7.22
2nd-order Gray-Edge [61]	3.28	2.34	2.58	0.66	7.44
Shades-of-Gray [23]	3.22	2.12	2.44	0.43	7.77
Cross-dataset CC [43]	2.47	1.94	-	-	-
Quasi-Unsupervised CC [14]	2.69	1.76	2.00	0.49	6.45
SIIE [3]	2.14	1.44	-	0.44	5.06
FFCC [13]	2.69	1.89	2.08	0.46	6.31
DMCC [66]	2.23	1.63	1.78	0.49	4.95
C5 ($m = 7$) [6]	1.87	1.27	1.40	0.41	4.36
C5 ($m = 9$) [6]	1.92	1.32	1.46	0.44	4.44
CCMNet (Ours)	**1.68**	**1.16**	**1.26**	**0.38**	**3.89**

NUS-8 [19]	**Mean**	**Med.**	**Tri.**	**B.25%**	**W.25%**
Gray-world [18]	4.59	3.46	3.81	1.16	9.85
Shades-of-Gray [23]	3.67	2.94	3.03	0.98	7.75
Local Surface Reflectance [28]	3.45	2.51	2.70	0.98	7.32
PCA-based B/W Colors [19]	2.93	2.33	2.42	0.78	6.13
Grayness Index [56]	2.91	1.97	2.13	0.56	6.67
Cross-dataset CC [43]	3.08	2.24	-	-	-
Quasi-Unsupervised CC [14]	3.00	2.25	-	-	-
FFCC [13]	2.87	2.14	2.30	0.71	6.23
C5 ($m = 7$) [6]	2.68	2.00	2.14	0.66	5.90
C5 ($m = 9$) [6]	2.54	1.90	2.02	0.61	5.61
CCMNet (Ours)	**2.32**	**1.71**	**1.83**	**0.53**	**5.18**

NUS-8 (CS) [19]	**Mean**	**Med.**	**Tri.**	**B.25%**	**W.25%**
DMCC (CS) [66]	2.80	2.12	2.25	0.74	5.88
SIIE (CS) [3]	2.05	1.50	-	0.52	4.48
C5 ($m = 9$, CS) [6]	1.77	1.37	1.46	0.48	3.75
CCMNet (Ours, CS)	**1.71**	**1.31**	**1.40**	**0.48**	**3.62**

Table 1. Experimental results on three benchmark datasets. CCMNet achieves the best performance across all metrics on various datasets, including additional cross-sensor (CS) validation protocol. For C5 model, m represents the total number of images used, including both the query image and additional images.

4.3. Generalization with SIIE Backbone

We further explore using CFE with SIIE [3]. SIIE learns a 3×3 matrix by processing the raw image uv-histogram to map raw colors to a color *working* space. In this experiment, we replace C5 with SIIE as our backbone. Specifically, we input our CFE-concatenated uv-histograms, augmented with the imaginary camera transformation, into the SIIE backbone. As shown in Table 2, the best performance

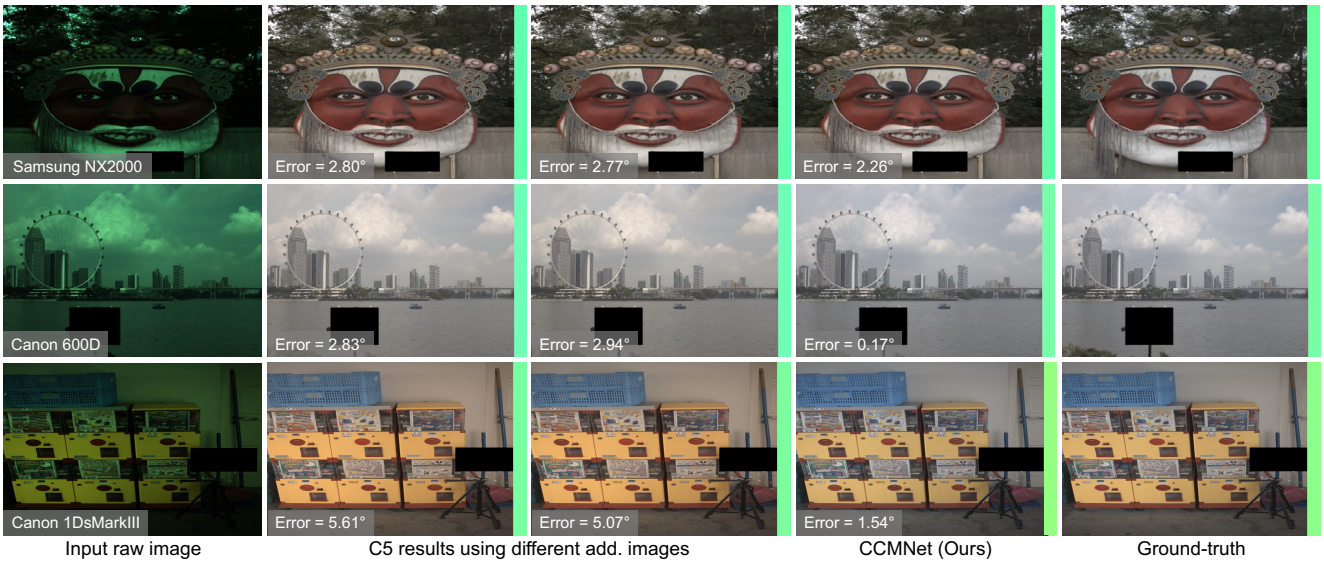

Figure 5. Visual comparison of the results from C5 [6] with different additional image sets (second, third column) and CCMNet (fourth column). While C5 relies on additional images, CCMNet is optimized for fixed CFE guidance, ensuring consistent performance.

Model	Cube+	Gehler-Shi	NUS-8
SIIE [3]	3.39	3.67	3.52
w/ CFE	2.60	3.62	3.36
w/ aug.	2.43	3.12	3.00
w/ CFE & aug.	**1.91**	**2.99**	**2.94**

Table 2. Generalization with the SIIE [3] backbone. Reported results show the mean angular error.

Model	Aug. method	Cube+	Gehler-Shi	NUS-8
Backbone f	w/o aug.	2.22	2.79	2.88
	$\alpha = 1$	1.94	2.87	2.50
	$0 \leq \alpha \leq 1$	1.78	2.53	2.54
CCMNet (f + CFE)	w/o aug.	2.23	2.74	2.70
	$\alpha = 1$	1.86	2.34	2.45
	$0 \leq \alpha \leq 1$	**1.68**	**2.23**	**2.32**

Table 3. Ablation studies on the impact of the CFE encoder and different augmentation strategies. The reported results are the mean angular error.

(MAE) is achieved when both CFE and augmentation are applied, confirming that CCMNet generalizes to different backbones utilizing uv-histograms.

4.4. Ablation Studies

Table 3 presents the performance of the backbone f and CCMNet trained under three setups: without augmentation (w/o aug), with augmentation at $\alpha = 1$, and with augmentation for $0 \leq \alpha \leq 1$. Architecturally, the backbone f mirrors the $m = 1$ structure from C5 [6], excluding additional images and encoders. Setting $\alpha = 1$ in the augmentation process replicates the camera-mapping strategy used in C5.

The training data is halved for the w/o aug. setup, and the number of iterations is doubled to ensure the same number of model updates as in the other experiments. The results indicate that the CFE encoder in CCMNet and the imaginary camera augmentation play crucial roles in the cross-camera color constancy task.

5. Conclusion and Discussion

In this paper, we propose CCMNet, a lightweight and efficient method for cross-camera color constancy that leverages pre-calibrated CCMs available in camera ISPs. The model utilizes these CCMs, which map a camera's raw color space to the device-independent CIE XYZ color space or vice versa, to encode the camera-specific illumination locus into a guidance embedding. This feature, termed CFE, directs a hypernetwork to quickly adapt to unseen cameras during testing, enabling the generation of appropriate filters and biases while achieving superior performance compared to previous methods.

By taking advantage of the linearity of CCM operations, the proposed imaginary camera augmentation technique allows the model to learn a broader range of virtual camera response functions during training, significantly improving CCMNet's generalization capability.

While most cameras include calibrated raw-to-XYZ CCMs in their ISPs and DNG files, some smartphones may not provide accurate CCMs in their DNGs. Instead, these devices often include a single fixed matrix to convert raw images to linear sRGB. This limitation could hinder our method's ability to process DNG files from such devices or necessitate an additional conversion step to adapt to the raw-to-linear sRGB matrix.

CCMNet: Leveraging Calibrated Color Correction Matrices for Cross-Camera Color Constancy

Supplementary Material

A. CCMs & CCTs Extraction

In this section, we describe the methodology used to extract the color correction matrices, low and high correlated color temperatures (CCT_{low}, CCT_{high}) information utilized in our approach. Since CCMs and their correlated CCTs are camera-dependent, they can be extracted once and remain consistent across all images captured by the same camera.

To extract the CCMs and CCTs of a specific camera, we followed these steps. First, to ensure consistency in data processing, we converted all raw images to the DNG format using Adobe DNG Converter, instead of relying on camera-specific raw file extensions. Second, we extracted metadata from the DNG files using ExifTool, specifically retrieving *ColorMatrix1*, *ColorMatrix2*, *ForwardMatrix1*, and *ForwardMatrix2*. These matrices were then used for our imaginary camera augmentation and for testing on previously unseen cameras during training. For convenience, we will refer to *ColorMatrix* and *ForwardMatrix* as CM and FM, respectively, throughout this supplementary material.

Fig. 6 illustrates the relationships between color spaces and the transformation matrices involved. As shown, the FM transforms white-balanced camera raw colors to the CIE XYZ color space, while the CM converts from CIE XYZ to the camera's native raw color space under a specific illuminant. The suffixes '1' and '2' in the matrix names indicate calibration for illuminants 1 and 2, corresponding to standard illuminant A and D65, respectively. Accordingly, we define CCT_{low} and CCT_{high} as the color temperatures of illuminant A (2856K) and D65 (6504K) and use these values for CCM interpolation, as described in Eq. (10) in the main paper.

As defined in Eq. (9) in the main paper, the CCM_{low} and CCM_{high} matrices used throughout this work correspond to CM1 and CM2, respectively. Additionally, CM1, CM2, FM1, FM2 are used in the imaginary camera augmentation process described in Sec. C.

B. Details of the CFE Encoding Process

In this section, we provide additional details on the CFE (Camera Fingerprint Embedding) encoding process described in Sec. 3.3.

Further explanations. As shown in Fig. 7, the essence of what CFE fundamentally encodes is the color trajectory on the CIE xy-plane within the correlated color temperature (CCT) range of 2500K–7500K. These colors correspond to the light emitted by a black body at a given CCT and are intrinsic, invariant values. However, due to differences in the spectral sensitivity of imaging sensors, each device *observes* these reference colors as distinct loci. These trajectories inherently represent the unique color characteristics of each device.

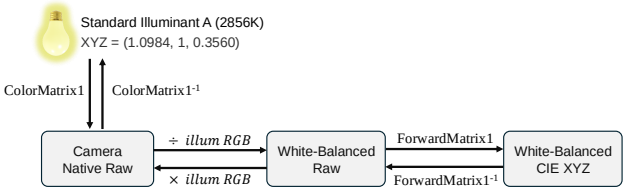

Figure 6. A schematic diagram illustrating the use of *ColorMatrix* and *ForwardMatrix*. The *ForwardMatrix* (FM) transforms white-balanced raw data into the CIE XYZ color space, while the *ColorMatrix* (CM) converts CIE XYZ values of a standard light source into the camera's native raw color space. FM1 and CM1 are calibrated for standard illuminant A (2856K), and FM2 and CM2 are calibrated for the D65 illuminant (6504K).

We leverage the fact that this *observation* process is precomputed for two illuminants during the ISP manufacturing stage and recorded as matrices (CCMs). By interpolating the two matrices, CCM_{low} and CCM_{high}, and then applying to the Planckian XYZ locus, we replace the observation process for each device. The resulting device-specific locus is then converted into a histogram, which is subsequently encoded into a CFE feature that captures the *fingerprint* of each camera using a CNN-based CFE encoder.

Due to this design approach of the CFE feature, the CCMNet leverages CFE as guidance, enabling it to infer and adapt to the color space of a previously unseen camera. This allows the model to learn a generalized approach to illuminant color estimation without requiring explicit training on every individual camera.

Technical details. For the XYZ locus corresponding to color temperatures from 2500K to 7500K, we used the `colour.temperature.CCT_to_xy` function from the `colour` Python library. A total of 51 chromaticity coordinates were sampled at 100K intervals, ranging from 2500K to 7500K.

As mentioned in the main paper, the sampled XYZ locus was transformed into the camera's native raw RGB space by interpolating between CM1 and CM2. This was further converted into a histogram with 64 bins, within the uv range of [-0.5, 1.5]. The resulting $64\times64\times1$ histogram was processed by the CFE encoder, which outputs an 8-dimensional embedding vector. The CFE encoder consists

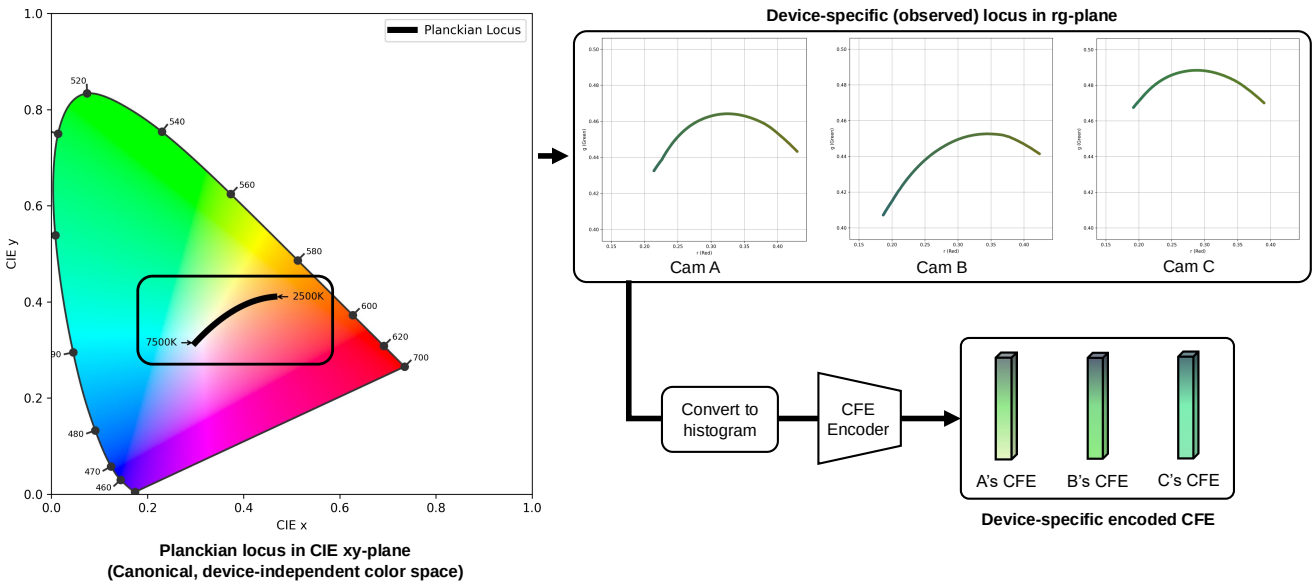

Figure 7. Detailed visualization of CFE encoding process. As mentioned in the main paper, the camera's *fingerprint* is derived by converting the reference CIE XYZ colors (locus) along the correlated color temperature (CCT) range of 2500K–7500K into the corresponding RGB locus as *observed* by each device, followed by an encoding process. Due to this characteristic, the CFE feature inherently reflects the color characteristics induced by each camera's spectral sensitivity.

of four `DoubleConvBlocks` followed by a projection head. Each `DoubleConvBlock` processes the input by applying two convolutional layers, each with a kernel size of 3×3, a stride of 1, and a padding of 1, followed by a LeakyReLU activation. This is then followed by a 2×2 max-pooling layer and batch normalization. The projection head flattens the feature map and maps it to an 8-dimensional embedding vector using an MLP with two hidden layers.

C. Camera-to-Camera Mapping

In Sec. 3.4 of the main paper, we introduced our imaginary camera augmentation, which assumes two versions of the same image in the camera's native raw RGB space. To satisfy this condition, we perform a camera-to-camera mapping inspired by [6]. In this section, we provide a detailed explanation of the camera-to-camera mapping process used in our work. Specifically, in Sec. C.1, we explain the process of computing the correlated color temperature (CCT) of a light source in the target camera's native raw RGB space. Then, in Sec. C.2, we describe how to generate a pool of white-balanced, camera-independent XYZ images using the RGB values of the light source and the corresponding CCT. In Sec. C.3, we describe the process of generating a device-specific illumination pool for random sampling. Finally, Sec. C.4 explains our camera-to-camera mapping, which presents a reference image in two different camera-native raw RGB spaces. The reference image is sampled from the XYZ image pool, while the illumination is sampled from the augmented ground-truth (GT) illumination pool of each camera. The overall process is visualized in Fig. 8.

While our camera-to-camera mapping is inspired by the C5 augmentation approach [6], it differs in the following ways. First, we remove C5's restriction that limits sampling from the illumination pool to similar scenes with matching capturing settings (e.g., ISO, exposure time) and illumination CCT. Specifically, in C5, both the sampled scene image from the CIE XYZ space and the sampled illuminant from the target camera were required to have similar capturing settings and CCT. In contrast, our approach removes this constraint, eliminating the need to rely on capturing settings and allowing for greater diversity in augmentation. Additionally, instead of sampling from a fitted cubic polynomial based on the target camera's illuminant samples, we use a fitted cubic polynomial based on the illuminant values from the source camera's dataset (i.e., the camera from which the reference XYZ image was taken). The sampled illuminant is then transferred to the CIE XYZ space using the inverse of the source camera's CM, followed by a transformation of these CIE XYZ illuminant values into the native raw RGB space of the target camera.

C.1. Illumination RGB to CCT Conversion

The illuminant estimation datasets used in the main paper provide GT illumination RGB labels for each scene in the camera's native raw RGB space. According to the Adobe DNG specification, given CM1 and CM2 (extracted for each camera as described in Sec. A), along with the GT il-

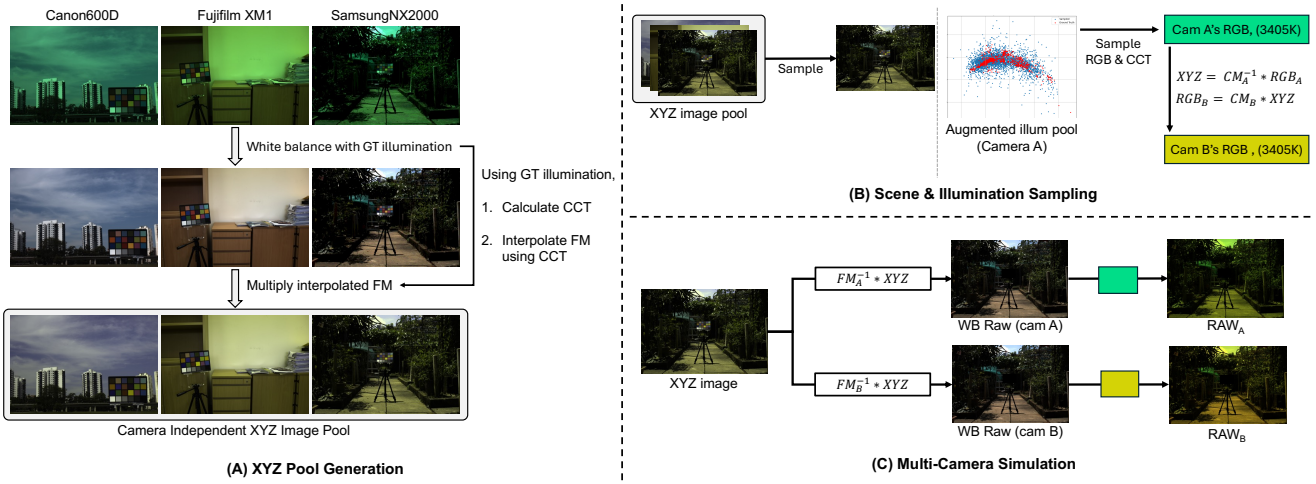

Figure 8. Overall process of camera-to-camera mapping. In (A), subsets of images taken by different cameras from multiple datasets are white-balanced using the corresponding ground-truth illuminants, and the *ForwardMatrix* is used to convert them to the CIE XYZ space, creating the XYZ image pool. In (B), a reference image is sampled from the pool, and an illumination color is sampled from the augmented illumination pool of the source camera (Camera A) that originally captured the image. The sampled illumination is then mapped to the native RGB space of a randomly selected target camera (Camera B) using the *ColorMatrix*. Finally, in (C), the XYZ image is transformed into the white-balanced color space of Cameras A and B using the inverse of their respective *ForwardMatrices*, and illumination casting is applied by multiplying the images with the illumination RGB values of each camera space.

lumination RGB, the CCT and CIE XYZ values of the light source can be computed using Algorithm 1.

Algorithm 1 Conversion of Illuminant Raw RGB to CCT and XYZ Coordinates

```
function CAMNTRL_TO_XYZ(illum, cm1, cm2)
    xy = [0.3127, 0.3290]
    while True do
        cct = colour.temperature.xy_to_CCT(xy)
        color_matrix = interpolate_ccm(cct, cm1, cm2)
        color_matrix_inv = np.linalg.inv(color_matrix)
        xyz = np.dot(color_matrix_inv, illum)
        X, Y, Z = xyz
        xy_new = [X / (X + Y + Z), Y / (X + Y + Z)]
        if np.allclose(xy, xy_new, atol=1e-6) then
            return xyz, cct
        end if
        xy = xy_new
    end while
end function
```

The algorithm iteratively estimates the CCT and converts illuminant RGB values to the CIE XYZ space. Using metadata such as CM1 and CM2, it interpolates the appropriate color correction matrix for the estimated CCT and applies it to transform the input illumination into the CIE XYZ space. The resulting XYZ coordinates and CCT values are then used either to generate the camera-independent XYZ image pool in Sec. C.2 or to transform the illumination into the target camera RGB space in Sec. C.4.

C.2. Unified XYZ Image Pool Generation

In this section, we describe the process of creating an XYZ image pool for camera-to-camera mapping by converting images captured by various cameras into the device-independent XYZ color space. The process involves two main steps: (1) white balancing with GT labels, and (2) transforming to the CIE XYZ color space using the *ForwardMatrix* (FM). Refer to Fig. 6 and Fig. 8-(A).

As explained in the main paper, we use multiple datasets captured by various cameras, each including GT illumination labels that enable accurate white balancing of images in the camera's native raw RGB space. As described in Sec. A, we extract FM1 and FM2 for each camera. Using the CCT of the GT illumination, we interpolate between FM1 and FM2 to transform the white-balanced images into the XYZ color space. The CCT is computed from the GT illumination RGB using the method detailed in Sec. C.1.

This process mitigates the dependency on camera specifications, and in theory, the images are independent of camera models and illumination conditions. By aggregating these images, we construct a unified XYZ image pool that serves as the foundation for camera-to-camera mapping.

C.3. Camera-specific Illumination Pool Generation

Next, we generate an illumination pool for each camera. While it is possible to use only the GT illuminations, we adopt the augmentation method proposed in [6] to enhance generality and diversity. This method involves fitting a cubic polynomial to the GT illuminations for each camera

and then introducing random shifts to augment the illuminations. For further details, please refer to the supplementary material of [6]. On the right side of Fig. 8-(B), we show a plot of the illumination pool for a specific camera (Camera A). In this plot, the red points represent the GT illumination labels extracted from the dataset, while the blue points correspond to the augmented illuminations.

C.4. Camera-to-Camera Image Synthesis

In this section, we describe a camera-to-camera mapping method that simulates the same scene as if it were captured by two different cameras, using the image pool from Sec. C.2 and the illumination pool from Sec. C.3. See Fig. 8-(B) and (C).

Scene and Illumination Sampling & Mapping. First, a scene is randomly selected from the XYZ image pool. Next, an illumination is randomly sampled from the illumination pool of the source camera that captured the selected scene. This sampled illumination is then transformed into the native raw color space of a randomly selected target camera from the set of cameras used (see Fig. 8-(B)). As illustrated in Fig. 6, the XYZ values of the sampled illumination are computed by applying the inverse of the source camera's *ColorMatrix* (CM). These XYZ values are then multiplied by the target camera's CM to obtain the native raw color of the illumination in the target camera's color space. The interpolation of each camera's CM is based on the CCT of the illumination, which is calculated using the steps described in Sec. C.1.

Synthesizing Paired Scene from Two Cameras. Finally, as illustrated in Fig. 8-(C), we generate two raw images of the sampled scene, as if it were captured by the selected two cameras under the same sampled illumination. As shown in Fig. 6, the white-balanced XYZ image is transferred to the cameras' native raw space in two steps. First, using the same CCT employed during CM interpolation in illumination mapping, the FMs of cameras A and B are interpolated, and their inverses are applied to the XYZ image. This step produces two white-balanced raw images, one for each camera. Next, the camera-native illumination RGB values–sampled from camera A and mapped to camera B as described in previous paragraph–are multiplied with these raw images. The resulting image pair simulates the same scene and lighting conditions as captured by two different cameras, all derived from a single XYZ image.

D. Imaginary Camera Augmentation Visualizations

Here, we provide additional visualizations of the imaginary camera augmentation. As shown in Fig. 9, Imaginary Camera Augmentation simulates images captured by virtual cameras that interpolate the properties of two real-world devices. This data augmentation technique also interpolates the CCMs at the same ratios to generate the CCMs for these virtual cameras.

E. Experimental Setup

As mentioned in the main paper, the backbone f uses the standard U-Net-like architecture from C5 [6]. However, unlike C5, we do not use additional images from the test camera, so no extra encoders are employed. Instead, we use a single Encoder-Decoder U-Net architecture. The encoder and decoder are connected via skip connections, with each consisting of four `DoubleConv` layers. In the encoder, each `DoubleConv` layer is followed by max pooling, while in the decoder, feature upsampling and skip connections are applied before each `DoubleConv` layer.

The batch size was set to 16, and training was conducted over 50 epochs with an initial learning rate of 5×10^{-4}. A learning rate decay of 0.5 was applied at epoch 25. The Adam optimizer [42] was used for training.

For data augmentation, camera-to-camera mapping and imaginary camera augmentation are applied exclusively using the camera subsets from the training datasets, excluding the test dataset. For instance, when evaluating the Cube+ dataset, the augmented dataset used for model training is generated from images and CCMs from the Gehler-Shi [57], NUS-8 [19], and Intel-TAU [44] datasets.

F. Additional Results

We present additional visualization results in Fig. 10 and Fig. 11. As shown in Fig. 10, CCMNet achieves satisfactory accuracy across various scenes captured by a camera it has never encountered during training. In Fig. 11, we demonstrate that CCMNet maintains robust accuracy across a set of unseen cameras.

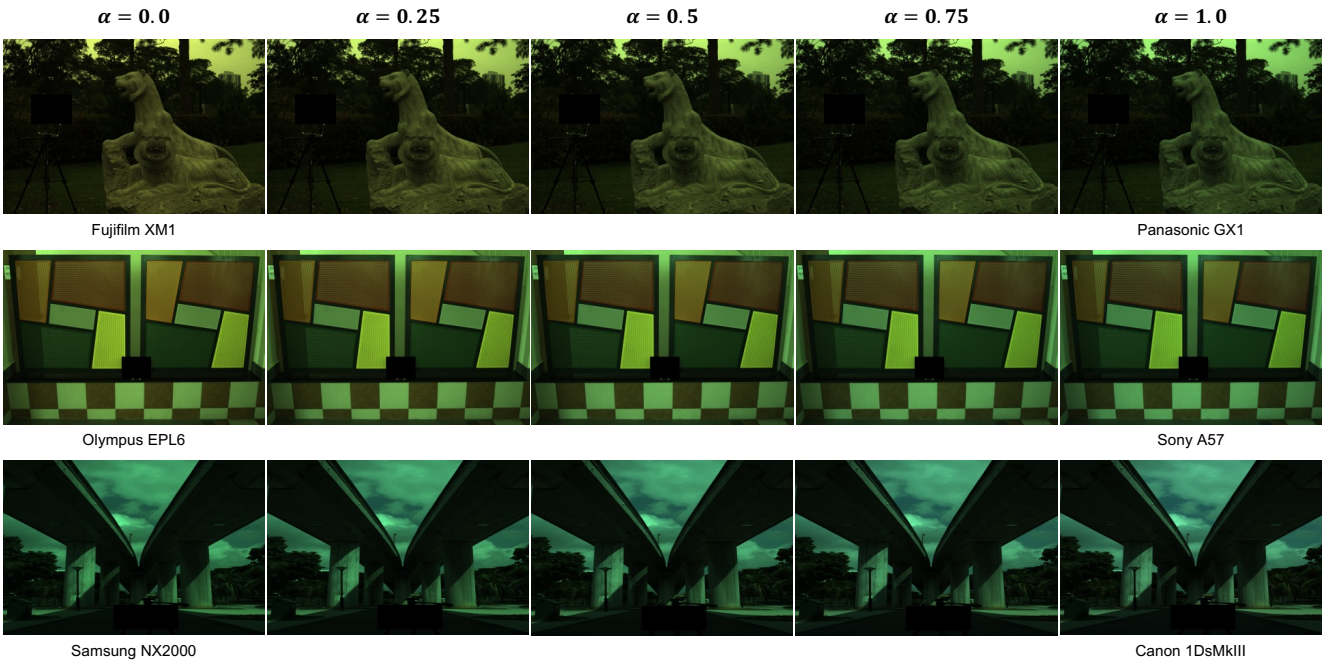

Figure 9. Results of our imaginary camera augmentation. In each row, the leftmost and rightmost images represent the source and target camera images generated using the method described in Sec. C, while the three middle images represent those produced by the *imaginary* camera, generated by interpolating between the two devices at ratios of 0.25, 0.5, and 0.75, respectively. As explained in Sec. 3.4 of the main paper, the CCMs of the imaginary cameras are interpolated using the same alpha values applied during image interpolation, and the resulting CFE embeddings are generated for training. Brightness is adjusted for visibility.

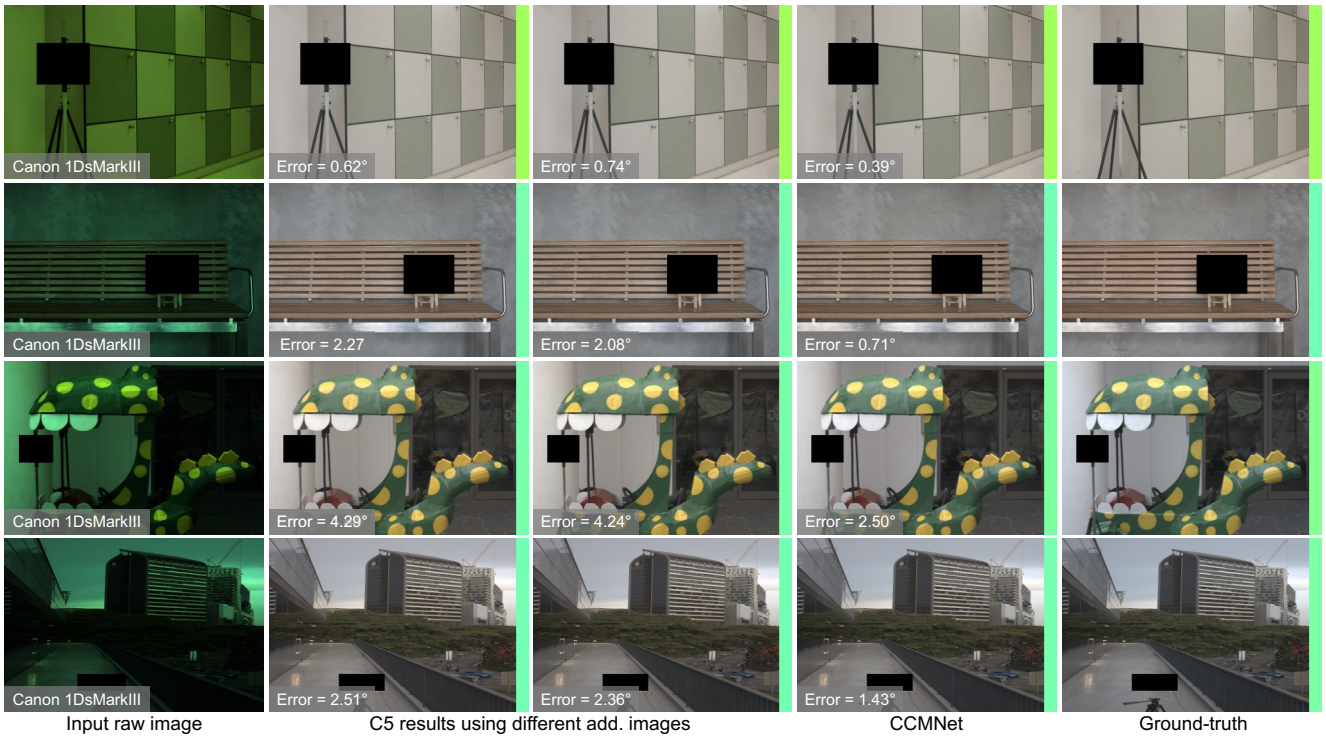

Figure 10. Additional results for Canon EOS 1Ds Mark III. CCMNet demonstrates superior performance on various scenes captured by unseen camera. Notably, CCMNet has never been exposed to any images or the CCM of the Canon 1Ds Mark III during training.

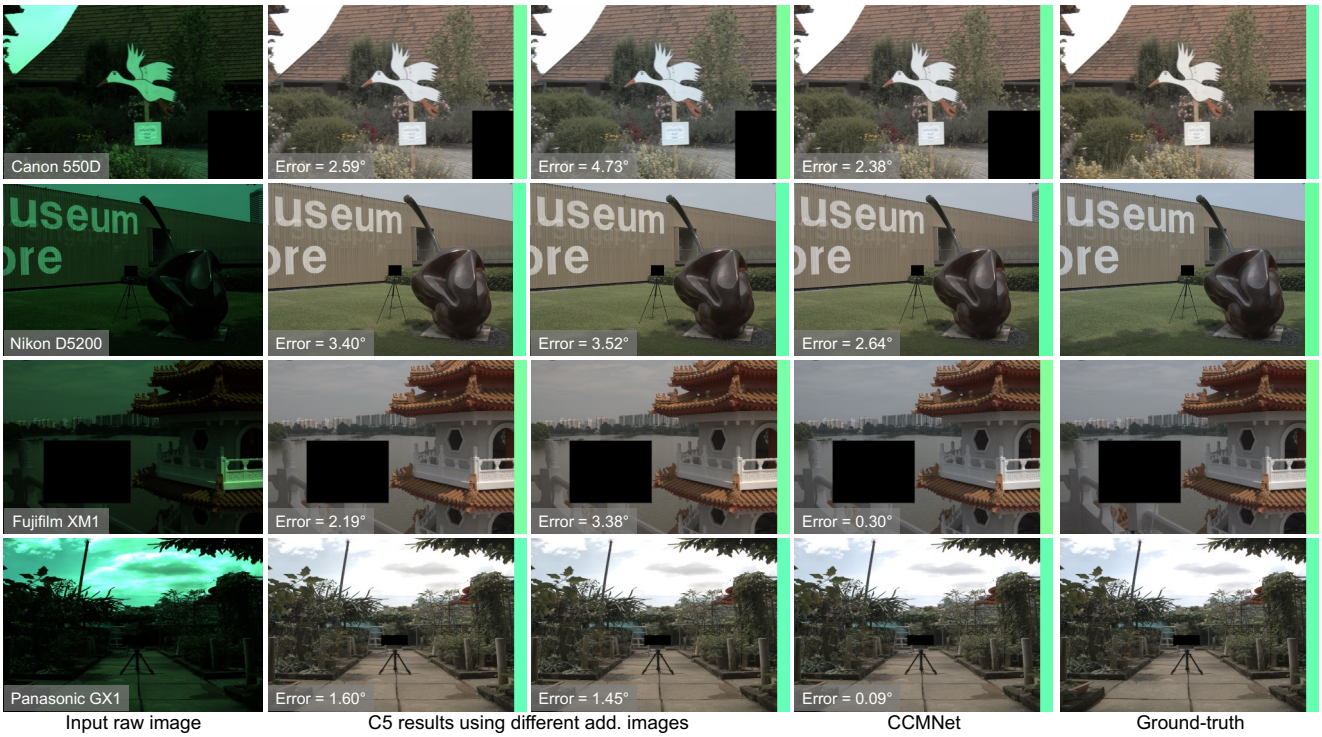

Figure 11. Additional results for various cameras show that CCMNet exhibits robust performance across a range of unseen cameras. Importantly, it has not been exposed to any images or CCMs from the cameras shown in the figure during training.

References

[1] Abdelrahman Abdelhamed, Abhijith Punnappurath, and Michael S Brown. Leveraging the availability of two cameras for illuminant estimation. In *CVPR*, 2021. 5

[2] Mahmoud Afifi and Abdullah Abuolaim. Semi-supervised raw-to-raw mapping. In *BMVC*, 2021. 1

[3] Mahmoud Afifi and Michael S Brown. Sensor-independent illumination estimation for DNN models. In *BMVC*, 2019. 1, 2, 7, 8

[4] Mahmoud Afifi, Brian Price, Scott Cohen, and Michael S Brown. When color constancy goes wrong: Correcting improperly white-balanced images. In *CVPR*, pages 1535–1544, 2019. 1

[5] Mahmoud Afifi, A. Abdelhamed, Abdullah Abuolaim, Abhijith Punnappurath, and M. S. Brown. CIE XYZ Net: Unprocessing images for low-level computer vision tasks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44:4688–4700, 2020. 5

[6] Mahmoud Afifi, Jonathan T Barron, Chloe LeGendre, Yun-Ta Tsai, and Francois Bleibel. Cross-camera convolutional color constancy. In *ICCV*, 2021. 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12

[7] Mahmoud Afifi, Zhenhua Hu, and Liang Liang. Optimizing illuminant estimation in dual-exposure HDR imaging. In *ECCV*, 2025. 1, 2

[8] Arash Akbarinia and C. Alejandro Párraga. Colour constancy beyond the classical receptive field. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40: 2081–2094, 2018. 7

[9] Nikola Banić, Karlo Koščević, and Sven Lončarić. Unsupervised learning for color constancy. *arXiv preprint arXiv:1712.00436*, 2017. 6, 7

[10] Kobus Barnard. Computational color constancy: Taking theory into practice. 1995. 1, 2

[11] Kobus Barnard, Lindsay Martin, Adam Coath, and Brian Funt. A comparison of computational color constancy algorithms–part II: Experiments with image data. *IEEE Transactions on Image Processing*, 11(9):985–996, 2002. 1

[12] Jonathan T Barron. Convolutional color constancy. In *ICCV*, 2015. 2, 3, 4

[13] Jonathan T Barron and Yun-Ta Tsai. Fast Fourier color constancy. In *CVPR*, 2017. 2, 3, 4, 7

[14] Simone Bianco and Claudio Cusano. Quasi-unsupervised color constancy. In *CVPR*, 2019. 2, 7

[15] Simone Bianco, Arcangelo R Bruna, Filippo Naccari, and Raimondo Schettini. Color correction pipeline optimization for digital cameras. *Journal of Electronic Imaging*, 22(2): 023014–023014, 2013. 3

[16] Simone Bianco, Claudio Cusano, and Raimondo Schettini. Color constancy using CNNs. In *CVPRW*, 2015. 2

[17] MichaelS Brown. Color processing for digital cameras. *Fundamentals and Applications of Colour Engineering*, pages 81–98, 2023. 1, 2

[18] Gershon Buchsbaum. A spatial processor model for object

colour perception. *Journal of the Franklin institute*, 310(1): 1–26, 1980. 2, 7

[19] Dongliang Cheng, Dilip K Prasad, and Michael S Brown. Illuminant estimation for color constancy: Why spatial-domain methods work and the role of the color distribution. *JOSA A*, 31(5):1049–1058, 2014. 2, 6, 7, 12

[20] Dongliang Cheng, Brian Price, Scott Cohen, and Michael S Brown. Beyond white: Ground truth colors for color constancy correction. In *CVPR*, 2015. 1

[21] Mauricio Delbracio, Damien Kelly, Michael S Brown, and Peyman Milanfar. Mobile computational photography: A tour. *Annual review of vision science*, 7(1):571–604, 2021. 1, 2

[22] Graham D Finlayson and Steven D Hordley. Color constancy at a pixel. *JOSA A*, 18(2):253–264, 2001. 3

[23] Graham D Finlayson and Elisabetta Trezzi. Shades of gray and colour constancy. In *Color and Imaging Conference*, 2004. 2, 7

[24] Graham D Finlayson and Yuteng Zhu. Designing color filters that make cameras more colorimetric. *IEEE Transactions on Image Processing*, 30:853–867, 2020. 3

[25] Graham D Finlayson, Michal Mackiewicz, and Anya Hurlbert. Color correction using root-polynomial regression. *IEEE Transactions on Image Processing*, 24(5):1460–1470, 2015. 3

[26] Toadere Florin. Color processing in a digital camera pipeline. In *Advanced Topics in Optoelectronics, Microelectronics, and Nanotechnologies IV*, pages 223–227, 2009. 1, 2

[27] Damien Fourure, Rémi Emonet, Elisa Fromont, Damien Muselet, Alain Trémeau, and Christian Wolf. Mixed pooling neural networks for color constancy. In *ICIP*, 2016. 5

[28] Shaobing Gao, Wangwang Han, Kaifu Yang, Chaoyi Li, and Y. Li. Efficient color constancy with local surface reflectance statistics. In *European Conference on Computer Vision*, 2014. 7

[29] Peter Vincent Gehler, Carsten Rother, Andrew Blake, Tom Minka, and Toby Sharp. Bayesian color constancy revisited. In *CVPR*, 2008. 2

[30] Arjan Gijsenij, Theo Gevers, and Joost Van De Weijer. Generalized gamut mapping using image derivative structures for color constancy. *International Journal of Computer Vision*, 86(2-3):127–139, 2010. 1, 2

[31] Arjan Gijsenij, Theo Gevers, and Joost Van De Weijer. Computational color constancy: Survey and experiments. *IEEE Transactions on Image Processing*, 20(9):2475–2489, 2011. 1

[32] Arjan Gijsenij, Theo Gevers, and Joost Van De Weijer. Improving color constancy by photometric edge weighting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(5):918–929, 2011. 2

[33] Samuel W Hasinoff, Dillon Sharlet, Ryan Geiss, Andrew Adams, Jonathan T Barron, Florian Kainz, Jiawen Chen, and Marc Levoy. Burst photography for high dynamic range and low-light imaging on mobile cameras. *ACM Transactions on Graphics (ToG)*, 35(6):1–12, 2016. 2

[34] Daniel Hernandez-Juarez, Sarah Parisot, Benjamin Busam, Ales Leonardis, Gregory Slabaugh, and Steven McDonagh. A multi-hypothesis approach to color constancy. In *CVPR*, 2020. 2

[35] Guowei Hong, M Ronnier Luo, and Peter A Rhodes. A study of digital camera colorimetric characterization based on polynomial modeling. *Color Research & Application*, 26 (1):76–84, 2001. 3

[36] Yuanming Hu, Baoyuan Wang, and Stephen Lin. FC4: Fully convolutional color constancy with confidence-weighted pooling. In *CVPR*, 2017. 2

[37] Paul M Hubel, Graham D Finlayson, and Steven D Hordley. White point estimation using color by convolution, 2007. US Patent 7,200,264. 2

[38] Po-Chieh Hung. Colorimetric calibration in electronic imaging devices using a look-up-table model and interpolations. *Journal of Electronic imaging*, 2(1):53–61, 1993. 3

[39] Adobe Systems Incorporated. Digital negative (DNG) specification. 2023. 2, 3

[40] Hakki Can Karaimer and Michael S Brown. A software platform for manipulating the camera imaging pipeline. In *ECCV*, 2016. 2, 3

[41] Hakki Can Karaimer and Michael S Brown. Improving color reproduction accuracy on cameras. In *CVPR*, 2018. 2, 3

[42] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 12

[43] Samu Koskinen12, Dan Yang, and Joni-Kristian Kämäräinen. Cross-dataset color constancy revisited using sensor-to-sensor transfer. BMVC, 2020. 7

[44] Firas Laakom, Jenni Raitoharju, Jarno Nikkanen, Alexandros Iosifidis, and Moncef Gabbouj. Intel-tau: A color constancy dataset. *IEEE access*, 9:39560–39567, 2021. 6, 12

[45] Edwin H Land. The retinex theory of color vision. *Scientific american*, 237(6):108–129, 1977. 2

[46] Bing Li, Haina Qin, Weihua Xiong, Yangxi Li, Songhe Feng, Weiming Hu, and Stephen Maybank. Ranking-based color constancy with limited training samples. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(10): 12304–12320, 2023. 2

[47] Shuwei Li and Robby T Tan. NightCC: Nighttime color constancy via adaptive channel masking. In *CVPRW*, 2024. 2

[48] Yi-Chen Lo, Chia-Che Chang, Hsuan-Chao Chiu, Yu-Hao Huang, Chia-Ping Chen, Yu-Lin Chang, and Kevin Jou. CLCC: Contrastive learning for color constancy. In *CVPR*, 2021. 1, 2

[49] Zhongyu Lou, Theo Gevers, Ninghang Hu, Marcel P Lucassen, et al. Color constancy by deep learning. In *BMVC*, 2015. 5

[50] Steven McDonagh, Sarah Parisot, Fengwei Zhou, Xing Zhang, Ales Leonardis, Zhenguo Li, and Gregory Slabaugh. Formulating camera-adaptive color constancy as a few-shot meta-learning problem. *arXiv preprint arXiv:1811.11788*, 2018. 2

[51] Jon S McElvain and Walter Gish. Camera color correction using two-dimensional transforms. In *Color and Imaging Conference*, 2013. 3

[52] Rang Nguyen, Dilip K Prasad, and Michael S Brown. Raw-to-raw: Mapping between image sensor color responses. In *CVPR*, 2014. 1

[53] Seoung Wug Oh and Seon Joo Kim. Approaching the computational color constancy as a classification problem through deep learning. *Pattern Recognition*, 61:405–416, 2017. 2

[54] Yanlin Qian, Ke Chen, Jarno Nikkanen, Joni-Kristian Kamarainen, and Jiri Matas. Recurrent color constancy. In *ICCV*, 2017. 2

[55] Yanlin Qian, Said Pertuz, Jarno Nikkanen, Joni-Kristian Kämäräinen, and Jiri Matas. Revisiting gray pixel for statistical illumination estimation. *arXiv preprint arXiv:1803.08326*, 2018. 2

[56] Yanlin Qian, Joni-Kristian Kamarainen, Jarno Nikkanen, and Jiri Matas. On finding gray pixels. In *CVPR*, 2019. 2, 7

[57] Lilong Shi. Re-processed version of the gehler color constancy dataset of 568 images. *http://www. cs. sfu. ca/~ color/data/*, 2000. 6, 7, 12

[58] Wu Shi, Chen Change Loy, and Xiaoou Tang. Deep specialized network for illuminant estimation. In *ECCV*, 2016. 2

[59] Yuxiang Tang, Xuejing Kang, Chunxiao Li, Zhaowen Lin, and Anlong Ming. Transfer learning for color constancy via statistic perspective. In *AAAI*, 2022. 2

[60] Oguzhan Ulucan, Diclehan Ulucan, and Marc Ebner. Multi-scale color constancy based on salient varying local spatial statistics. *The Visual Computer*, 40(9):5979–5995, 2024. 2

[61] Joost Van De Weijer, Theo Gevers, and Arjan Gijsenij. Edge-based color constancy. *IEEE Transactions on image processing*, 16(9):2207–2214, 2007. 2, 7

[62] Sung-Min Woo, Sang-Ho Lee, Jun-Sang Yoo, and Jong-Ok Kim. Improving color constancy in an ambient light environment using the phong reflection model. *IEEE Transactions on Image Processing*, 27(4):1862–1877, 2017. 7

[63] Jin Xiao, Shuhang Gu, and Lei Zhang. Multi-domain learning for accurate and few-shot color constancy. In *CVPR*, 2020. 2

[64] Bolei Xu, Jingxin Liu, Xianxu Hou, Bozhi Liu, and Guoping Qiu. End-to-end illuminant estimation based on deep metric learning. In *CVPR*, 2020. 2

[65] Huanglin Yu, Ke Chen, Kaiqi Wang, Yanlin Qian, Zhaoxiang Zhang, and Kui Jia. Cascading convolutional color constancy. In *AAAI*, 2020. 1, 2

[66] Shuwei Yue and Minchen Wei. Effective cross-sensor color constancy using a dual-mapping strategy. *Journal of the Optical Society of America. A, Optics, image science, and vision*, 41 2:329–337, 2023. 7

[67] Shuwei Yue and Minchen Wei. Color constancy from a pure color view. *JOSA A*, 40(3):602–610, 2023. 2

[68] Shuwei Yue and Minchen Wei. Effective cross-sensor color constancy using a dual-mapping strategy. *JOSA A*, 41(2): 329–337, 2024. 2