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

Supplementary Material

A. Additional Experiments

Single CCM. While the main paper assumes the availability of two calibrated CCMs corresponding to different illuminants, some mobile cameras provide only a single CCM. We evaluate CCMNet under this constraint by testing it with a single CCM to assess its robustness in such scenarios. To validate this, we fixed the interpolation weight g in Eq. 10 to 0.5 (i.e., equal averaging of the two CCMs in existing datasets) and to 0 (i.e., using the CCM corresponding to D65 only). In both cases, a single fixed CCM was used across all color temperatures for CFE computation. As shown in Table 1, the performance with a single CCM is either slightly degraded or remains nearly unchanged compared to the dual-CCM setup. Given the histogram resolution (64×64), we believe that *projecting* the Planckian locus into the camera RGB space is more impactful than performing *precise* CCM interpolation. We expect this trend to hold even when more than two CCMs are available. However, using too many CCMs during training may increase the train-test domain gap (e.g., training with 4–5 CCMs but testing with only one), particularly when testing on an unseen mobile camera with a single CCM. Therefore, using two CCMs remains a reasonable and practical choice.

Inference Time. Previous works like CCC [2] and FFCC [3] are designed for single-camera settings and are known for their fast inference. We compare the inference speed of CCMNet with these CCC-based models. Table 2 compares the runtime of CCMNet with other CCC family models. Since the official implementation of FFCC is in MATLAB, we used the PyTorch implementation of its dynamic extension, C5, to enable a fair comparison. For reference, we report the runtime of two C5 variants: (1) a version that performs only the filtering and bias operations equivalent to FFCC/CCC (m=1, FFCC ops only), and (2) the full model including the dynamic CCC generator (m=1, full). We also measured the runtime of C5 in cross-camera settings (m=7 & m=9). CCMNet achieves significantly lower inference time than the cross-camera versions of C5, while maintaining a competitive runtime compared to FFCC/CCC–a highly efficient single-camera AWB method.

	Cube+		Gehler-Shi		NUS-8	
	Mean	Median	Mean	Median	Mean	Median
Two CCMs (original)	1.68	1.16	2.23	1.53	2.32	1.71
Single CCM ($g = 0.5$)	1.69	1.17	2.42	1.67	2.31	1.72
Single CCM ($g = 0$)	1.67	1.12	2.42	1.60	2.33	1.73

Table 1. Ablation results using a single CCM.

Feature	Model	GPU	CPU
Single camera	C5 (m=1, FFCC ops only)	0.18±0.01	0.23±0.01
	C5 (m=1, full)	1.16±0.05	4.55±0.03
Cross camera	C5 (m=7)	3.65±0.11	12.24±0.12
	C5 (m=9)	4.46±0.18	14.46±0.20
	CCMNet (ours)	1.32±0.03	10.45±0.06

Table 2. Per-image runtime (milliseconds) comparison of CCC-variant AWB models. Experiments were conducted on an NVIDIA GeForce RTX 3060 GPU and an Intel i9-12900K CPU.

B. 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. 1 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,

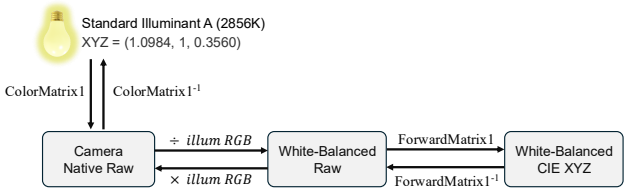

Figure 1. A schematic diagram illustrating the use of *ColorMatrix* and *ForwardMatrix*. The *ForwardMatrix* (FM) transforms white-balanced raw data into the CIE XYZ color space, while the *ColorMatrix* (CM) converts CIE XYZ values of a standard light source into the camera's native raw color space. FM1 and CM1 are calibrated for standard illuminant A (2856K), and FM2 and CM2 are calibrated for the D65 illuminant (6504K).

FM1, FM2 are used in the imaginary camera augmentation process described in Sec. D.

C. 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. 2, 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.

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 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.

D. 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 [1]. In this section, we provide a detailed explanation of the camera-to-camera mapping process used in our work. Specifically, in Sec. D.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. D.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. D.3, we describe the process of generating a device-specific illumination pool for random sampling. Finally, Sec. D.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. 3.

While our camera-to-camera mapping is inspired by the C5 augmentation approach [1], 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

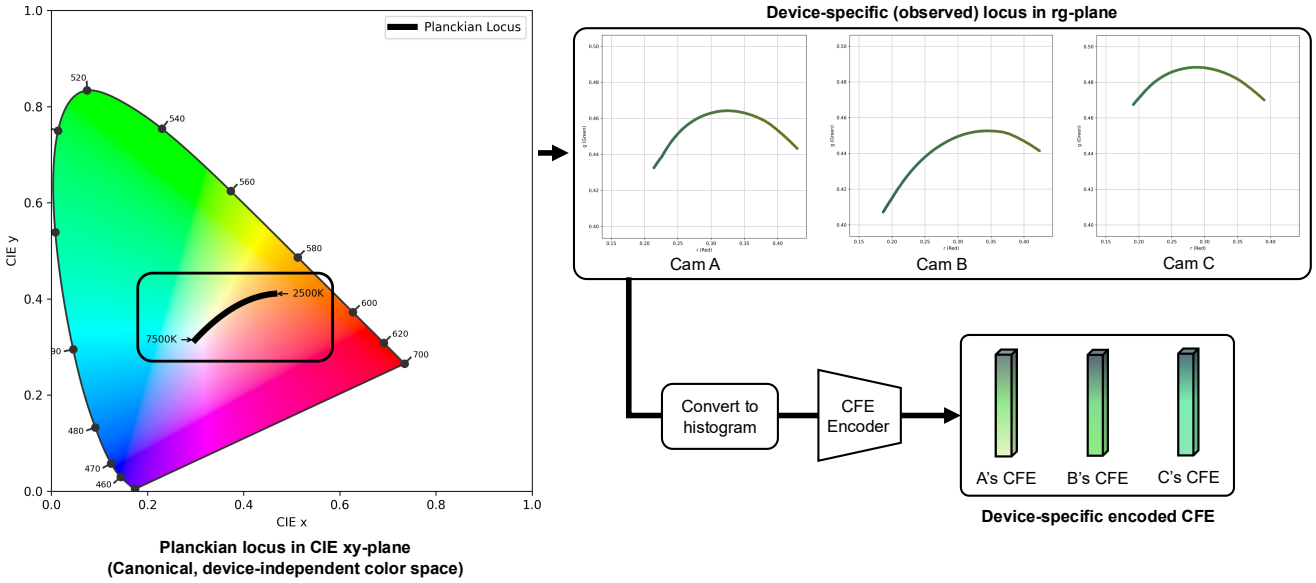

Figure 2. Detailed visualization of CFE encoding process. As mentioned in the main paper, the camera's *fingerprint* is derived by converting the reference CIE XYZ colors (locus) along the correlated color temperature (CCT) range of 2500K–7500K into the corresponding RGB locus as *observed* by each device, followed by an encoding process. Due to this characteristic, the CFE feature inherently reflects the color characteristics induced by each camera's spectral sensitivity.

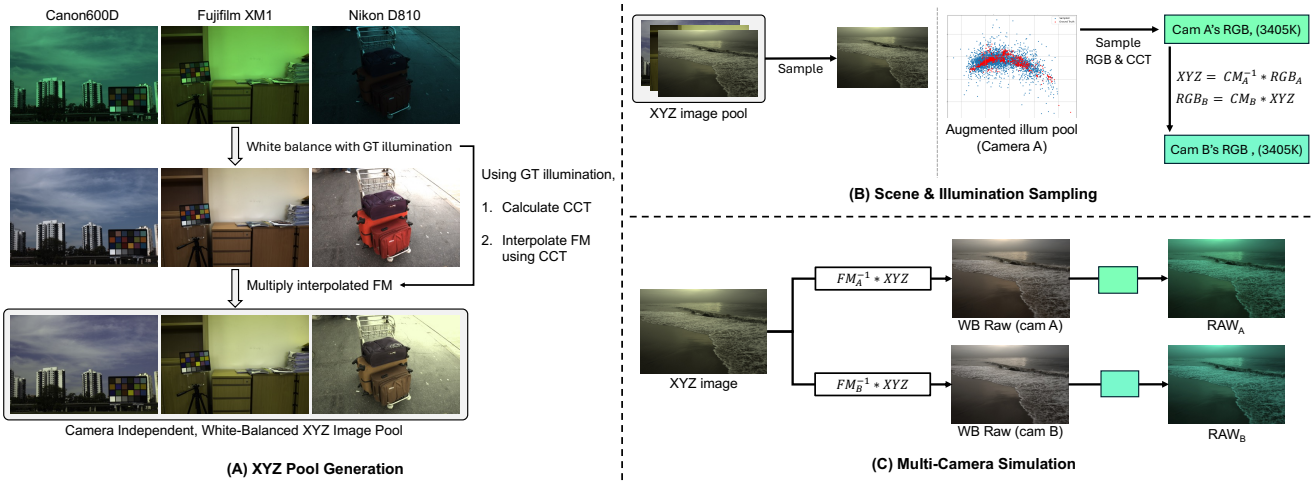

Figure 3. Overall process of camera-to-camera mapping. In (A), subsets of images taken by different cameras from multiple datasets are white-balanced using the corresponding ground-truth illuminants, and the *ForwardMatrix* is used to convert them to the CIE XYZ space, creating the XYZ image pool. In (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.

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.

D.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. B), along with the GT illumination 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. D.2 or to transform the illumination into the target camera RGB space in Sec. D.4.

D.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. 1 and Fig. 3-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. B, 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. D.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.

D.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 [1] 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 [1]. On the right side of Fig. 3-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.

D.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. D.2 and the illumination pool from Sec. D.3. See Fig. 3-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. 3-B). As illustrated in Fig. 1, 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. D.1.

Synthesizing Paired Scene from Two Cameras. Finally, as illustrated in Fig. 3-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. 1, 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.

E. Imaginary Camera Augmentation Visualizations

Here, we provide additional visualizations of the imaginary camera augmentation. As shown in Fig. 4, 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.

F. Experimental Setup

As mentioned in the main paper, the backbone f uses the standard U-Net-like architecture from C5 [1]. 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 [5] 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 [7], NUS-8 [4], and Intel-TAU [6] datasets.

G. Additional Results

We present additional visualization results in Fig. 5 and Fig. 6. As shown in Fig. 5, CCMNet achieves satisfactory accuracy across various scenes captured by a camera it has never encountered during training. In Fig. 6, we demonstrate that CCMNet maintains robust accuracy across a set of unseen cameras.

References

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

[2] Jonathan T Barron. Convolutional color constancy. In *ICCV*, 2015. 1

[3] Jonathan T Barron and Yun-Ta Tsai. Fast Fourier color constancy. In *CVPR*, 2017. 1

[4] 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. 5

[5] Diederik P Kingma. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[6] Firas Laakom, Jenni Raitoharju, Jarno Nikkanen, Alexandros Iosifidis, and Moncef Gabbouj. Intel-tau: A color constancy dataset. *IEEE access*, 9:39560–39567, 2021. 5

[7] Lilong Shi. Re-processed version of the gehler color constancy dataset of 568 images. *http://www. cs. sfu. ca/~ color/data/*, 2000. 5

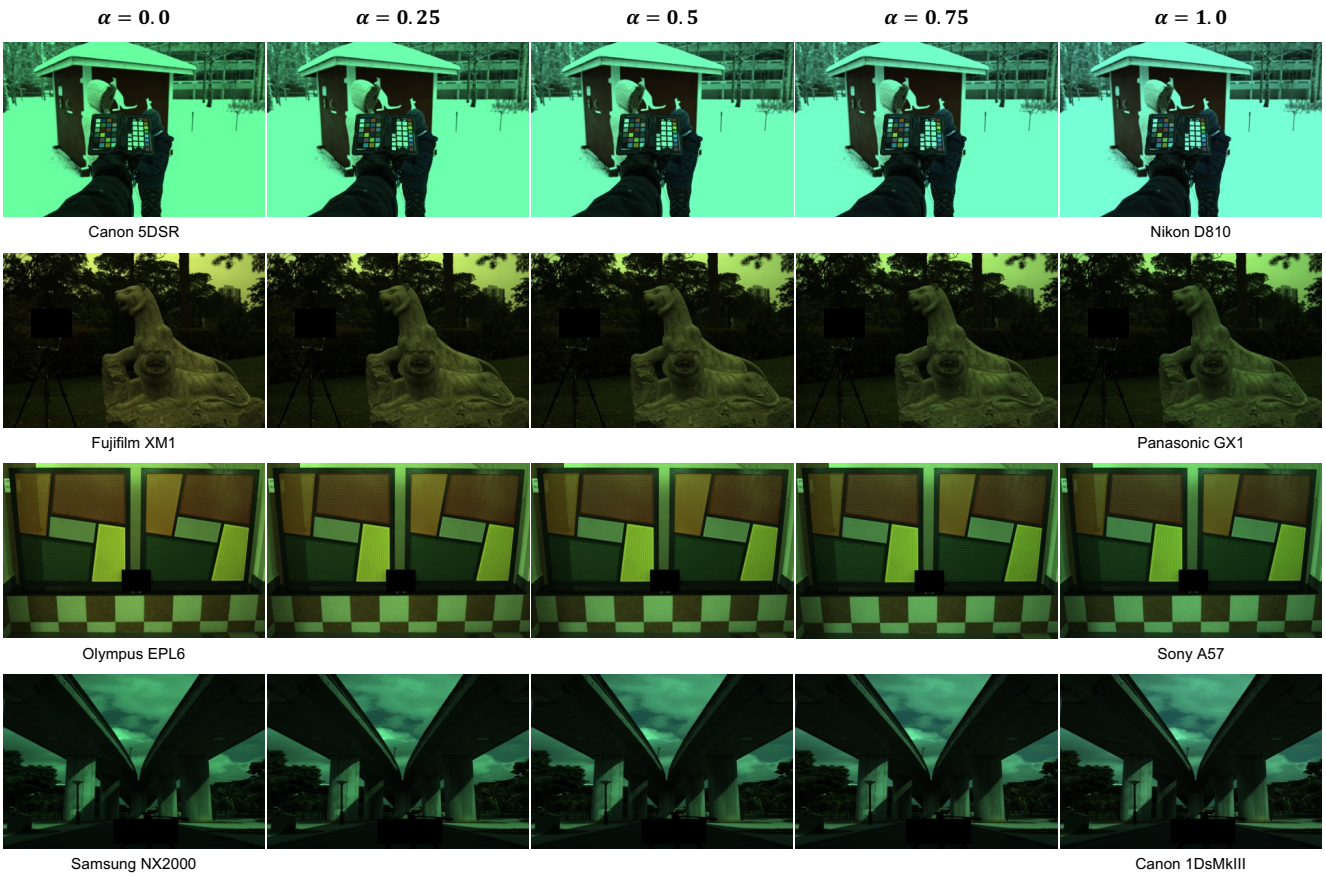

Figure 4. Results of our imaginary camera augmentation. In each row, the leftmost and rightmost images represent the source and target camera images generated using the method described in Sec. D, while the three middle images represent those produced by the *imaginary* camera, generated by interpolating 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.

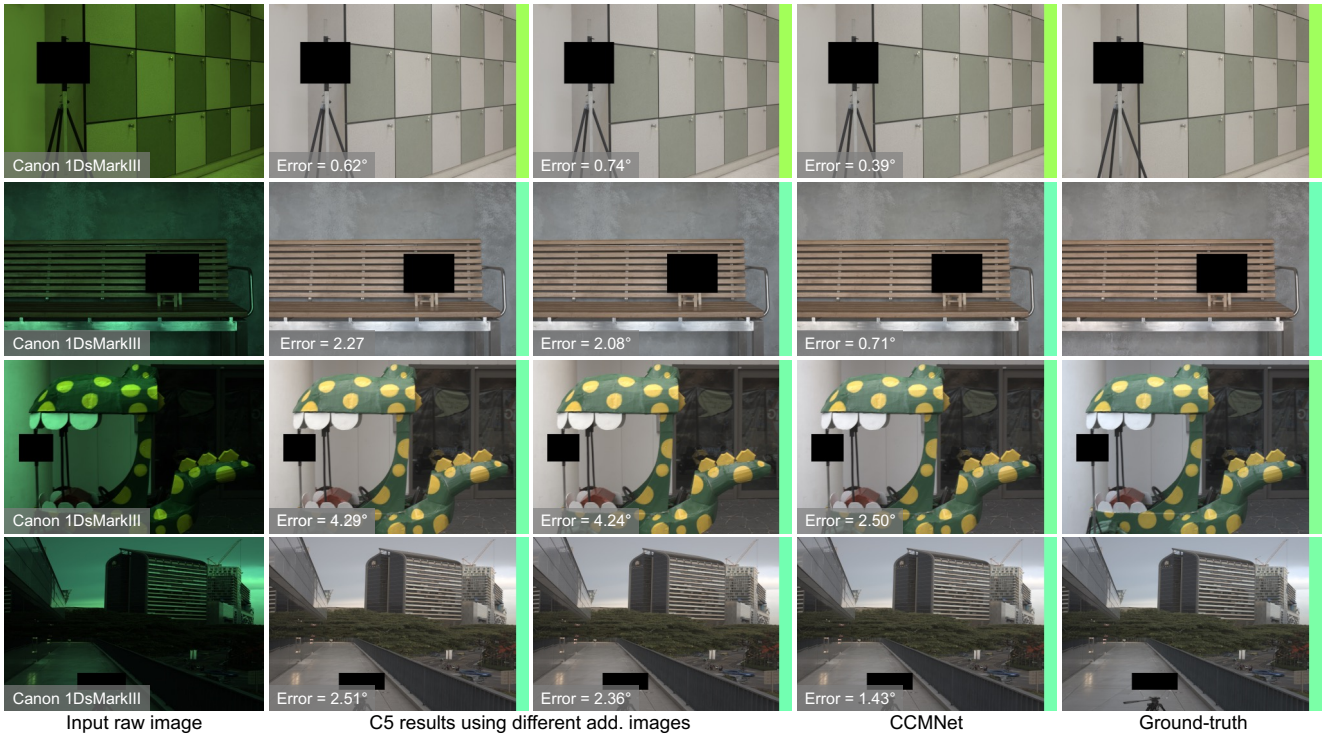

Figure 5. Additional results for Canon EOS 1Ds Mark III. CCMNet demonstrates superior performance on various scenes captured by unseen camera. Notably, CCMNet has never been exposed to any images or the CCM of the Canon 1Ds Mark III during training.

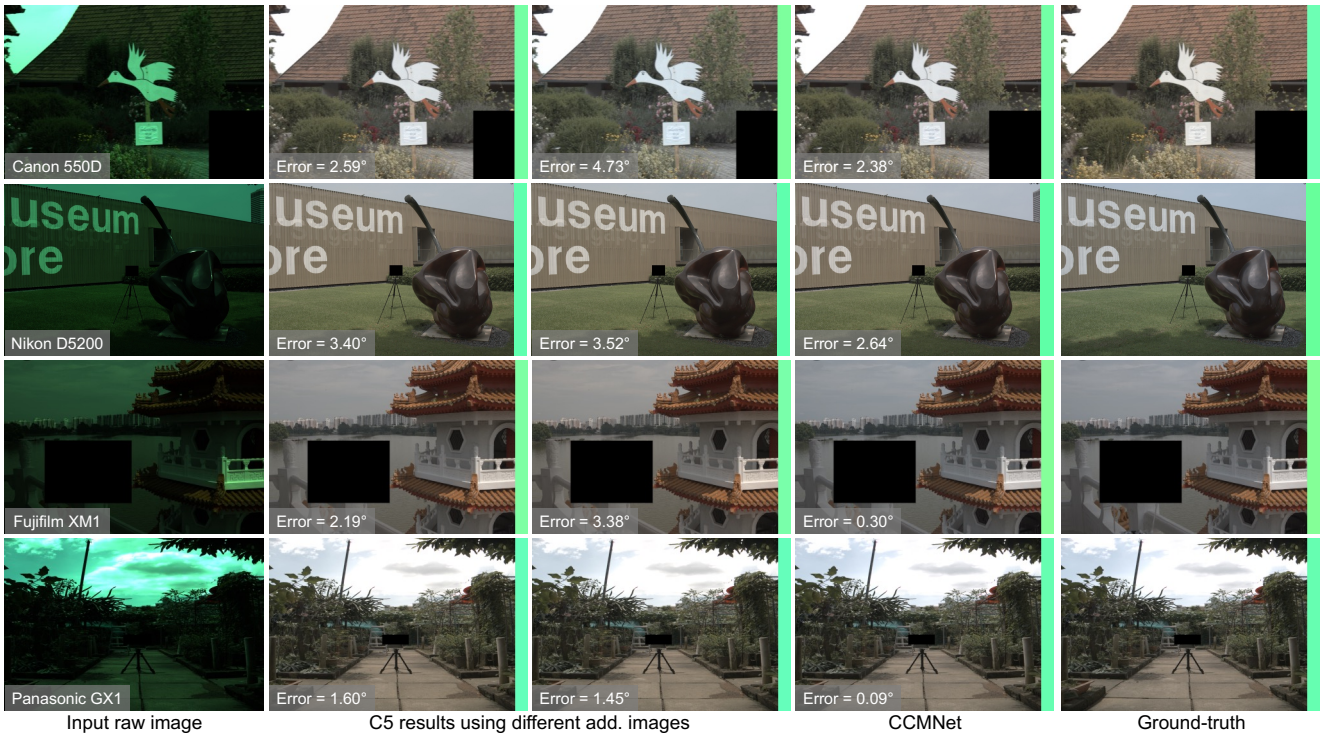

Figure 6. Additional results for various cameras show that CCMNet exhibits robust performance across a range of unseen cameras. Importantly, it has not been exposed to any images or CCMs from the cameras shown in the figure during training.