Color Temperature Tuning: Allowing Accurate Post-Capture White-Balance Editing

Mahmoud Afifi[1], Abhijith Punnappurath[1], Abdelrahman Abdelhamed[1], Hakki Can Karaimer[1,3], Abdullah Abuolaim[1], and Michael S. Brown[1,2]
[1] York University, Toronto, Canada [2] Samsung Research, AI Center, Toronto, Canada
[3] School of Computer and Communication Sciences (IC), Ecole Polytechnique Fédérale de Lausanne (EPFL)

Abstract

The in-camera imaging pipeline consists of several routines that render the sensor's scene-referred raw-RGB image to the final display-referred standard RGB (sRGB) image. One of the crucial routines applied in the pipeline is white balance (WB). WB is performed to normalize the color cast caused by the scene's illumination, which is often described using correlated color temperature. WB is applied early in the in-camera pipeline and is followed by a number of nonlinear color manipulations. Because of these nonlinear steps, it is challenging to modify an image's WB with a different color temperature in the sRGB image. As a result, if an sRGB image is processed with the wrong color temperature, the image will have a strong color cast that cannot be easily corrected. To address this problem, we propose an imaging framework that renders a small number of "tiny versions" of the original image (e.g., 0.1% of the full-size image), each with different WB color temperatures. Rendering these tiny images requires minimal overhead from the camera pipeline. These tiny images are sufficient to allow color mapping functions to be computed that can map the full-sized sRGB image to appear as if it was rendered with any of the tiny images' color temperature. Moreover, by blending the color mapping functions, we can map the output sRGB image to appear as if it was rendered through the camera pipeline with any color temperature. These mapping functions can be stored as a JPEG comment with less than 6 KB overhead. We demonstrate that this capture framework can significantly outperform existing solutions targeting post-capture WB editing.

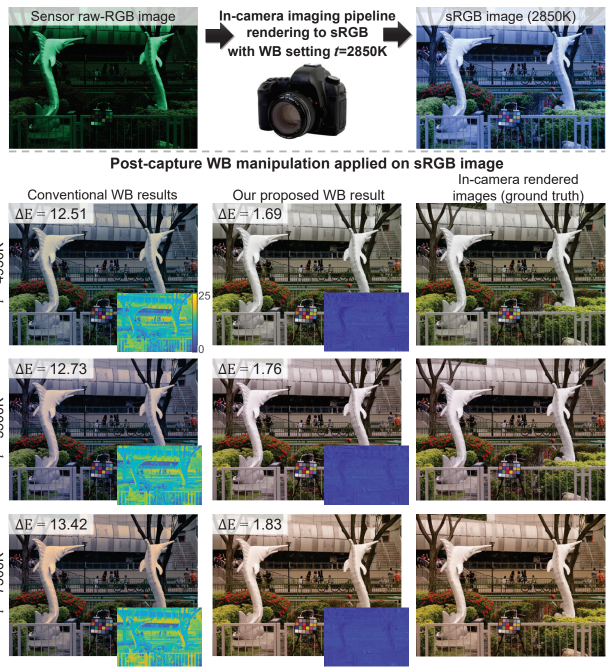

Figure 1. An sRGB image rendered using our framework with a WB setting (i.e., t =2850K). Using metadata embedded in the sRGB image, our method can modify the sRGB image to appear as if we rendered it with any *WB color temperature (e.g., t =4950K, t =5500K, t =7500K). The results (column 2) are almost identical to what the actual in-camera pipeline would have produced (column 3). Error maps (△E) insets, and average (△E) show that our method is superior to conventional WB manipulation (column 1).*

1. Introduction

A camera's scene-referred raw-RGB sensor image is converted to the final display-referred standard RGB (sRGB) output image through a series of operations collectively referred to as the in-camera imaging pipeline [25, 33]. One of the first operations in the camera pipeline is white balance (WB), which is applied as an approximation to the human visual system's ability to perceive scene content as the same color even when viewed under different illuminations [22]. WB is applied to the raw-RGB sensor image and aims to remove the color cast due to the scene's illumination, which is often described by its correlated color temperature. An image's WB temperature can either be specified by a manual setting (e.g., Tungsten, Daylight) or be estimated from the image using the camera's auto-white-balance (AWB) function.

After the WB step, the camera pipeline applies several nonlinear camera-specific operations to convert the image from the raw-RGB color space to sRGB or other color spaces, such as Adobe RGB. The pipeline's nonlinear operations make it challenging to modify the WB post-capture. This is particularly troublesome if the WB setting was incorrect, resulting in the captured image having an undesirable color cast [4]. State-of-the-art methods for post-capture WB manipulation attempt to reverse the camera pipeline and map the sRGB colors back to raw-RGB. Not only does this process necessitate careful camera calibration, but also it requires reapplying the pipeline to re-render the image into sRGB space after modifying the WB in raw-RGB space.

Contribution We propose an image capture framework that enables accurate post-capture WB manipulation *directly* in the sRGB color space without having to revert to the raw-RGB space. Our method creates a tiny downsampled version of the raw-RGB image and renders it through the camera pipeline multiple times using a set of pre-defined WB color temperature settings. Using these tiny sRGB images, color mapping functions are computed that transform the full-sized sRGB output image's colors to appear as if it they were rendered through the pipeline with the color temperature of any of the tiny images. By blending these mapping functions, the user can navigate the full parameter space of WB settings—that is, we can produce a full-sized sRGB output image with *any* color temperature that is almost identical to what the camera pipeline would have produced. Fig. 1 shows an example. The color mapping functions can be efficiently computed and stored in the sRGB-rendered image as a JPEG comment with less than 6 KB overhead.

2. Related Work

2.1 Computational Color Constancy (WB)

Unprocessed raw-RGB images contain noticeable color casts due to the captured scenes' illumination. Computational color constancy is the process performed onboard cameras to compensate for scene illumination. This is often referred to as WB in photography. The scene illumination is often described in terms of its correlated color temperature in the CIE Yxy chromaticity space [9]. WB is generally a two-step procedure: first the illumination of the scene is estimated, either by having the user select the temperature from a set of camera presets, or by using an automatic selection computed from the raw-RGB image (i.e., an AWB routine). After the illuminant has been estimated, WB correction can be performed using a simple 3×3 diagonal matrix.

Most computational color constancy research is focused on the illumination estimation (i.e., AWB) component. Early work on illumination estimation was either statistics-based [8, 19, 36] or gamut-based [17,20,21]. In recent years, learning-based methods [3, 6, 7, 14, 24, 32, 35] have also been successfully applied to this task. All of these methods estimate the illuminant and apply the correction in the camera's raw-RGB color space; they are not intended to be applied to sRGB-rendered images.

The recent work in [4] proposes to correct improperly white-balanced sRGB-rendered images by estimating polynomial mapping functions from a large set of training data. Our work is close to [4] in the sense that we also use a set of polynomial mapping functions to manipulate the WB of sRGB-rendered images. However, in contrast to [4], our work embeds the required mapping functions within the rendered sRGB images during the rendering process itself, and does not require any kind of training.

Manipulating WB in the sRGB color space is challenging. Even when the *exact* illumination color temperature is used – for example, measured from a color chart placed in the scene – and WB correction is applied in a "linearized" sRGB space [5,16], the results are still poor, as shown in Fig. 1, column 1. This is because the linearization is erroneous and fails to undo the nonlinear camera pipeline operations [4]. Our result, shown in Fig. 1, column 2, is significantly more accurate.

2.2 Raw Image Reconstruction

Radiometric calibration [15,23,28], which is closely related to raw image reconstruction, aims to undo the nonlinear operations applied onboard cameras to linearize the sRGB output. More recent raw reconstruction techniques do not just linearize the sRGB values but attempt to convert sRGB values back to their original raw-RGB values. The raw-RGB reconstruction method of Yuan and Sun [37] is based on a guided up-sampling of a low-resolution raw image, stored along with the sRGB image, to reconstruct the full-resolution raw image. Kim et al. [26] and Chakrabarti et al. [10,11] proposed the use of more complex models for the onboard-camera processing stages to achieve higher raw reconstruction accuracy. Work by Nguyen and Brown [30,31] showed that the metadata needed for accurate raw reconstruction can be computed directly from an sRGB-raw image pair, and this metadata (of only 64 KB) can be embedded within the JPEG image itself. Recently, a deep neural network architecture has been proposed [29] to emulate the camera pipeline in both directions—from raw-RGB to sRGB, and from sRGB to raw-RGB. However, to use any of these methods for WB manipulation, it is still required to revert from sRGB to raw-RGB, manipulate the WB, and reprocess the image through the camera pipeline. Our method does not require going back from sRGB to raw, and forward to sRGB again. Instead, our method allows processing the sRGB-rendered images directly in the sRGB color space.

3. Proposed Method

Our proposed method is based on the computation of a set of nonlinear color mapping functions between a set of tiny downsampled camera-rendered sRGB images. We discuss the details in the Sections 3.1–3.3. An overview of our image capture framework is shown in Fig. 2-(A). In this paper, we represent each image as a $3\times P$ matrix that contains the image's RGB triplets, where P is the total number of pixels in the image.

3.1 Rendering Tiny Images

The first step of our imaging framework is to create a tiny downsampled copy of the raw-RGB image $\mathbf{I}$. The tiny version is denoted as $\mathbf{X}$ and, in our experiments, is only 150×150 pixels, as compared to, say, a 20-megapixel full-sized image $\mathbf{I}$. Our tiny raw-RGB image $\mathbf{X}$, which is approximately 0.1% of the full-size image, can be stored in memory easily. The full-sized raw-RGB image $\mathbf{I}$ is first rendered through the camera pipeline with some WB color temperature to produce the full-sized sRGB output image $\mathbf{O}$. This color temperature is either obtained from a manually selected WB setting or estimated by the camera's AWB. The tiny image $\mathbf{X}$ is then processed by the camera pipeline N times, each time with a different WB color temperature setting $\{T_i\}_{i=1}^{N}$. These settings $\{T_i\}_{i=1}^{N}$ can correspond to the camera's preset WB options, such as Tungsten, Daylight, Fluorescent, Cloudy, and Shade, or their color temperature values, such as 2850K, 3800K, 5500K, 6500K, and 7500K, or any other chosen set of color temperatures. The resulting tiny sRGB images processed by the camera pipeline are denoted as $\{\mathbf{Y}_{T_i}\}_{i=1}^{N}$ corresponding to the WB settings $\{T_i\}_{i=1}^{N}$.

3.2 Mapping Functions

The full-sized sRGB output image $\mathbf{O}$ is downsampled to have the same dimensions as $\{\mathbf{Y}_{T_i}\}_{i=1}^{N}$ and is denoted as $\mathbf{O}_{\text{tiny}}$. For each tiny image $\{\mathbf{Y}_{T_i}\}_{i=1}^{N}$, we compute a nonlinear mapping function $\mathbf{M}_{T_i}$, which maps $\mathbf{O}_{\text{tiny}}$ to $\mathbf{Y}_{T_i}$ [4], by solving the following minimization problem:

$$\underset{\mathbf{M}_{T_i}}{\arg\min} \parallel \mathbf{M}_{T_i}\, \Phi\left(\mathbf{O}_{\text{tiny}}\right) - \mathbf{Y}_{T_i} \parallel_{\text{F}}^{2}, \tag{1}$$

where $\Phi : \mathbb{R}^3 \to \mathbb{R}^u$ is a kernel function that transforms the RGB triplets to a u-dimensional space, where $u > 3$, and $\parallel . \parallel_{\text{F}}$ denotes the Frobenius norm. For each image $\mathbf{Y}_{T_i}$, this equation finds an $\mathbf{M}_{T_i}$ that minimizes the errors between the RGB colors in the downsampled image $\mathbf{O}_{\text{tiny}}$ and its corresponding image $\mathbf{Y}_{T_i}$.

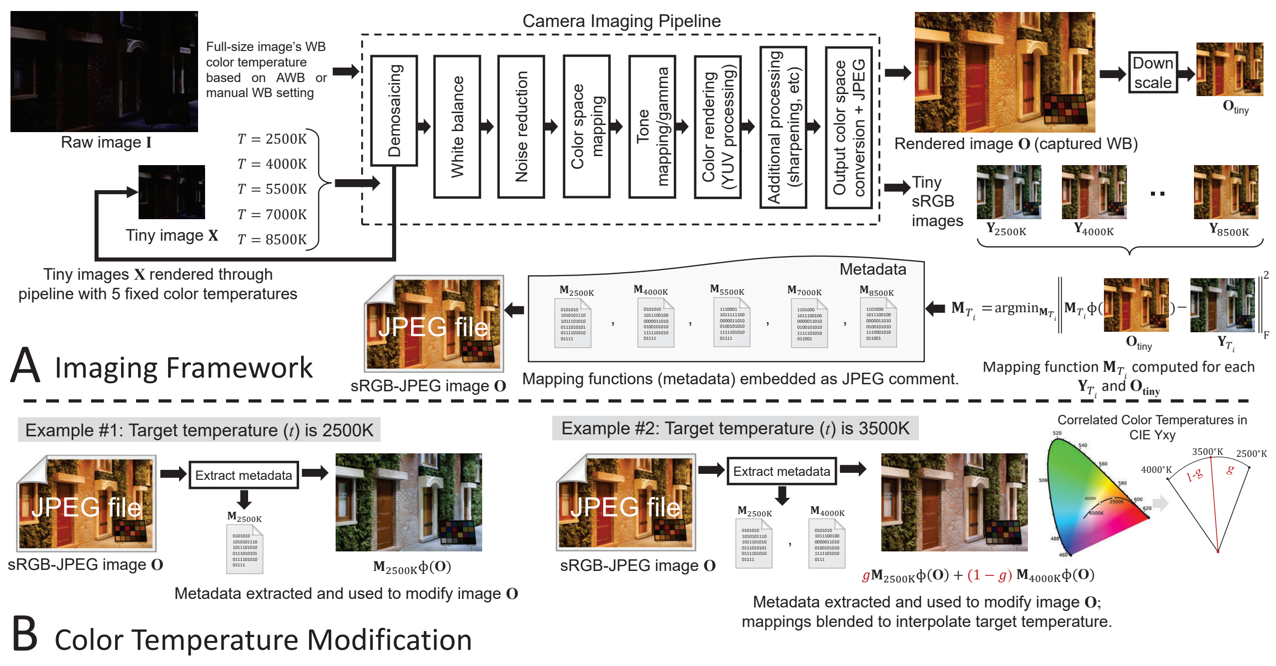
Camera Imaging Pipeline
Full-size image's WB color temperature based on AWB or manual WB setting
Raw image I
T = 2500K
T = 4000K
T = 5500K
T = 7000K
T = 8500K
Tiny image X
Tiny images X rendered through pipeline with 5 fixed color temperatures
Demosaicing
White balance
Noise reduction
Color space mapping
Tone mapping/gamma
Color rendering (YUV processing)
Additional processing (sharpening, etc)
Output color space conversion + JPEG
Rendered image O (captured WB)
Down scale
O_tiny
Tiny sRGB images
Y_2500K
Y_4000K
Y_8500K
Metadata
M_2500K
M_4000K
M_5500K
M_7000K
M_8500K
JPEG file
sRGB-JPEG image O
Mapping functions (metadata) embedded as JPEG comment.
Mapping function M_Ti computed for each Y_Ti and O_tiny
A Imaging Framework
Example #1: Target temperature (t) is 2500K
Extract metadata
M_2500K Φ(O)
Metadata extracted and used to modify image O
Example #2: Target temperature (t) is 3500K
gM_2500K Φ(O) + (1 − g) M_4000K Φ(O)
Metadata extracted and used to modify image O; mappings blended to interpolate target temperature.
Correlated Color Temperatures in CIE Yxy
3500°K
2500°K
4000°K
1-g
g
B Color Temperature Modification

Figure 2. *(A) Overview of our proposed imaging framework. The raw-RGB image $\mathbf{I}$ is downsampled to produce a tiny raw-RGB image $\mathbf{X}$. Image $\mathbf{X}$ is then rendered through the pipeline with N pre-selected WB settings T_i. This produces N tiny sRGB-rendered images $\{\mathbf{Y}_{T_i}\}_{i=1}^N$. Mapping functions $\{\mathbf{M}_{T_i}\}_{i=1}^N$ are computed between each tiny image $\{\mathbf{Y}_{T_i}\}_{i=1}^N$ and a downsampled version of the full-size sRGB output image $\mathbf{O}$ denoted as $\mathbf{O}_{tiny}$. The mappings $\{\mathbf{M}_{T_i}\}_{i=1}^N$ are stored inside the JPEG comment field of the sRGB output image $\mathbf{O}$. (B) The metadata is used to modify the sRGB image. Example 1 shows a case where the sRGB image $\mathbf{O}$ is mapped to one of the color temperatures in the metadata. The corresponding color mapping function can be extracted and used to modify the image. Example 2 shows an example where the target color temperature is not in the metadata. In this case, the target temperature mapping function is interpolated by blending between the two closest mapping functions in the metadata.*

In general, relying on kernel functions based on high-degree polynomials can hinder generalization; however, in our case, the mapping function is computed specifically for a pair of images. Hence, a kernel function with a higher degree is preferable. In our experiments, we adopted a polynomial kernel function given in [18], where $\Phi : [\text{R, G, B}]^T \rightarrow [\text{R, G, B, R}^2, \text{G}^2, \text{B}^2, \text{RG, GB, RB, R}^3, \text{G}^3, \text{B}^3, \text{RG}^2, \text{GB}^2, \text{RB}^2, \text{GR}^2, \text{BG}^2, \text{BR}^2, \text{RGB, R}^4, \text{G}^4, \text{B}^4, \text{R}^3\text{G, R}^3\text{B, G}^3\text{R, G}^3\text{B, B}^3\text{R, B}^3\text{G, R}^2\text{G}^2, \text{G}^2\text{B}^2, \text{R}^2\text{B}^2, \text{R}^2\text{GB, G}^2\text{RB, B}^2\text{RG}]^T$. Hence, each mapping function is represented by a 3×34 matrix. Once the mapping functions are computed, the set of downsampled images $\mathbf{X}$, $\{\mathbf{Y}_{T_i}\}_{i=1}^N$, and $\mathbf{O}_{\text{tiny}}$ can be discarded.

For all of our experiments in this paper, we rendered tiny images using five (5) color temperature values, 2500K, 4000K, 5500K, 7000K, and 8500K, and computed the corresponding mapping functions $\mathbf{M}_{T_i}$. The five functions require less than 6 KB of metadata to represent and can be saved inside the final JPEG image $\mathbf{O}$ as a comment field.

3.3 Color Temperature Manipulation

Once the mapping functions have been computed, we can use them to post-process the sRGB output image $\mathbf{O}$ to appear as if it was rendered through the camera pipeline with any of the WB settings $\{T_i\}_{i=1}^N$. This process can be described using the following equation:

$$\mathbf{O}_{\text{modified}} = \mathbf{M}_{T_i}\, \Phi\left(\mathbf{O}\right), \tag{2}$$

where $\mathbf{O}_{\text{modified}}$ is the full-resolution sRGB image as if it was "re-rendered" with the WB setting T_i. This is demonstrated in Example 1 of Fig. 2(B). By blending between the mapping functions, we can post-process the sRGB output image $\mathbf{O}$ to appear as if it was processed by the camera pipeline using *any* color temperature value, and not just the settings $\{T_i\}_{i=1}^N$.

Given a new target WB setting with a color temperature t, we can interpolate between the nearest pre-computed mapping functions to generate a new mapping function for t as follows:

$$\mathbf{M}_t = g\mathbf{M}_a + (1-g)\mathbf{M}_b, \qquad g = \frac{1/t - 1/b}{1/a - 1/b}, \tag{3}$$

where $a, b \in \{T_i\}_{i=1}^N$ are the nearest pre-computed color temperatures to t, such that $a < t < b$, and $\mathbf{M}_a$ and $\mathbf{M}_b$ are the corresponding mapping functions computed for temperatures a and b, respectively. The final modified image $\mathbf{O}_{\text{modified}}$ is generated by using $\mathbf{M}_t$ instead of $\mathbf{M}_{T_i}$ in Eq. 2. Example 2 of Fig. 2(B) demonstrates this process.

Recomputing the Mapping Functions Our metadata was computed for the original sRGB-rendered image $\mathbf{O}$. After $\mathbf{O}$ has been modified to $\mathbf{O}_{\text{modified}}$, this metadata has to be updated to facilitate future modifications of $\mathbf{O}_{\text{modified}}$. The update is performed so as to map $\mathbf{O}_{\text{modified}}$, with color temperature t, to our preset color temperatures $\{T_i\}_{i=1}^N$. To that end, each pre-computed mapping function $\{\mathbf{M}_{T_i}\}_{i=1}^N$ is updated based on the newly generated image $\mathbf{O}_{\text{modified}}$ as follows:

$$\underset{\mathbf{M}_{T_i}}{\arg\min} \left\| \mathbf{M}_{T_i}\Phi\left(\mathbf{M}_t\Phi\left(\mathbf{O}_{\text{tiny}}\right)\right) - \mathbf{M}_{T_i(\text{old})}\Phi\left(\mathbf{O}_{\text{tiny}}\right) \right\|_{\text{F}}^2, \tag{4}$$

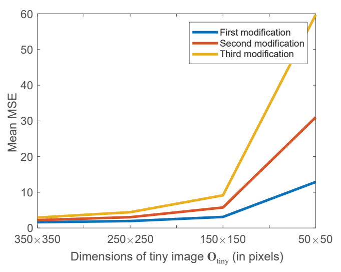

Figure 3. *The effect of downsampling size versus mean squared error (MSE) between our results and the target in-camera ground truth images after updating our metadata for three successive WB modifications with different target color temperatures.*

where $\mathbf{M}_{T_i}$ is the new mapping function and $\mathbf{M}_{T_i(\text{old})}$ is the old mapping function for the i^{th} WB before current WB modification.

4. Results

We evaluated our method on six cameras from the NUS dataset [13]. Images in this dataset were captured using DSLR cameras with a color chart placed in the scene. All images are saved in raw-RGB format and we can convert them to sRGB format using conventional in-camera pipelines [1, 25]. We rendered sRGB images with five different color temperature values: 2850K, 3800K, 5500K, 6500K, and 7500K. These correspond approximately to the common WB presets available on most cameras—namely, Tungsten, Fluorescent, Daylight, Cloudy, and Shade. For our chosen six cameras totaling 1,340 images, this process yields $1340 \times 5 = 6700$ sRGB-rendered images. We also rendered the same images using our proposed framework with the color chart masked out. During this rendering process, mapping functions corresponding to our pre-selected color temperature values (i.e., 2500K, 4000K, 5500K, 7000K, and 8500K) are computed and stored as metadata in each image.

For each image generated with a particular WB preset, we chose the other four WB settings as the target WB values. Following the procedure described in Sec. 3.3, we then processed the input image using our pre-computed mapping functions to appear as though it was rendered through the camera pipeline with the four target WB settings. For example, given an input image originally rendered with the WB Tungsten preset, we generated four WB modified versions with the following WB settings: Fluorescent, Daylight, Cloudy, and Shade. In this manner, we generated $6700 \times 4 = 26800$ WB modified images using our proposed approach. These modified images can be compared with their corresponding ground truth images rendered using the camera.

We adopted four commonly used error metrics for quantitative evaluation: (i) mean squared error (MSE), (ii) mean angular error (MAE), (iii) ΔE 2000 [34], and (iv) ΔE 76. In Table 1, the mean, lower quartile (Q1), median (Q2), and the upper quartile (Q3) of the error between the WB modified images and the corresponding ground truth images are reported. It can be observed that our method consistently outperforms the diagonal manipulation, both with and without the commonly used pre-linearization step using a 2.2 inverse gamma [16], in all metrics.

Fig. 3 shows the effect of different downsampling sizes on the post-processing WB modification compared with rendering the original raw-RGB image with each target color temperature (i.e., ground truth). In this example, we randomly selected 300 raw-RGB images from the NUS dataset. Each image was rendered using our method, and then modified to different target WB settings. This process was repeated three times to study the error propagation of our post-processing modifications.

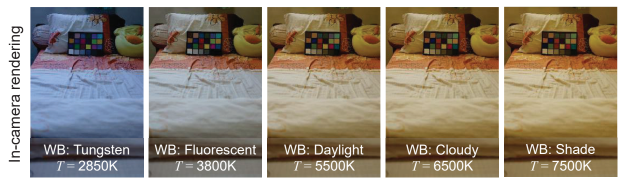

Image $\mathbf{O}$ rendered with different color temperatures (i.e., WB settings). These are produced by the camera pipeline (ground truth).

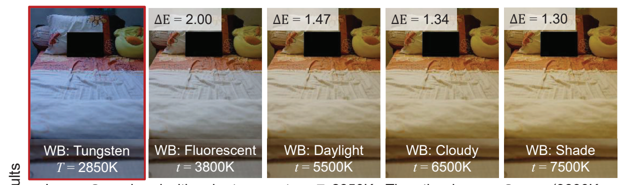

Image $\mathbf{O}$ rendered with color temperature T=2850K. The other images $\mathbf{O}_{\text{modified}}$ (3800K, 5500K, 6500K, and 7500K) are produced by applying our mapping functions $\mathbf{M}_t$ on $\mathbf{O}$.

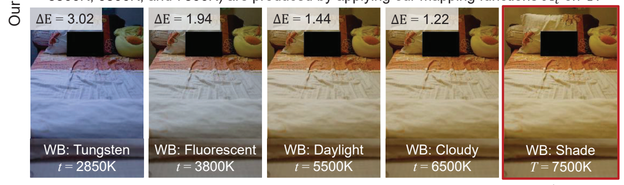

Image $\mathbf{O}$ rendered with color temperature T=7500K. The other images $\mathbf{O}_{\text{modified}}$ (2850K, 3800K, 5500K, and 6500K) are produced by applying our mapping functions $\mathbf{M}_t$ on $\mathbf{O}$.

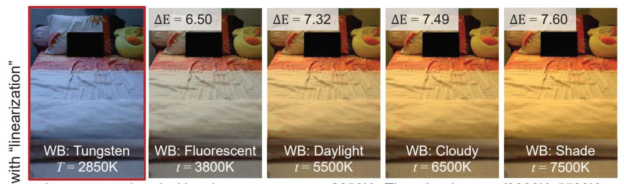

Image $\mathbf{O}$ rendered with color temperature T=2850K. The other images (3800K, 5500K, 6500K, and 7500K) are produced by diagonal manipulation with "linearization".

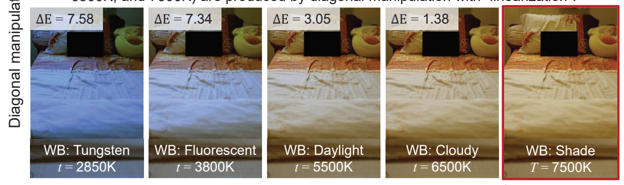

Image $\mathbf{O}$ rendered with color temperature T=7500K. The other images (2850K, 3800K, 5500K, and 6500K) are produced by diagonal manipulation with "linearization".

Figure 4. *The first row shows images rendered by the camera pipeline with different WB settings. The next two rows show how an image* $\mathbf{O}$ *(in the red box) originally captured with a particular WB can be modified post-capture using our mapping functions* $\mathbf{M}_{T_i}$ *to any other WB setting. The last two rows show image* $\mathbf{O}$ *modified to other WB settings with linearization applied.*

Representative qualitative result are shown in Figs. 4, 5, and 6. As can be seen, our method produces better results compared to the sRGB diagonal WB manipulation and Adobe Lightroom.

An interesting extension of our proposed approach lies in its ability to effectively white balance images having two illuminants in the scene [12]. We first computed an image with the WB setting corresponding to one of the illuminants. We then applied our post-capture manipulation to generate another image with the WB setting suitable for the second illuminant. Now, each of the two sRGB images was corrected for only one of the two illuminants. Relying on a user to identify the regions containing the two different illuminantions, we can spatially blend between these two images to produce an image that has both illuminants corrected. An example is shown in Fig. 7. For this experiment, we used the

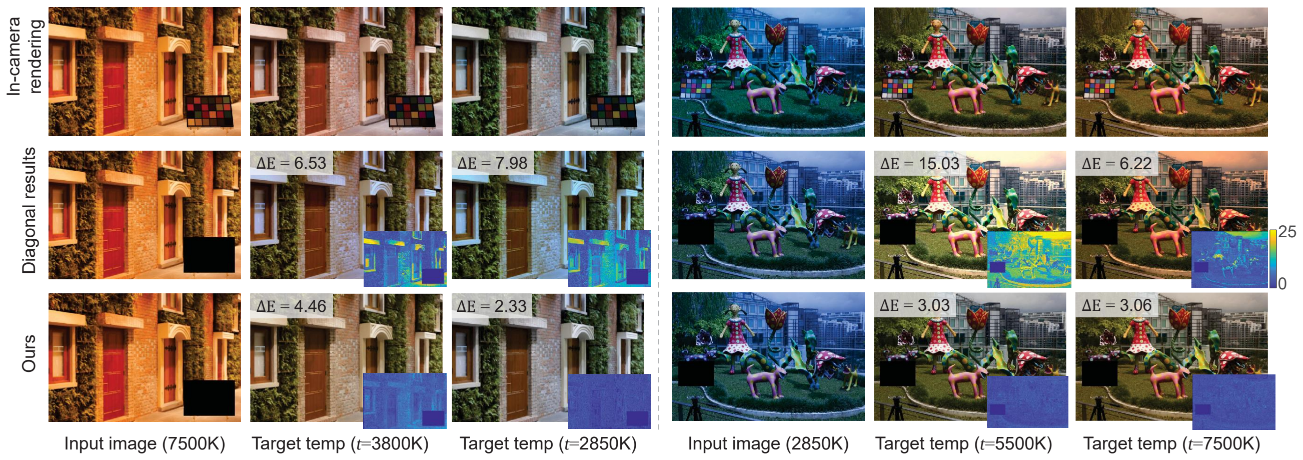

Figure 5. *First row: In-camera sRGB images rendered with different color temperatures. Second row: Results obtained by diagonal manipulation using the exact achromatic patch from the color chart. Third row: Our results. ΔE error of each result is reported and shown as an error map.*

Table 1. **Quantitative results on the NUS dataset [13] versus diagonal (Diag) WB, applied directly in sRGB (using the exact achromatic reference point), and on "linearized" sRGB [5, 16]. The terms Q1, Q2, and Q3 denote 1st, 2nd (median), and 3rd quartile, respectively. The terms MSE and MAE stand for mean square error and mean angular error, respectively. Best results are in bold.**

Method	MSE				MAE				Δ E 2000				Δ E 76			
	Mean	Q1	Q2	Q3	Mean	Q1	Q2	Q3	Mean	Q1	Q2	Q3	Mean	Q1	Q2	Q3
Diag WB	1196.93	455.21	825.38	1441.58	5.75	2.24	4.85	8.03	8.98	4.88	7.99	11.73	13.53	7.18	11.84	17.98
Diag WB w linearization	1160.16	428.18	771.78	1400.49	5.49	2.19	4.56	7.61	8.61	4.62	7.58	11.09	12.87	6.82	11.22	16.99
Ours	**75.58**	**35.51**	**61.05**	**98.68**	**2.04**	**1.42**	**1.86**	**2.45**	**3.09**	**2.33**	**3.00**	**3.74**	**4.39**	**3.21**	**4.26**	**5.30**

lazy snapping method [27] with superpixels [2] to quickly mark the regions containing the two illuminants. The blending mask was then smoothed by applying a Gaussian blur.

5. Concluding Remarks

We have described an imaging framework that enables users to accurately modify the WB color temperature of an sRGB-rendered image. Such functionality is currently not possible with the conventional imaging pipeline. Our method allows easy recovery from white-balance errors, or color temperature manipulation to change the image's aesthetics. Our approach requires a minor modification to the existing imaging pipeline and produces metadata that can be stored in an image file (e.g., JPEG) with only 6 KB of overhead. Finally, we note that our imaging framework can be modified to accommodate other types of camera settings (e.g., picture styles) or color space options, such as allowing post-capture display white-point selection.

Acknowledgments This study was funded in part by the Canada First Research Excellence Fund for the Vision: Science to Applications (VISTA) programme; an NSERC Discovery Grant; and an AdeptMind Scholarship. Dr. Brown contributed to this article in his personal capacity as a professor at York University. The views expressed are his own and do not necessarily represent the views of Samsung Research.

References

[1] A. Abdelhamed, S. Lin, and M. S. Brown. A high-quality denoising dataset for smartphone cameras. In *CVPR*, 2018.

[2] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE TPAMI*, 34(11):2274–2282, 2012.

[3] M. Afifi and M. S. Brown. Sensor-independent illumination estimation for DNN models. In *BMVC*, 2019.

[4] M. Afifi, B. Price, S. Cohen, and M. S. Brown. When color constancy goes wrong: Correcting improperly white-balanced images. In *CVPR*, 2019.

[5] M. Anderson, R. Motta, S. Chandrasekar, and M. Stokes. Proposal for a standard default color space for the internet - sRGB. In *Color Imaging Conference*, 1996.

[6] J. T. Barron. Convolutional color constancy. In *ICCV*, 2015.

[7] J. T. Barron and Y.-T. Tsai. Fast Fourier color constancy. In *CVPR*, 2017.

[8] G. Buchsbaum. A spatial processor model for object colour perception. *Journal of the Franklin institute*, 310(1):1–26, 1980.

[9] H. Can Karaimer and M. S. Brown. Improving color reproduction accuracy on cameras. In *CVPR*, 2018.

[10] A. Chakrabarti, D. Scharstein, and T. E. Zickler. An empirical camera model for internet color vision. In *BMVC*, 2009.

[11] A. Chakrabarti, Y. Xiong, B. Sun, T. Darrell, D. Scharstein, T. Zickler, and K. Saenko. Modeling radiometric uncertainty for vision with tone-mapped color images. *IEEE TPAMI*, 36(11):2185–2198, 2014.

[12] D. Cheng, A. Abdelhamed, B. Price, S. Cohen, and M. S. Brown. Two illuminant estimation and user correction preference. In *CVPR*, 2016.

[13] D. Cheng, D. K. Prasad, and M. 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.

[14] D. Cheng, B. Price, S. Cohen, and M. S. Brown. Effective learning-based illuminant estimation using simple features. In *CVPR*, 2015.

[15] P. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *SIGGRAPH*, 1997.

[16] M. Ebner. *Color Constancy*. Wiley Publishing, 1st edition, 2007.

[17] G. D. Finlayson. Color in perspective. *IEEE TPAMI*, 18(10):1034–

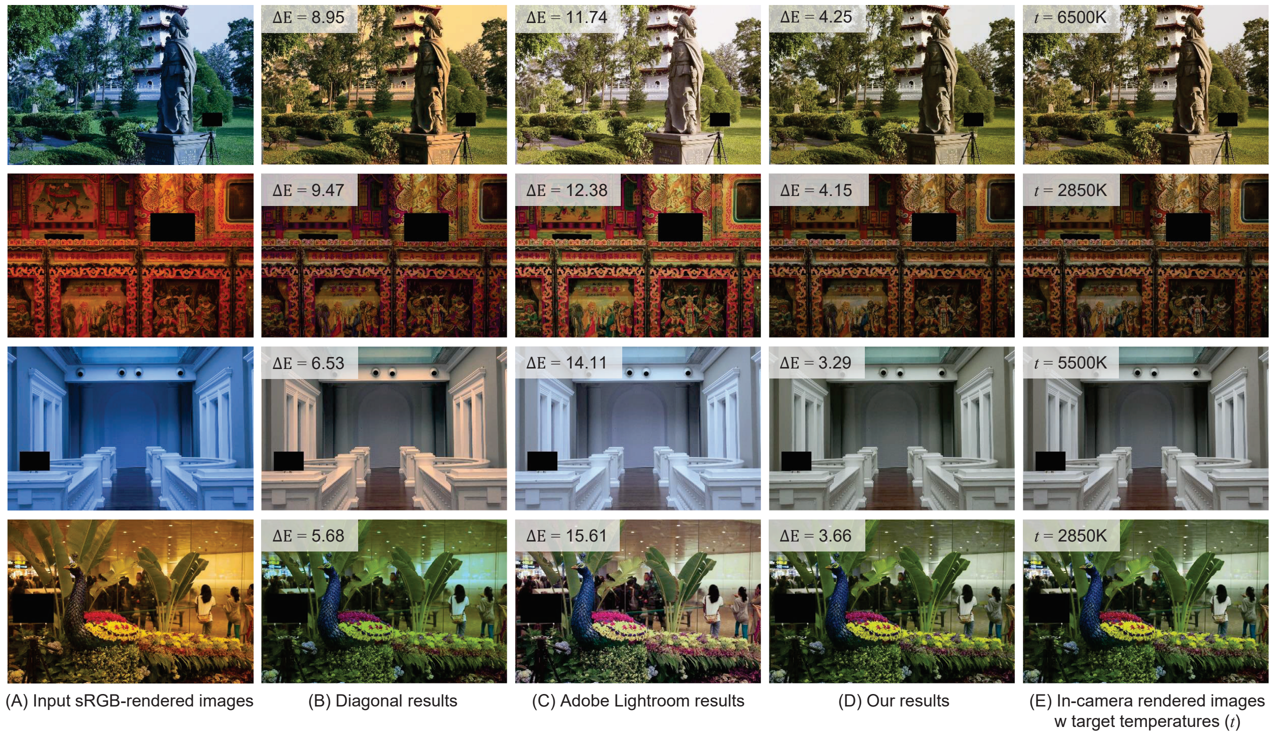

Figure 6. *(A) Input sRGB-rendered images. (B) Diagonal manipulation results. (C) Adobe Lightroom results. (D) Our results. (E) In-camera sRGB images rendered with the target color temperature t. The average ΔE error is shown for each image.*

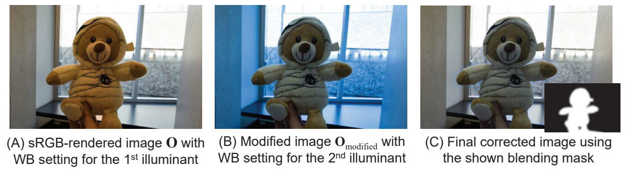
(A) sRGB-rendered image O with WB setting for the 1st illuminant
(B) Modified image O_modified with WB setting for the 2nd illuminant
(C) Final corrected image using the shown blending mask

Figure 7. *Our method can be used to correct or modify the WB settings for two-illuminant scenes. (A) Camera-rendered sRGB image with the WB settings used for the first illuminant. (B) Generated sRGB image for the second illuminant using our metadata. (C) Our final result after blending.*

1038, 1996.

[18] G. D. Finlayson, M. Mackiewicz, and A. Hurlbert. Color correction using root-polynomial regression. *IEEE TIP*, 24(5):1460–1470, 2015.

[19] G. D. Finlayson and E. Trezzi. Shades of gray and colour constancy. In *Color and Imaging Conference*, 2004.

[20] D. A. Forsyth. A novel algorithm for color constancy. *IJCV*, 5(1):5–36, 1990.

[21] A. Gijsenij, T. Gevers, and J. Weijer. Generalized gamut mapping using image derivative structures for color constancy. *IJCV*, 86(2-3):127–139, 2010.

[22] A. Gilchrist. *Seeing Black and White*. Number 40. OUP USA, 2006.

[23] M. D. Grossberg and S. K. Nayar. Determining the camera response from images: What is knowable? *IEEE TPAMI*, 25(11):1455–1467, 2003.

[24] Y. Hu, B. Wang, and S. Lin. FC4: Fully convolutional color constancy with confidence-weighted pooling. In *CVPR*, 2017.

[25] H. C. Karaimer and M. S. Brown. A software platform for manipulating the camera imaging pipeline. In *ECCV*, 2016.

[26] S. J. Kim, H. T. Lin, Z. Lu, S. Süsstrunk, S. Lin, and M. S. Brown. A new in-camera imaging model for color computer vision and its application. *IEEE TPAMI*, 34(12):2289–2302, 2012.

[27] Y. Li, J. Sun, C.-K. Tang, and H.-Y. Shum. Lazy snapping. *ACM Transactions on Graphics (ToG)*, 23(3):303–308, 2004.

[28] T. Mitsunaga and S. K. Nayar. Radiometric self calibration. In *CVPR*, 1999.

[29] S. Nam and S. J. Kim. Modelling the scene dependent imaging in cameras with a deep neural network. In *ICCV*, pages 1726–1734, 2017.

[30] R. M. Nguyen and M. S. Brown. RAW image reconstruction using a self-contained sRGB-JPEG image with only 64 KB overhead. In *CVPR*, 2016.

[31] R. M. Nguyen and M. S. Brown. RAW image reconstruction using a self-contained sRGB–JPEG image with small memory overhead. *IJCV*, 126(6):637–650, 2018.

[32] S. W. Oh and S. J. Kim. Approaching the computational color constancy as a classification problem through deep learning. *Pattern Recognition*, 61:405–416, 2017.

[33] R. Ramanath, W. E. Snyder, Y. Yoo, and M. S. Drew. Color image processing pipeline. *Signal Processing Magazine*, 22(1):34–43, 2005.

[34] G. Sharma, W. Wu, and E. N. Dalal. The CIEDE2000 color-difference formula: Implementation notes, supplementary test data, and mathematical observations. *Color Research & Application*, 30(1):21–30, 2005.

[35] W. Shi, C. C. Loy, and X. Tang. Deep specialized network for illuminant estimation. In *ECCV*, 2016.

[36] J. Van De Weijer, T. Gevers, and A. Gijsenij. Edge-based color constancy. *IEEE TIP*, 16(9):2207–2214, 2007.

[37] L. Yuan and J. Sun. High quality image reconstruction from RAW and JPEG image pair. In *ICCV*, 2011.