When Color Constancy Goes Wrong: Correcting Improperly White-Balanced Images

Mahmoud Afifi
York University
mafifi@eecs.yorku.ca

Brian Price
Adobe Research
bprice@adobe.com

Scott Cohen
Adobe Research
scohen@adobe.com

Michael S. Brown
York University
mbrown@eecs.yorku.ca

Abstract

This paper focuses on correcting a camera image that has been improperly white-balanced. This situation occurs when a camera's auto white balance fails or when the wrong manual white-balance setting is used. Even after decades of computational color constancy research, there are no effective solutions to this problem. The challenge lies not in identifying what the correct white balance should have been, but in the fact that the in-camera white-balance procedure is followed by several camera-specific nonlinear color manipulations that make it challenging to correct the image's colors in post-processing. This paper introduces the first method to explicitly address this problem. Our method is enabled by a dataset of over 65,000 pairs of incorrectly white-balanced images and their corresponding correctly white-balanced images. Using this dataset, we introduce a k-nearest neighbor strategy that is able to compute a nonlinear color mapping function to correct the image's colors. We show our method is highly effective and generalizes well to camera models not in the training set.

1. Introduction

When taking a photograph, we expect our images to be correctly white-balanced. Computer vision algorithms implicitly assume a correct white balance (WB) by expecting their input image colors to be correct. What happens when the WB is not correct? In such cases, the images have the familiar bluish/reddish color casts that not only are undesirable from a photography standpoint but also can adversely affect the performance of vision algorithms.

Correcting improperly white-balanced images is poorly understood. Sources such as Matlab [1] purports misleading solutions that suggest the problem is a matter of identifying what the correct WB should have been and then applying this as a post-correction. However, this solution is not effective and does not consider WB within the full context of the in-camera processing pipeline.

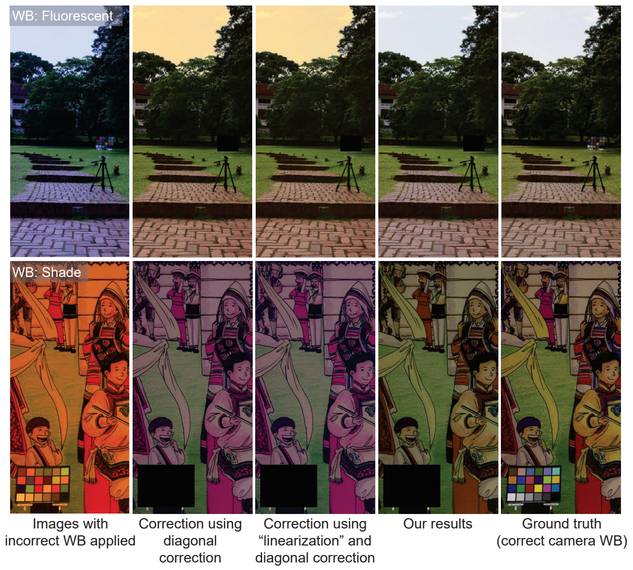

WB: Fluorescent
WB: Shade
Images with incorrect WB applied
Correction using diagonal correction
Correction using "linearization" and diagonal correction
Our results
Ground truth (correct camera WB)

Figure 1. Two incorrectly white-balanced images produced by different cameras and attempts to correct them using (1) a linear WB correction, (2) correction by first applying a gamma linearization ([3,17]), and (3) our results. Also shown is the ground truth image produced by the camera using the correct WB.

WB is applied on board cameras to remove the color cast in images caused by the scene's illumination. WB falls into the research area of computational color constancy that aims to mimic the human visual system's ability to perceive scene colors similarly even when observed under different illuminations [21]. WB correction is performed with a simple linear diagonal 3×3 matrix. The WB transform is applied to the camera's raw-RGB image and is often one of the first steps in the in-camera processing pipeline. After WB is applied, several additional transforms are applied to convert the image values from a sensor-specific raw-RGB color space to an output-referred color space–namely standard RGB (sRGB). These additional transforms include camera-specific nonlinear color manipulations that make up a camera's photo-finishing routines (for more details, see [26]). A simple model for the in-camera color manipulation from sensor raw-RGB to sRGB can be expressed as:

$$\mathbf{I}_{\text{sRGB}} = f_{\text{XYZ}\rightarrow\text{sRGB}}(\mathbf{T}_{\text{raw}\rightarrow\text{XYZ}}\mathbf{D}_{\text{WB}}\mathbf{I}_{\text{raw}}), \qquad (1)$$

where $\mathbf{I}_{\text{sRGB}}$ and $\mathbf{I}_{\text{raw}}$ are $3\times N$ matrices containing the image values in the sRGB and raw-RGB spaces respectively, N is the total number of pixels, $\mathbf{D}$ represents the 3×3 diagonal WB matrix, $\mathbf{T}$ is a 3×3 linear transform that maps from white-balanced raw-RGB to a device-independent color space, such as CIE-XYZ (or one of its derivatives), and $f(\cdot)$ is a nonlinear function that compounds various operations, including color enhancement, tone-manipulation, and a final sRGB gamma encoding. We can think of $f(\cdot)$ as the collective in-camera color rendering operation performed on the camera after WB. Prior works [11,27,29] have shown that $f(\cdot)$ not only is specific to camera models but also depends on the camera settings used during image capture.

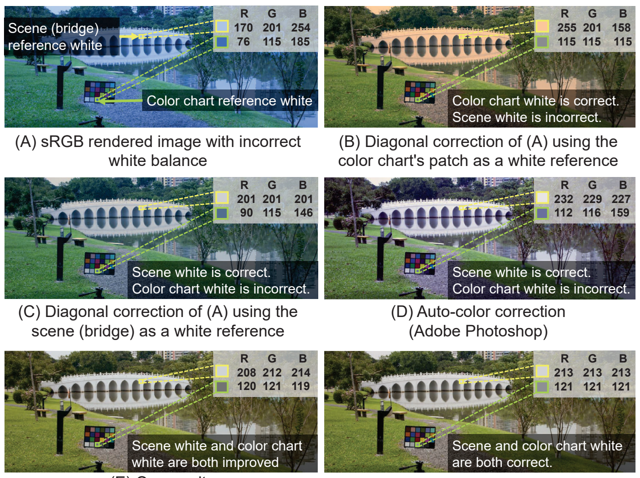
From Eq. 1, it is clear that because WB is applied early in the processing chain, attempting to correct it using a diagonal matrix will not work. Matlab suggests using an optional pre-linearization step using a 2.2 gamma [3, 17]. However, it has long been known that a 2.2 gamma does not reflect the true nature of the $f(\cdot)$ function [15]. Fig. 1 shows examples.

Contribution We propose a data-driven approach to correct images that have been improperly white-balanced. As part of this effort, we have generated a new dataset of over 65,000 images from different cameras that have been rendered to sRGB images using each camera's pre-defined WB settings and picture styles. Each incorrect white-balanced image in the dataset has a corresponding correct white-balanced sRGB image rendered to a standard picture style. Given an improperly white-balanced camera image, we outline a k-nearest neighbor strategy that is able to find similar incorrectly white-balanced images in the dataset. Based on these similar example images, we describe how to construct a nonlinear color correction transform that is used to remove the color cast. Our approach gives good results and generalizes well to camera makes and models not found in the training data. In addition, our solution requires a small memory overhead (less than 24 MB) and is computationally fast.

2. Related Work

As far as we are aware, this paper is the first to directly address the problem of correcting colors in an incorrectly white-balanced image. The problem is related to three areas in computer vision: (i) computational color constancy, (ii) radiometric calibration, and (iii) general color manipulation. These are discussed within the context of our problem.

Computational Color Constancy WB is performed to mimic our visual system's ability to perceive objects as having the same color even when viewed under different illuminations. WB requires the camera's sensor response to the scene's illumination. Once the color of the illumination is known, a 3×3 diagonal matrix is used to normalize the illumination's colors by mapping them to the achromatic line in the camera's raw-RGB color space (i.e., the raw-RGB values corresponding to the the scene's illumination are mapped to lie on the R=G=B "white" line). The vast majority of computational color constancy research is focused on illumination estimation. Representative examples include [5, 6, 8, 10, 12, 20–22, 25, 35]. Illumination estimation is computed in the camera's raw-RGB color space and makes up the camera's auto WB mechanism.


Scene (bridge) reference white
R G B
170 201 254
76 115 185
Color chart reference white
R G B
255 201 158
115 115 115
Color chart white is correct. Scene white is incorrect.
R G B
201 201 201
90 115 146
Scene white is correct. Color chart white is incorrect.
R G B
232 229 227
112 116 159
Scene white is correct. Color chart white is incorrect.
R G B
208 212 214
120 121 119
Scene white and color chart white are both improved
R G B
213 213 213
121 121 121
Scene and color chart white are both correct.

(A) sRGB rendered image with incorrect white balance
(B) Diagonal correction of (A) using the color chart's patch as a white reference
(C) Diagonal correction of (A) using the scene (bridge) as a white reference
(D) Auto-color correction (Adobe Photoshop)
(E) Our result
(F) sRGB rendered image with correct white balance (ground truth image)

Figure 2. (A) A camera sRGB image with a wrong white balance applied. (B) and (C) show traditional white balance correction applied to the image using different reference white points manually selected from the image (note: this would represent the best solution for an automatic illumination estimation algorithm). (D) shows the result of auto-color correction from Adobe Photoshop. (E) Result from our approach. (F) The ground truth camera image with the correct white balance applied.

Note that *none* of the aforementioned illumination estimation methods are intended to be applied to sRGB images. Even if an illumination estimation method could be used to determine the color of an achromatic region in a sRGB image, this information is still insufficient to correct an incorrectly white-balanced image. To stress this point, we provide an example in Fig. 2. Fig. 2-(A) shows a camera image rendered through a camera pipeline to an sRGB output with the incorrect WB. Two achromatic regions in the scene are highlighted: (i) a patch from a white bridge and (ii) a neutral patch from the color chart. The same scene is rendered with the correct WB in Fig. 2-(F). Because the WB was applied correctly in Fig. 2-(F), both the scene achromatic regions lie on the white line (i.e., R=G=B). Fig. 2-(B) and (C) show attempts at using standard diagonal WB correction using

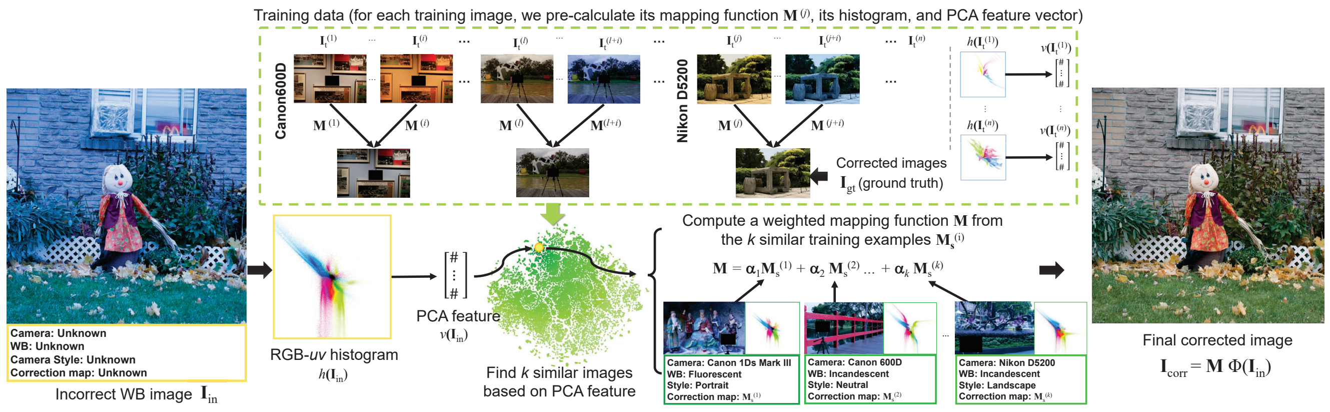


Figure 3. An overview diagram of our overall procedure. For the input sRGB image and our training data, we first extract the histogram feature of the input image, followed by generating a compact PCA feature to find the most similar k nearest neighbors to the input image in terms of colors. Based on the retrieved similar images, a color transform $\mathbf{M}$ is computed to correct the input image.

the color chart's patch and bridge scene region as reference white, respectively. We can see that in both cases, only the selected reference white region is corrected, while the other region remains incorrect.

Radiometric Calibration Radiometric calibration is the process of parameterizing a camera's nonlinear color manipulation in order to reverse it. Radiometric calibration essentially attempts to approximate f^{-1} from Eq. 1. There have been a number of methods addressing radiometric calibration (e.g., [11,23,27–31,38]). Radiometric calibration is performed to linearize the photo-finished camera sRGB images in order to aid low-level computer vision tasks that require a linear response to scene irradiance (e.g., photometric stereo, image deblurring, HDR imaging) [33,34]. When radiometric calibration data is available, it can be used to undo the photo-finishing in an sRGB image in order to correct WB, as demonstrated by [27]. However, performing radiometric calibration requires a tedious calibration procedure. The radiometric metadata needs to be stored and associated with each captured image [34]. As a result, radiometric calibration data is rarely available for most users.

Color Transformations Another topic related to our work is color transformations that are used to map an input image's color space to some desired target color space. These methods are typically used for the purpose of colorimetric calibration. Representative examples include [2, 7, 16, 18, 19, 24]. Most of these methods rely on explicit color correspondences between the input and target color spaces. The most effective methods are those based on polynomial color correction [19, 24] and recent root polynomial color correction [19]. These approaches use kernel functions that project the original color data from the three-channel RGB color space to a higher-dimensional space.

Additional ad hoc methods include routines in software such as Adobe Photoshop that perform auto-color and auto-tone manipulation [14]. Fig. 2-(D) shows correction based on Adobe Photoshop's auto-color. Similar to Fig. 2-(B) and (C), this can correct one of the regions, but not the other.

To date, there are no effective methods for correcting incorrectly white-balanced camera images and often images like Fig. 2-(A) are simply discarded.

3. Proposed Method

We begin with an overview of our approach followed by specific implementation details. Our method is designed with the additional constraints of fast execution and a small memory overhead to make it suitable for incorporation as a mobile app or as a software plugin. Alternative designs and additional evaluation of parameters are provided in the supplemental material.

3.1. Method Overview

Fig. 3 provides an overviews our framework. Given an incorrectly white-balanced sRGB image, denoted as $\mathbf{I}_{\text{in}}$, our goal is to compute a mapping $\mathbf{M}$ that can transform the input image's colors to appear as if the WB was correctly applied.

Our method relies on a large set of n training images expressed as $\mathbf{I}_{\text{t}} = \{\mathbf{I}_{\text{t}}^{(1)}, \ldots, \mathbf{I}_{\text{t}}^{(n)}\}$ that have been generated using the *incorrect* WB settings. Each training image has a corresponding *correct* white-balanced image (or ground truth image), denoted as $\mathbf{I}_{\text{gt}}^{(i)}$. Note that multiple training images may share the same target ground truth image. Section 3.2 details how we generated this dataset.

For each pair of training image $\mathbf{I}_{\text{t}}^{(i)}$ and its ground truth image $\mathbf{I}_{\text{gt}}^{(i)}$, we compute a nonlinear color correction matrix $\mathbf{M}^{(i)}$ that maps the incorrect image's colors to its target

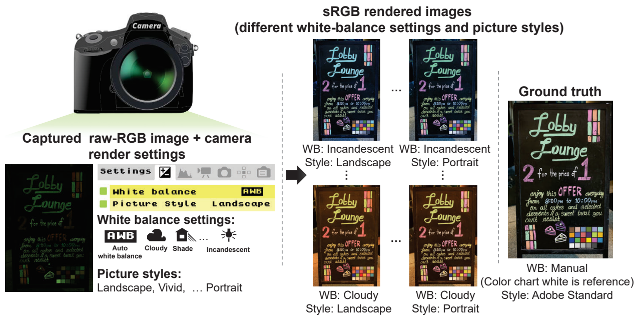

sRGB rendered images
(different white-balance settings and picture styles)
Camera
Captured raw-RGB image + camera render settings
Settings
White balance AWB
Picture Style Landscape
White balance settings:
AWB
Auto white balance
Cloudy
Shade
Incandescent
Picture styles:
Landscape, Vivid, ... Portrait
WB: Incandescent Style: Landscape
WB: Incandescent Style: Portrait
WB: Cloudy Style: Landscape
WB: Cloudy Style: Portrait
Ground truth
WB: Manual
(Color chart white is reference)
Style: Adobe Standard

Figure 4. Example of rendering sRGB training images. Working directly from the raw-RGB camera image, we render sRGB output images using the camera's pre-defined white balance settings and different picture styles. A target white balance sRGB image is also rendered using the color rendition chart in the scene to provide the ground truth.

ground truth image's colors. The details of this mapping are discussed in Section 3.3.

Given an input image, we search the training set to find images with similar color distributions. This image search is performed using compact features derived from input and training image histograms as described in Section 3.4. Finally, we obtain a color correction matrix $\mathbf{M}$ for our input image by blending the associated color correction matrices of the similar training image color distributions, denoted as $\mathbf{M}_{\text{s}}$. This is described in Section 3.5.

3.2. Dataset Generation

Our training images are generated from two publicly available illumination estimation datasets: the NUS dataset [12] and the Gehler dataset [20]. Images in these datasets were captured using digital single-lens reflex (DSLR) cameras with a color rendition chart placed in the scene that provides ground truth reference for illumination estimation. Since these datasets are intended for use in illumination estimation, they were captured in raw-RGB format. Because the images are in the camera's raw-RGB format, we can convert them to sRGB output emulating different WB settings and picture styles on the camera. To do this, we use the Adobe Camera Raw feature in Photoshop to render different sRGB images using different WB presets in the camera. In addition, each incorrect WB can be rendered with different camera picture styles (e.g., Vivid, Standard, Neutral, Landscape). Depending on the make and model of the camera, a single raw-RGB image can be rendered to more than 25 different camera-specific sRGB images. These images make up our training images $\{\mathbf{I}_{\text{t}}^{(1)}$, ..., $\mathbf{I}_{\text{t}}^{(n)}\}$.

To produce the correct target image, we manually select the "ground truth" white from the middle gray patches in the color rendition chart, followed by applying a camera-independent rendering style—namely, *Adobe Standard*. This provides the target ground truth sRGB image $\mathbf{I}_{\text{gt}}$. Fig. 4 illustrates an example of a raw-RGB image from the NUS dataset and the corresponding sRGB images rendered with different WB settings and picture styles. In the end, we generated 62,535 images from these datasets (there is an additional set generated for cross-dataset validation using the same approach; more details are given in Sec. 4.1).

3.3. Color Correction Transform

After generating our training images, we have n pairs of images representing an incorrect WB image $\mathbf{I}_{\text{t}}^{(i)}$ and its *correct* WB image $\mathbf{I}_{\text{gt}}^{(i)}$. These are represented as $3\times N$ matrices, where N is the total number of pixels in the image and the three rows represent the red, green, and blue values in the camera's output sRGB color space.

We can compute a color correction matrix $\mathbf{M}^{(i)}$, which maps $\mathbf{I}_{\text{t}}^{(i)}$ to $\mathbf{I}_{\text{gt}}^{(i)}$, by minimizing the following equation:

$$\underset{\mathbf{M}^{(i)}}{\arg\min} \left\| \mathbf{M}^{(i)} \; \Phi\left(\mathbf{I}_{\text{t}}^{(i)}\right) - \mathbf{I}_{\text{gt}}^{(i)} \right\|_{\text{F}}, \tag{2}$$

where $\|.\|_{\text{F}}$ is the Frobenius norm and Φ is a kernel function that projects the sRGB triplet to a high-dimensional space.

We have examined several different color transformation mappings and found the polynomial kernel function proposed by Hong et al. [24] provided the best results for our task (additional details are given in the supplemental materials). Based on [24], Φ:[R, G, B]^T $\rightarrow$ [R, G, B, RG, RB, GB, R^2, G^2, B^2, RGB, 1]^T and $\mathbf{M}^{(i)}$ is represented as a 3×11 matrix. Note that spatial information is not considered when estimating the $\mathbf{M}^{(i)}$.

3.4. Image Search

Since our color correction matrix is related to the image's color distribution, our criteria for finding similar images are based on the color distribution. We also seek compact representation as these features represent the bulk of information that will need to be stored in memory.

Inspired by prior work [5, 6], we construct a histogram feature from the log-chrominance space, which represents the color distribution of an image $\mathbf{I}$ as an $m\times m\times 3$ tensor that is parameterized by uv. We refer to this as an RGB-uv histogram. This histogram is generated by the function $h(\mathbf{I})$ described by the following equations:

$$\begin{aligned}
\mathbf{I}_{y(i)} &= \sqrt{\mathbf{I}_{\text{R}(i)}^2 + \mathbf{I}_{\text{G}(i)}^2 + \mathbf{I}_{\text{B}(i)}^2}, \\
\mathbf{I}_{u1(i)} &= \log\left(\mathbf{I}_{\text{R}(i)}\right) - \log\left(\mathbf{I}_{\text{G}(i)}\right), \\
\mathbf{I}_{v1(i)} &= \log\left(\mathbf{I}_{\text{R}(i)}\right) - \log\left(\mathbf{I}_{\text{B}(i)}\right), \\
\mathbf{I}_{u2} &= -\mathbf{I}_{u1}\,, \mathbf{I}_{v2} = -\mathbf{I}_{u1} + \mathbf{I}_{v1}, \\
\mathbf{I}_{u3} &= -\mathbf{I}_{v1}\,, \mathbf{I}_{v3} = -\mathbf{I}_{v1} + \mathbf{I}_{u1}.
\end{aligned} \tag{3}$$

$$\mathbf{H}\left(\mathbf{I}\right)_{(u,v,C)} = \textstyle\sum_i \mathbf{I}_{y(i)} \left[\left|\mathbf{I}_{uC(i)} - u\right| \leqslant \frac{\varepsilon}{2} \wedge \left|\mathbf{I}_{vC(i)} - v\right| \leqslant \frac{\varepsilon}{2}\right], \tag{4}$$

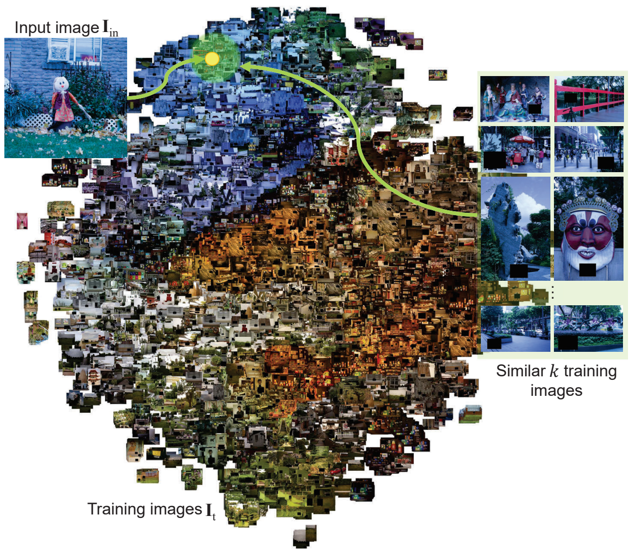


Figure 5. Visualization of the training images based on their corresponding PCA feature vectors. In this figure t-SNE [32] is used to aid visualization of the training space. Shown is an example input image and several of the nearest images retrieved using the PCA feature.

$$h(\mathbf{I})_{(u,v,C)} = \sqrt{\frac{\mathbf{H}\left(\mathbf{I}\right)_{(u,v,C)}}{\sum_{u'}\sum_{v'}\mathbf{H}\left(\mathbf{I}\right)_{(u',v',C)}}}, \tag{5}$$

where $i = \{1, ..., N\}$, $\text{R}, \text{G}, \text{B}$ represent the color channels in $\mathbf{I}$, $C \in \{1, 2, 3\}$ represents each color channel in the histogram, and ε is the histogram bin's width. Taking the square root after normalizing $\mathbf{H}$ increases the discriminatory ability of our projected histogram feature [4, 5].

For the sake of efficiency, we apply a dimensionality reduction step in order to extract a compact feature representing each RGB-uv histogram. We found that the linear transformation is adequate for our task to map the vectorized histogram $\text{vec}(h\left(\mathbf{I}\right)) \in \mathbb{R}^{m\times m\times 3}$ to a new lower-dimensional space. The principal component analysis (PCA) feature vector is computed as follows:

$$v(\mathbf{I}) = \mathbf{W}^{T}\left(\text{vec}\left(h\left(\mathbf{I}\right)\right) - \mathbf{b}\right), \tag{6}$$

where $v\left(\mathbf{I}\right) \in \mathbb{R}^{c}$ is the PCA feature vector containing c principal component (PC) coefficients, $c \ll m \times m \times 3$, $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_c], \mathbf{w} \in \mathbb{R}^{m\times m\times 3}$ is the PC coefficient matrix computed by the singular value decomposition, and $\mathbf{b} \in \mathbb{R}^{m\times m\times 3}$ is the mean histogram vector. As a result, each training image $\mathbf{I}_{\text{t}}^{(i)}$ can be represented by a small number of PC coefficients $v\left(\mathbf{I}_{\text{t}}^{(i)}\right)$. The input image is finally represented by $v(\mathbf{I}_{\text{in}})$. The L_2 distance is used to measure the similarity between the PCA feature vectors. Fig. 5 visualizes the training images based on their corresponding PCA features.

3.5. Final Color Correction

Given a new input image, we compute its PCA feature and search the training dataset for images with similar features. We extract the set of color correction matrices $\mathbf{M}_{\text{s}}$ associated with the k similar PCA features. The final correction matrix $\mathbf{M}$ is then computed as a weighted linear combination of the correction matrices $\mathbf{M}_{\text{s}}$ as follows:

$$\mathbf{M} = \sum_{j=1}^{k} \boldsymbol{\alpha}_j \mathbf{M}_{\text{s}}^{(j)}, \tag{7}$$

where $\boldsymbol{\alpha}$ is a weighting vector represented as a radial basis function:

$$\boldsymbol{\alpha}_j = \frac{\exp\left(-\mathbf{d}_j^2/2\sigma^2\right)}{\sum_{k'=1}^{k}\exp\left(-\mathbf{d}_{k'}^2/2\sigma^2\right)}, \; j \in [1, ..., k], \tag{8}$$

where σ is the radial fall-off factor and $\mathbf{d}$ represents a vector containing the L_2 distance between the given input feature and the similar k training features.

As shown in Fig. 3, the final color transformation is generated based on correction transformations associated with training images taken from different cameras and render styles (see supplemental materials for a study of the effect of having different picture styles on the results). By blending the mapping functions from images produced by a wide range of different cameras and their different photo-finishing styles, we can interpret $\mathbf{M}$ correction as mapping the input image to a *meta-camera*'s output composed from the most similar images to the input.

Lastly, the corrected image $\mathbf{I}_{\text{corr}}$ is produced by the following equation:

$$\mathbf{I}_{\text{corr}} = \mathbf{M}\;\Phi\left(\mathbf{I}_{\text{in}}\right). \tag{9}$$

Since our training data includes examples of WB settings that are close to the manual ground truth, our method implicitly can also deal with test images that were rendered with the correct WB settings (see supplemental materials for examples).

3.6. Implementation Details

Our Matlab implementation requires approximately 0.54 seconds to compute the histogram feature. Once the PCA histogram feature is computed, the correction process takes an average of 0.73 seconds; this process includes the PCA feature extraction, the brute-force search of the k nearest neighbors, blending the correction matrix, and the final image correction. All the reported runtimes were computed on an Intel® Xeon® E5-1607 @ 3.10 GHz machine and for a 12 mega-pixel image. The accelerated GPU implementation runs on average in 0.12 seconds to correct a 12 mega-pixel image using GTX 1080 GPU.

Our method requires 23.3 MB to store 62,535 feature vectors, mapping matrices, the PCA coefficient matrix, and the mean histogram vector using single-precision floating-point representation without affecting the accuracy.

In our implementation, each PCA feature vector was represented by 55 PC scores (i.e., $c = 55$), the PC coefficient matrix $\mathbf{W}$ was represented as a $(60 \times 60 \times 3) \times 55$ matrix (i.e., $m = 60$), and the mean vector $\mathbf{b} \in \mathbf{R}^{60\times60\times3}$. We used a fall-off factor $\sigma = 0.25$ and $k = 25$. Comparisons using different parameter values are given in the supplemental materials.

4. Experimental Results

Our method is compared with common approaches that are currently used to correct an improperly white-balanced sRGB image. We first describe the data used to test our method (Sec. 4.1). Afterwards, we show both quantitative (Sec. 4.2) and qualitative results (Sec. 4.3).

4.1. Dataset

As described in Sec. 3.2, we have generated a dataset of 62,535 images that contains pairs of incorrect and correct (ground truth) images. An additional set has been generated using the same procedure containing 2,881 images.

Intrinsic set (Set 1): consists of 62,535 images generated from the raw-RGB images provided in the NUS dataset [12], its extension [13], and the Gehler dataset [20].

Extrinsic set (Set 2): is generated for cross-dataset validation, where it consists of 2,881 sRGB images rendered from four mobile phones (iPhone 7, Google Pixel, LG G4, and Galaxy S6 Edge) and one of the DSLR cameras (Olympus) from the NUS dataset that was excluded from Set 1. Set 2 does not contain any cameras from Set 1. There are 1,874 DSLR images and 1,007 mobile phone images (468 of them rendered from raw-RGB images provided by Karaimer and Brown [9]).

We use Set 1 for training and evaluation, using three-fold validation, such that the three folds are disjointed in regards to the imaged scenes, meaning if the scene (i.e., original raw-RGB image) appears in a fold, it is excluded from the other folds. The color rendition chart is masked out in the image and ignored during training and testing. For evaluation on Set 2, we use the entire training from Set 1 of $\sim$62K.

4.2. Quantitative Results

We compared our results against a diagonal WB correction that is computed using the center gray patch in the color checker chart placed in the scene. We refer to this as the *exact achromatic* reference point, as it represents a true neutral point found in the scene. This exact white point represents the best results that an illumination estimation algorithm could achieve when applied to our input in order to determine the diagonal WB matrix. This is equivalent to the example in Fig. 2-(B).

For the sake of completeness, we also compared our results against a "linearized" diagonal correction that applies an inverse gamma operation [3, 17], then performed WB using the exact reference point, and then reapplied the gamma to produce the result in the sRGB color space. We also include results using Adobe Photoshop corrections—specifically, the auto-color function (AC) and auto-tone function (AT).

We adopted three commonly used error metrics for the evaluation, which are: (i) mean squared error (MSE), (ii) mean angular error (MAE), and (iii) $\triangle$E, which is widely used to measure changes in visual perception between two colors. There are different versions of $\triangle$E; we report the results of $\triangle$E 2000 [37] (results of using $\triangle$E 76 [36] are also reported in the supplemental materials). In Table 1, the mean, lower quartile (Q1), median (Q2), and the upper quartile (Q3) of the error between the corrected images and the corresponding ground truth images are reported.

Table 1 shows that our proposed method consistently outperforms the other approaches in all metrics.

4.3. Qualitative Results and User Study

As mentioned in Sec. 2, the inverse of the nonlinear photo-finishing in an sRGB image can be performed by applying a full radiometric calibration [27] or from radiometric metadata that has been embedded in the sRGB rendered image to restore the original raw-RGB image [34]. Fig. 6 shows a comparison between our result and applying diagonal WB correction to the reconstructed raw-RGB image proposed in [34]. We are able to achieve comparable results without the need for radiometric calibration.

Qualitative visual results for Set 1 and Set 2 are shown in Fig. 7. It is arguable that our results are the most visually similar to the ground truth images. To confirm this independently, we have conducted a user study of 35 participants (18 males and 17 females), ranging in age from 21 to 46. Each one was asked to choose the most visually similar image to the ground truth image between the results of our method and the diagonal correction with the exact reference point. Experiments were carried out in a controlled environment. The monitor was calibrated using a Spyder5 colorimeter. Participants were asked to compare 24 pairs of images, such that for each quartile, based on the MSE of each method, 4 images were randomly picked from Set 1 and Set 2. That means the selected images represent the best, median, and worst results of each method and for each set. On average, 93.69% of our results were chosen as the most similar to the ground truth images. Fig. 8 illustrates that the results of this study are statistically significant with p-value < 0.01.

We note that our algorithm does fail on certain types of

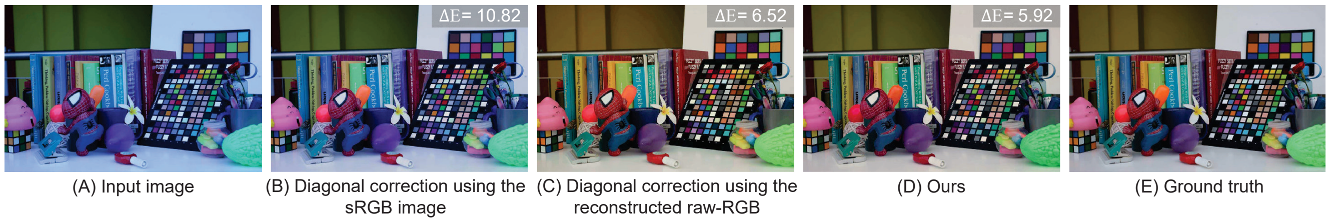

ΔE= 10.82
ΔE= 6.52
ΔE= 5.92
(A) Input image
(B) Diagonal correction using the sRGB image
(C) Diagonal correction using the reconstructed raw-RGB
(D) Ours
(E) Ground truth

Figure 6. Comparison of our results against applying white balance to the reconstructed raw-RGB image as proposed by [34]. The work in [34] is able to reconstruct the raw-RGB image by embedding radiometric calibration metadata in the sRGB data. By reversing back to the raw-RGB, they can re-apply white balance correctly and then re-render the sRGB image. (A) Input sRGB image. (B) Result of diagonal correction on sRGB color space. (C) Result of diagonal correction on the reconstructed raw-RGB image [34]. (D) Our result. (E) Ground truth image. (A)-(C) and (E) are adapted from [34].

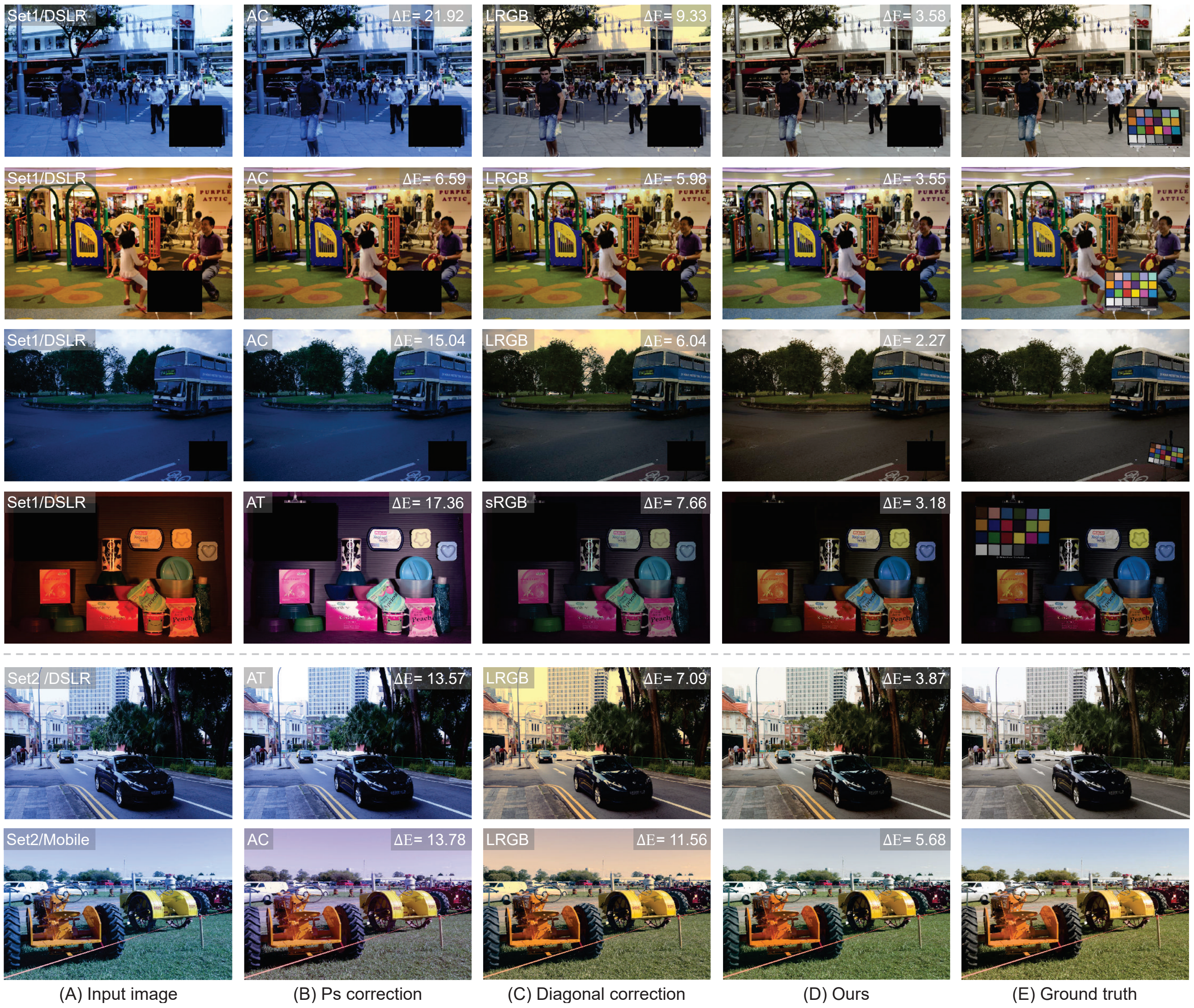


Figure 7. Comparisons between the proposed approach and other techniques on **Set 1** (first four rows) and **Set 2** (last two rows). (A) Input image in sRGB. (B) Results of Adobe Photoshop (Ps) color correction functions. (C) Results of diagonal correction using the *exact* reference point obtained directly from the color chart. (D) Our results. (E) Ground truth images. In (B) and (C), we pick the best result between the auto-color (AC) and auto-tone (AT) functions and between the sRGB (sRGB) and "linearized" sRGB (LRGB) [3, 17] based on △E values, respectively.

Table 1. Comparisons between our method with existing solutions for white balancing sRGB images. We compare our results against the diagonal white balance correction using an *exact achromatic* reference point obtained from the color chart in the image. The diagonal correction is applied directly to the sRGB images, denoted as (sRGB) and on the "linearized" sRGB [3, 17], denoted as (LRGB). Also, we compare our results against the Adobe Photoshop functions: auto-color (AC) and auto-tone (AT). The terms Q1, Q2, and Q3 denote the first, second (median), and third quartile, respectively. The terms MSE and MAE stand for mean square error and mean angular error, respectively. The top results are indicated with yellow and bold.

Method	**MSE**				**MAE**				**△E**			
	Mean	**Q1**	**Q2**	**Q3**	**Mean**	**Q1**	**Q2**	**Q3**	**Mean**	**Q1**	**Q2**	**Q3**
Intrinsic set (Set 1): DSLR multiple cameras (62,535 images)												
Photoshop-AC	780.52	157.39	430.96	991.28	7.96°	3.43°	5.59°	10.58°	10.06	5.75	8.92	13.30
Photoshop-AT	1002.93	238.33	606.74	1245.51	7.56°	3.08°	5.75°	10.83°	11.12	6.55	10.54	14.68
Diagonal WB (sRGB)	135.77	20.20	71.74	196.15	4.63°	1.99°	3.56°	6.14°	4.69	2.25	4.00	6.68
Diagonal WB (LRGB)	130.01	19.73	68.54	183.65	4.29°	1.85°	3.35°	5.70°	4.59	2.24	3.89	6.51
Ours	**77.79**	**13.74**	**39.62**	**94.01**	**3.06°**	**1.74°**	**2.54°**	**3.76°**	**3.58**	**2.07**	**3.09**	**4.55**
Extrinsic set (Set 2): DSLR and mobile phone cameras (2,881 images)												
Photoshop-AC	745.49	240.58	514.33	968.27	10.19°	5.25°	8.60°	14.13°	11.71	7.56	11.41	15.00
Photoshop-AT	953.85	386.7	743.84	1256.94	11.91°	7.01°	10.70°	15.92°	13.12	9.63	13.18	16.5
Diagonal WB (sRGB)	422.31	110.70	257.76	526.16	7.99°	4.36°	7.11°	10.57°	8.53	5.52	8.38	11.11
Diagonal WB (LRGB)	385.23	99.05	230.86	475.72	7.22°	3.80°	6.34°	9.54°	8.15	5.07	7.88	10.68
Ours	**171.09**	**37.04**	**87.04**	**190.88**	**4.48°**	**2.26°**	**3.64°**	**5.95°**	**5.60**	**3.43**	**4.90**	**7.06**

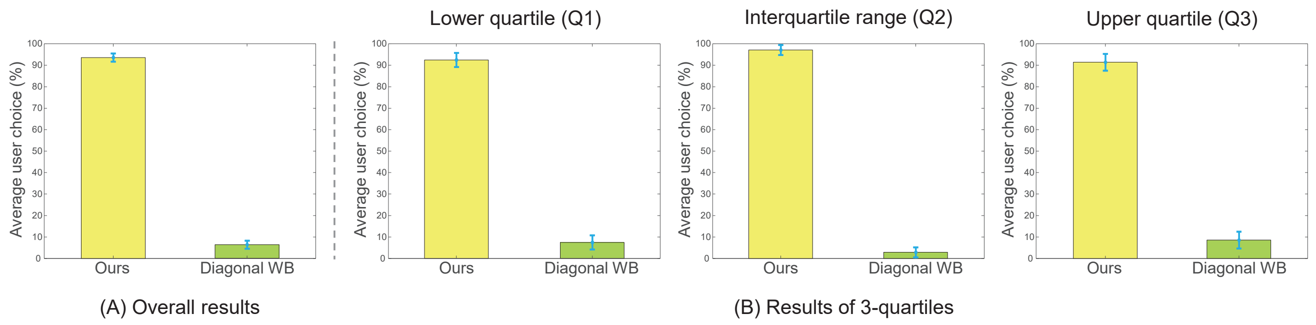


Figure 8. The results of a user study with 35 people in which users are asked which output is most visually similar to the ground truth image. An equal number of images are selected randomly from the different quartiles. The outcome of the user study is shown via interval plots, with error bars shown at a 99% confidence interval. (A) shows the overall user preference of all results, while (B) shows the results categorized by Q1, Q2, and Q3.

inputs. These are generally images with strong color casts that have a large number of saturated colors (see Fig. 9).

5. Concluding Remarks

White balance is a critical low-level computer vision step that is often taken for granted. Most computer vision datasets, especially those composed of images crawled from the web, are implicitly biased towards correctly white-balanced images as improperly white-balanced images are rarely uploaded in the first place. As discussed in this paper, correcting an sRGB image that has been rendered by a camera with the wrong WB setting is challenging.

This paper has proposed a data-driven method to correct improperly white-balanced images. Extensive quantitative and qualitative experiments, including a user study, demonstrate the effectiveness of this approach. More importantly, our approach is practical, requiring less than 24 MB of data and less than 1.5 seconds to correct a full-resolution image. In addition, our approach generalizes well to images not contained within our training set. We believe this dataset, the only one of its kind, will serve as a useful resource for future work on this and related topics.

Acknowledgment 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 Adobe Gift Award.

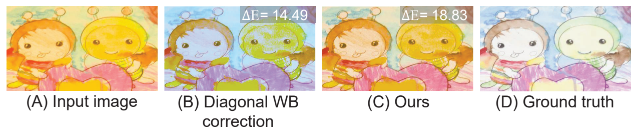


Figure 9. Failure example of the proposed method. (A) Input sRGB image. (B) Results using a diagonal white balance correction based on the *exact achromatic* reference obtained from the color chart. (C) Our result. (D) Ground truth image.

References

[1] Mathworks: Adjust color balance of RGB image with chromatic adaptation (chromadapt). `https://www.mathworks.com/help/images/ref/chromadapt.html`. Accessed: 2019-04-05.

[2] Casper Find Andersen and David Connah. Weighted constrained hue-plane preserving camera characterization. *IEEE Transactions on Image Processing*, 25(9):4329–4339, 2016.

[3] Matthew Anderson, Ricardo Motta, Srinivasan Chandrasekar, and Michael Stokes. Proposal for a standard default color space for the Internet—sRGB. In *Color and Imaging Conference*, 1996.

[4] Relja Arandjelovic and Andrew Zisserman. Three things everyone should know to improve object retrieval. In *CVPR*, 2012.

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

[6] Jonathan T Barron and Yun-Ta Tsai. Fast fourier color constancy. In *CVPR*, 2017.

[7] Roy S Berns and M James Shyu. Colorimetric characterization of a desktop drum scanner using a spectral model. *Journal of Electronic Imaging*, 4(4):360–373, 1995.

[8] Simone Bianco and Raimondo Schettini. Adaptive color constancy using faces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(8):1505–1518, 2014.

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

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

[11] A. Chakrabarti, Ying Xiong, Baochen Sun, T. Darrell, D. Scharstein, T. Zickler, and K. Saenko. Modeling radiometric uncertainty for vision with tone-mapped color images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(11):2185–2198, 2014.

[12] 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. *Journal of the Optical Society of America A*, 31(5):1049–1058, 2014.

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

[14] Brad Dayley and DaNae Dayley. *Adobe Photoshop CS6 Bible*. John Wiley & Sons, 2012.

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

[16] Mark S Drew and Brian V Funt. Natural metamers. *CVGIP: Image Understanding*, 56(2):139–151, 1992.

[17] Marc Ebner. *Color Constancy*. John Wiley & Sons, 2007.

[18] Graham Finlayson, Han Gong, and Robert B Fisher. Color homography: Theory and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 41(1):20–33, 2019.

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

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

[21] 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.

[22] 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, 2012.

[23] M.D. Grossberg and S.K. Nayar. What is the space of camera response functions? In *CVPR*, 2003.

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

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

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

[27] Seon Joo Kim, Hai Ting Lin, Zheng Lu, S. Süsstrunk, S. Lin, and M. S. Brown. A new in-camera imaging model for color computer vision and its application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(12):2289–2302, 2012.

[28] Seon Joo Kim and M. Pollefeys. Robust radiometric calibration and vignetting correction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(4):562–576, 2008.

[29] Haiting Lin, Seon Joo Kim, S. Süsstrunk, and M. S. Brown. Revisiting radiometric calibration for color computer vision. In *ICCV*, 2011.

[30] S. Lin, Jinwei Gu, S. Yamazaki, and Heung-Yeung Shum. Radiometric calibration from a single image. In *CVPR*, 2004.

[31] S. Lin and L. Zhang. Determining the radiometric response function from a single grayscale image. In *CVPR*, 2005.

[32] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[33] Seonghyeon Nam and Seon Joo Kim. Modelling the scene dependent imaging in cameras with a deep neural network. In *ICCV*, 2017.

[34] Rang M. H. Nguyen and Michael S. Brown. Raw image reconstruction using a self-contained sRGB–JPEG image with small memory overhead. *International Journal of Computer Vision*, 126(6):637–650, 2018.

[35] 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.

[36] Gaurav Sharma and Raja Bala. *Digital Color Imaging Handbook*. CRC press, 2017.

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

[38] Ying Xiong, K. Saenko, T. Darrell, and T. Zickler. From pixels to physics: Probabilistic color de-rendering. In *CVPR*, 2012.