Signal, Image and Video Processing (2023) 17:277–284
https://doi.org/10.1007/s11760-022-02230-2

ORIGINAL PAPER

Two-stage deep learning framework for sRGB image white balance

Marwa Farghaly[1] · Romany F. Mansour[1] · Adel A. Sewisy[2]

Received: 24 December 2021 / Revised: 14 March 2022 / Accepted: 29 March 2022 / Published online: 20 May 2022

Abstract
This work aims to correct white-balance errors in sRGB images. These white-balance errors are hard to fix due to the nonlinear color-processing procedures applied by camera image signal processors (ISP) to produce the final sRGB colors. Camera ISPs apply these nonlinear procedures after the essential white-balance step to render sensor raw images to the sRGB space through a camera-specific set of tone curves and look-up tables. To correct improperly white-balanced images, projecting non-linear sRGB colors back to their original raw space is required. Recent work formulates the problem as an image translation problem, where input sRGB colors are mapped using nonlinear polynomial correction functions to fix such white-balance errors. In this work, we show that correcting white-balance errors in sRGB images through a global color mapping followed by spatially local adjustments, learned in an end-to-end training, introduces perceptual improvements in the final results. Qualitative and quantitative comparisons with recently published methods for camera-rendered image white balancing validate our method's efficacy and show that our method achieves competitive results with state-of-the-art methods.

Keywords Color constancy · White balance · Color correction · Image enhancement

1 Introduction and related work

White balance is a basic procedure that is nearly applied to all images by the camera image signal processor (ISP). The goal of image white balancing is to remove undesirable color casts caused by scene lights. That is, this process aims to normalize the camera-rendered image's colors such that any achromatic object appears grayish (i.e., $R = G = B$) [1]. While it cannot fix all other colors, white-balance correction is often assumed to approximate the color constancy (i.e., object color appearance remains constant under different lighting conditions of the scene) [2]. Image white balancing does not target only the aesthetic aspect of camera-rendered images but also improves the accuracy of other computer vision tasks, such as image classification and image semantic segmentation [3–5].

Cameras apply white balance by first estimating the scene illumination color. This estimation is performed by a color constancy method, which typically falls into one of the following categories: (1) statistical methods (e.g., [6–8]) that use heuristic statistical-based hypotheses to estimate the illuminant color of the captured image, and (2) learning methods (e.g., [9–15]) that rely on machine learning techniques to learn to predict the illuminant color given the input image [10,13,14] or its color histogram [9,11,15]. Then, a global color channel scaling operation is applied to remove such undesirable color casts. This white-balance procedure is applied to camera sensor raw image (i.e., a linear representation of the incoming light), and afterwards camera ISPs apply a set of nonlinear procedures to render the final output image in the sRGB space [16,17]. Due to these nonlinear procedures, adopting the simple color scaling process—that is intended to fix linear raw images—does not work properly to fix camera-rendered images with white-balance errors [16].

An intuitive way to deal with such errors is to reconstruct a linear version of the input image followed by fixing its white

Romany F. Mansour and Adel A. Sewisy authors contributed equally to this study.

✉ Marwa Farghaly
marwa.farghaly@aun.edu.eg

Romany F. Mansour
romanyf@sci.nvu.edu.eg

Adel A. Sewisy
sewisy@aun.eun.eg

[1] Department of Mathematics, Faculty of Science, New Valley University, El-Kharga 72511, Egypt

[2] Deparment of Computer Science, Faculty of Computers and Information, Assiut University, Assiut University, Assiut 71515, Egypt

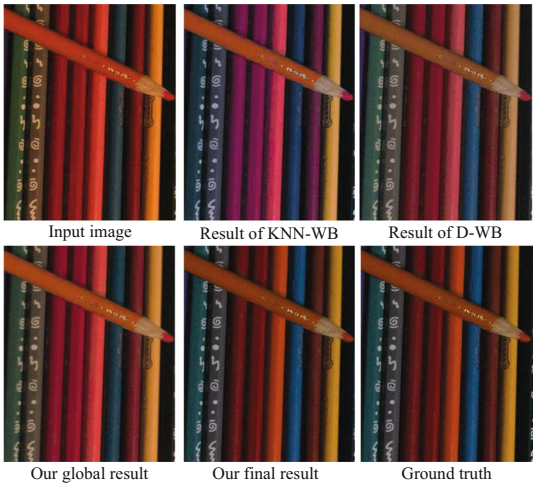

Fig. 1 This work focuses on fixing potential white-balance errors in camera rendered sRGB images. Our method consists of a two-stage learning framework that first applies a global correction to input image colors, and then a local process is applied to produce the final result. Our method produces competitive results compared to state-of-the-art methods (i.e., KNN-WB [16] and D-WB [20]) (color figure online)

balance, and then re-processing the image to its final form in the sRGB space. Though it seems trivial, this solution is impractical as camera ISP's procedures are unknown and a careful camera calibration process is required to accomplish this linearization task [16,17].

Recently, Afifi et al. [16,18–20] proposed solutions to deal with such white-balance errors in the camera-rendered sRGB images without a need for raw reconstruction. The authors in [16,18–20] proposed to process the input improperly white-balanced images in the sRGB space to generate the correctly white-balanced images. This sRGB-level processing was performed by a polynomial color mapping—e.g., the KNN white balance method (KNN-WB) [16] and the interactive white-balance method (I-WB) [19]—or via deep neural networks—e.g., the deep white balance methods (D-WB) [18,20].

Inspired by this research direction, our solution also proposes to avoid any raw reconstruction and processes the input image in its original space (i.e., the sRGB space). Due to the remarkable results achieved by deep learning methods in several research fields, such as computer networking and communications [21–23], we propose a deep learning-based method to solve our problem. In contrast with prior work (e.g., [16,18–20]), our proposed solution consists of two different stages to re-white balance the input image. The first stage estimates global mapping parameters to correct the input image colors without considering its spatial information. Then, the second stage locally processes our initial corrected image based on its spatial information to generate the final corrected image. Figure 1 shows our result compared to the state-of-the-art methods [16,20].

The major contributions of this work can be summarized as follows:

1. We propose a novel framework to fix white-balance errors in sRGB images through a two-stage correction procedure.
2. Unlike recent work that treats improperly white-balanced images through global color mapping operation, our two-stage framework first processes input image colors globally based on its color distribution to generate an initial corrected image. This initial solution is then improved by learning a residual layer to locally adjust our initial result to generate the final image. This two-stage strategy improves the perceptual results by considering local enhancements of the input image.

Extensive experiments are conducted on challenging test sets, and we show that our method produces competitive results when comparing with the state-of-the-art methods for correcting improperly white-balanced sRGB images. The rest of this paper is organized as follows. In Sect. 2, the methodology is presented. Section 3 presents the evaluation of our method through a set of experiments, ablation studies, and comparisons between the proposed method and the state-of-the-art methods. Finally, the paper is concluded in Sect. 4.

2 Method

We present a learning framework to correct the sRGB colors of input images that were rendered with camera white-balance errors. Figure 2 shows an overview of our framework. As shown, our framework consists of two steps to re-white balance the input sRGB image. In the first step, we apply a global mapping to the input sRGB image, I_{in}, in order to correct its colors in the sRGB space. This mapping process can be described as follows:

$$\hat{I}_{Gcrr} = C \quad \varphi\left(I_{in}\right), \tag{1}$$

$$C = N_1\left(H\left(I_{in}\right), \Theta_1\right), \tag{2}$$

where φ is a kernel operator that projects the three sRGB color channels into the high-dimensional space, as follows $\varphi\left(\langle R, G, B\rangle^T\right) = \langle R, G, B, RG, RB, GB, R^2, G^2, B^2\rangle^T$, and C is a 3×9 matrix generated by our network N_1 that accepts the color histogram $H(\cdot)$ of I_{in} and processes it with its trainable weights Θ_1.

In Eq. (2), we mentioned that our first network, N_1, accepts a color histogram feature of the input image, I_{in}. This histogram feature represents the color distribution of the input sRGB image, I_{in}. To create a dense histogram feature,

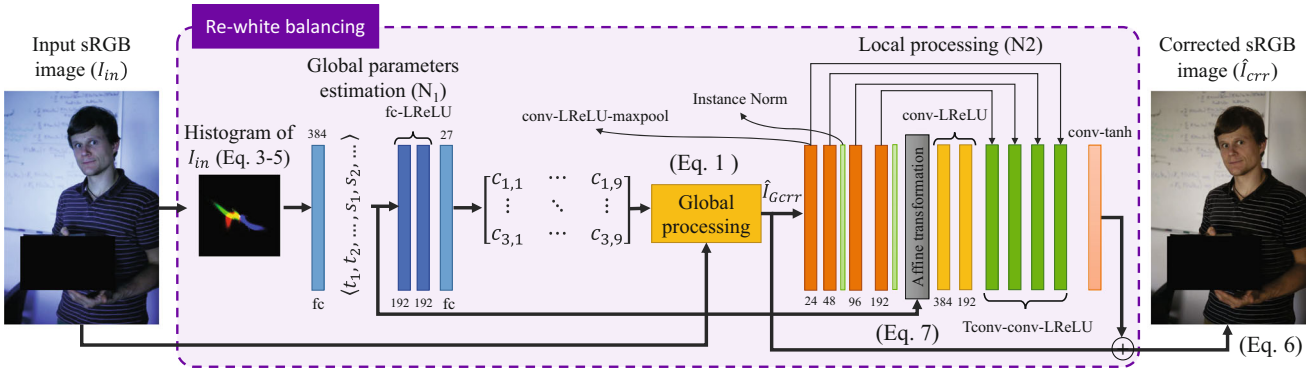

Fig. 2 Our re-white-balancing method processes the input image through two networks (N_1 and N_2). Our networks (i.e., N_1 and N_2) are trained in an end-to-end manner to produce the final sRGB image, $\hat{I}_{crr}$ with correctly white-balanced colors. See Sect. 2 for more details (color figure online)

we rely on the RGB-uv histogram proposed in prior work [1,13,16,24]. In particular, we create a $64 \times 64 \times 3$ histogram feature of the input image I_{in} in the the log-chrominance space [9,11,25] as follows:

$$H(I_{in}, d) = \sum_i I_{in_{y(i)}} abs(I_{in_{uj(i)}} - u) < \frac{\varepsilon}{2} \wedge \\ abs(I_{in_{vj(i)}} - v) < \frac{\varepsilon}{2}, \tag{3}$$

where d refers to $\langle u, v, c \rangle$, j refers to each color channel in the generated histogram feature, ε is the histogram's bin size, and $i = \{1, \ldots, n\}$ is the pixel index (here, n is the total number of pixels in the input image). The values of $I_{in_{y(i)}}$, $I_{in_{uj(i)}}$, $I_{in_{vj(i)}}$ are computed as follows:

$$\begin{aligned} I_{in_{y(i)}} &= \sqrt{I^2_{in_{R(i)}} + I^2_{in_{G(i)}} + I^2_{in_{B(i)}}}, \\ I_{in_{u1(i)}} &= \log\left(I_{in_{R(i)}}\right) - \log\left(I_{in_{G(i)}}\right), \\ I_{in_{v1(i)}} &= \log\left(I_{in_{R(i)}}\right) - \log\left(I_{in_{B(i)}}\right). \end{aligned} \tag{4}$$

Likewise, $I_{in_{u2}}$, $I_{in_{v2}}$, $I_{in_{u3}}$, and $I_{in_{v3}}$ are generated as follows:

$$\begin{aligned} &I_{in_{u2}} = -I_{in_{u1}},\ I_{in_{v2}} = -I_{in_{u1}} + I_{in_{v1}}, \\ &I_{in_{u3}} = -I_{in_{v1}},\ I_{in_{v3}} = -I_{in_{v1}} + I_{in_{u1}}. \end{aligned} \tag{5}$$

Finally, the histogram feature is normalized by the summation of the histogram's bin values.

This global correction is similar to what was proposed by prior work [16,19] in the sense that the work in [16,19] also uses global correction to map from the sRGB input colors to the corresponding sRGB ground truth colors. This global correction was computed based on a K nearest-neighbor search (KNN) in prior work [16]. In contrast, our proposed mapping is learned by a neural network and it is followed by our second step that locally processes the image. This is motivated by the fact that camera ISPs apply local tone mapping that results in spatially varying color changes in the final sRGB image [17]. Thus, we first generate our initial correction through a global mapping followed by a local processing that is applied in the second step.

The second step of our framework locally adjusts the fine-details of the globally corrected image, $\hat{I}_{Gcrr}$, via another neural network N_2. This fine-details adjustment is applied to improve the quality of our initial correction and to deal with over-saturated pixels that are hard to correct by solely global polynomial mapping. Our second network, N_2, accepts the output of the first step and produces a residual layer to generate our final output image $\hat{I}_{crr}$ as described in the following equation:

$$\hat{I}_{crr} = N_2\left(\hat{I}_{Gcrr}, \Theta_2\right) + \hat{I}_{Gcrr}, \tag{6}$$

where Θ_2 represents the trainable weights of our second network N_2.

2.1 Network architecture

As explained earlier, our framework consists of two neural networks (i.e., N_1 and N_2). The first network, N_1, accepts a $64 \times 64 \times 3$ histogram feature and produces the mapping parameters in C. This network, N_1, consists of four fully connected layers, as shown in Fig. 2. We use the leaky ReLU (LReLU) operator as our activate function applied to the output of second and third fully connected layers. A dropout rate of 0.5 is applied to the output of the third fully connected layer. The output fully connected layer has 27 output neurons to construct the mapping matrix C.

After processing the histogram feature of the input sRGB image, I_{in}, we produce the globally corrected image, $\hat{I}_{Gcrr}$ (as described in Eq. 1). This image is then fed to the second network, N_2, which is a U-Net-based network [26] with four encoder blocks, four decoder blocks, two bottleneck blocks and skip connections.

Table 1 Results on the Rendered WB dataset [16]

Method	MAE	△E 2000 [35]	Model size	Test time
Extrinsic test set (2881 *images*)				
GW [34]	8.89°	10.74	–	–
SoG [6]	9.54°	10.01	–	–
FC4 [32]	8.92°	12.12	5.89	0.13
Quasi-U CC [33]	12.95°	11.82	622 MB	0.56
D-WB [20]	3.75°	4.90	16.7	0.21
KNN-WB [16]	4.48°	5.60	21.8	0.12
I-WB [19]	5.35°	6.74	0.038	0.04
Ours	**4.33°**	**6.83**	**31.6**	**0.19**
Cube test set (10,242 *images*)				
GW [34]	6.85°	9.01	–	–
SoG [6]	6.69°	7.70	–	–
FC4 [32]	6.7°	10.4	5.89	0.13
Quasi-U CC [33]	6.35°	7.29	622	0.56
M-WB [31]	4.05°	4.89	5.10	0.23
D-WB [20]	3.45°	4.59	16.7	0.21
KNN-WB [16]	4.12°	5.68	21.8	0.12
I-WB [19]	4.64°	6.2	0.038	0.04
Ours	**3.77°**	**6.44**	**31.6**	**0.19**

In addition to our results, this table shows the results of the following methods: gray-world (GW) [34], shades-of-gray (SoG) [6], FC4 [32], quasi-unsupervised color constancy (Quasi-U CC) [33], interactive white balance (I-WB) [19], KNN white balance (KNN-WB) [16], and deep white-balance (D-WB) [20]. We used the following evaluation metrics: the mean angular error (MAE) and 4E 2000 [35]. We also report model sizes in mega-bytes of learning-based methods (including ours) and average testing GPU time to process a single image in seconds. Our results are indicated with boldface

Table 2 Ablation study on the effect of the networks (i.e., N_1 and N_2) on our final results

Method	MAE	△E 2000 [35]
Ours (w/o N_2)	10.12°	10.95
Ours (w/o N_1)	4.72°	6.97
Ours (w/o $\mathcal{L}_{per}$, $\lambda = 0$)	6.07°	8.31
Ours (w/ $\mathcal{L}_{per}$, $\lambda = 1$)	5.85°	7.94
Ours (w/ $\mathcal{L}_{per}$, $\lambda = 0.5$)	4.79°	7.25
Ours (w/ $\mathcal{L}_{per}$, $\lambda = 0.1$)	**4.33°**	**6.83**

We further report the results of our method trained with and without the perceptual loss $\mathcal{L}_{per}$. In these experiments, we used the Extrinsic Test Set in the Rendered WB dataset [16]. The best results are indicated with boldface

Each encoder block in N_2 consists of conv–LReLU–conv–LReLU-maxpool layers, each bottleneck block has conv–LReLU layers, and each decoder block consists of applying a 2D transposed conv (Tconv) operator followed by conv–LReLU–conv–LReLU layers. The first encoder block maps the input images into a latent space with 24 output channels. The output of each proceeding encoder block is doubled by a factor of 2 to reach 192 output channels by the last encoder block. After the second and fourth encoder blocks, we apply an instance normalization operation [27]. Then, the latent representation, $\mathcal{X}$, produced by the last encoder block is first processed by the latent feature generated by the first fully connected layer in N_1. This dual-use of the latent feature in N_1 helps the local network (i.e., N_2) to get some cues of the global color distribution in the image. We use the leaked feature from N_1 to apply an affine transformation to the latent representation $\mathcal{X}$ as follows:

$$\mathcal{X}'_{(i,j,f)} = t_f \mathcal{X}_{(i,j,f)} + s_f, \tag{7}$$

where $\{t_1, t_2, \ldots, t_{192}\}$ and $\{s_1, s_2, \ldots, s_{192}\}$ represent the latent feature vector produced by the first fully connected layer in N_1 (see Fig. 2).

Fig. 3 Qualitative results for post-capture white-balance correction on the Extrinsic Test Set of the Rendered WB dataset [16]. This figure shows the results of: (1) KNN-WB [16], (2) I-WB [19], and (3) our method. For each image, we show the corresponding ground truth correctly white-balanced sRGB image

After applying this affine transformation (Eq. 7), the output feature is processed by two bottleneck blocks with 384 and 192 output channels, respectively. Our decoders then processes this latent representation along with the skipped latent representation produced by each corresponding encoder blocks. Note that the skip connections are taken before applying the maxpool operator in each encoder block and concatenated with the result of the 2D transposed conv operator in each corresponding decoder block.

Finally, we apply a single conv layer with 1×1 kernel followed by a tanh activation function to generate a residual layer that is then used to generated the final output image $\hat{I}_{crr}$ (as described in Eq. 6).

2.2 Training details

We trained our networks (i.e., N_1 and N_2) for 100 epochs in an end-to-end manner to minimize the following loss function:

$$\mathcal{L} = \mathcal{L}_{N_1} + \mathcal{L}_{N_2} + \lambda \mathcal{L}_{per}, \tag{8}$$

where $\mathcal{L}_{N_2}$ is the L2 between our corrected image, $\hat{I}_{crr}$, and the ground-truth white-balanced image, I_{crr}, $\mathcal{L}_{N_1}$ is the L2 between the globally corrected image, $\hat{I}_{Gcrr}$, and the ground-truth white-balanced image, I_{crr}, $\mathcal{L}_{per}$ is the perceptual loss [28] between $\hat{I}_{crr}$ and I_{crr}, and λ and is a hyperparameter used to control the effect of $\mathcal{L}_{per}$ on the final loss. In our experiments, we set λ 0.1.

Note that as our framework is trained in an end-to-end scheme, the output 3×9 correction matrix from N_1 is first applied to the input image, and then the first loss term in Eq. (8), $\mathcal{L}_{N_1}$, is computed. This $\mathcal{L}_{N_1}$ loss term encourages the network to produce proper parameters in the output matrix to correct the colors of I_{in}. That is, the network learns the parameters of this matrix unsupervisedly. Afterward, the second network, N_2, receives the output of the globally corrected image, $\hat{I}_{Gcrr}$, and the second loss term, $\mathcal{L}_{N_2}$, encourages the second network, N_2, to correct local residual errors in $\hat{I}_{Gcrr}$ to get the final corrected image, $\hat{I}_{crr}$.

To optimize Eq. (8), we used Adam optimizer [29] with beta values 0.9 and 0.999 and learning rate of 10^{-4} dropped by a factor of 0.5 each 25 epochs. We used mini-batch size of 16 and regularized the weights Θ_1 and Θ_2 of our networks using L2 regularization with a multiplier of 10^{-5}.

In order to improve the training process, we interchangeably optimize Θ_1 to separately minimize $\mathcal{L}_{N_1}$ (by processing the input data by only N_1 and disabling N_2), and then we process the input data through the entire framework (i.e., N_1 and N_2) to minimize Eq. (8) at each iteration.

We train our framework using patch-wise training, where we randomly select 128×128 training patches from the full-size training images, while generate the histogram feature H from the entire training images. This allows our first network, N_1, to have global cues of the color distribution in the training image in order to predict suitable mapping parameters for each image. We followed the same procedure in the inference phase by using the histogram of the entire test image, while feeding the full-size test image to our second fully connected network, N_2.

3 Experimental results

We trained our network on the Intrinsic Set of the Rendered WB dataset [16]. This set includes 62,535 sRGB images rendered by several DSLR camera devices. For evaluation, we tested our method on the Extrinsic Test Set of the Rendered WB dataset [16], which includes 2881 images captured by a DSLR camera and several mobile phone cameras. Furthermore, we tested our method on the Cube Test Set that has 10,242 camera-rendered images with several white-balance settings [16,30].

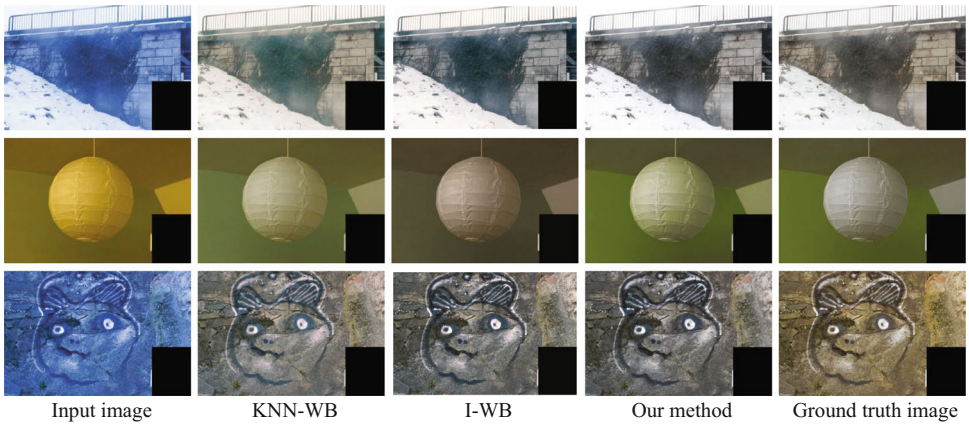

Fig. 4 Qualitative results for post-capture white-balance correction on the Cube Test Set of the Rendered WB dataset [16]. This figure shows the results of: (1) KNN-WB [16], (2) I-WB [19], and (3) our method. For each image, we show the corresponding ground truth correctly white-balanced sRGB image

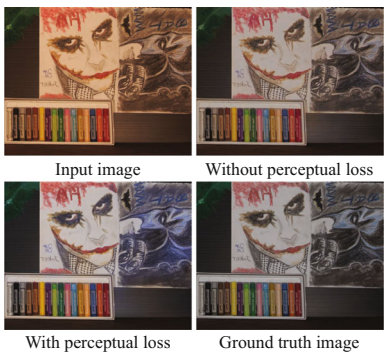

Fig. 5 Qualitative comparison of our results with and without the perceptual loss $\mathcal{L}_{per}$. For each image, we show the corresponding ground truth correctly white-balanced sRGB image

3.1 Comparisons

We compared our results with the recently published work for correcting improperly white-balanced images. In particular, we reported the results of the following methods: deep white-balance (D-WB) [20], KNN white balance (KNN-WB) [16], the interactive white balance (I-WB) [19], and mixed white-balance (M-WB) [31]. We further compare our method with other deep-learning-based and statistical-based color constancy methods: FC4 [32], quasi-unsupervised color constancy (Quasi-U CC) [33], the shades-of-gray (SoG) method [6], and the gray-world (GW) method [34] (Table 1).

We followed prior work [16,20] in adopting the mean angular error (MAE) and 4E 2000 [35] as our evaluation metrics. Table 3 shows the quantitative results. Finally, we show qualitative comparisons in Figs. 1, 3, and 4.

3.2 Ablation studies

We studied the behavior of our method with different options in design and loss function. The first part of Table 2 shows the results of our ablation study. We reported our results on the Extrinsic Set of the Rendered WB dataset [16] after training our network without the global and local processing networks; i.e., N_1, N_2, respectively. It is worth mentioning that when we disabled N_1, we substituted $\hat{I}_{Gcrr}$ with I_{in} in Eq. (6). In addition to this ablation study, we showed the results of our framework after training without the perceptual loss term, $\mathcal{L}_{per}$ (i.e., $\lambda = 0$), and with different values of λ, which controls the contribution of the perceptual loss in Eq. (8). The qualitative evaluation of our result with and without the perceptual loss is shown in Fig. 5. Finally, we studied the effect of different training mini-batch and image patch sizes on our results. The results of this study are shown in Table 3.

Table 3 Ablation study on the effect the mini-batch size and the input patch size on our results

Setting	MAE	△E 2000 [35]
Mini-batch size = 4, patch size = 128 × 128	8.68°	9.69
Mini-batch size = 12, patch size = 128 × 128	7.27°	8.12
Mini-batch size = 24, patch size = 128 × 128	5.89°	7.99
Mini-batch size = 48, patch size = 128 × 128	5.37°	7.76
Mini-batch size = 12, patch size = 256 × 256	8.67°	9.69
Mini-batch size = 24, patch size = 256 × 256	**4.33°**	**6.83**

In these experiments, we used the Extrinsic Test Set in the Rendered WB dataset [16]. The best results are indicated with boldface

4 Conclusion

We have presented a deep learning framework to correct sRGB images that were saved by camera ISPs with wrong white-balance settings. Our framework processes the input image through two main stages to first correct its colors by learning a global mapping parameters that map the input image colors to the corresponding corrected white-balanced ones. Then, we produce a learnable residual layer that locally adjusts our initial result. We have evaluated our two-stage correction method on large test sets of improperly white-balanced sRGB images and showed promising results that are on par or better compared to recently published methods for color constancy and image white balancing. One interesting direction in future work may be the use of XAI techniques [36,37] for understanding (and possibly improving) the behavior of the proposed network.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

References

1. Afifi, M.N.M.: Image color correction, enhancement, and editing. PhD thesis, York University (2021)
2. Gijsenij, A., Gevers, T., Weijer, J.V.D.: Computational color constancy: survey and experiments. IEEE Trans. Image Process. **20**(9), 2475–2489 (2011)
3. Afifi, M., Brown, M.S.: What else can fool deep learning? Addressing color constancy errors on deep neural network performance. In: ICCV, pp. 243–252 (2019)
4. Hussain, K.F., Afifi, M., Moussa, G.: A comprehensive study of the effect of spatial resolution and color of digital images on vehicle classification. IEEE Trans. Intell. Transp. Syst. **20**(3), 1181–1190 (2018)
5. Mansour, R.F., Escorcia-Gutierrez, J., Gamarra, M., Villanueva, J.A., Leal, N.: Intelligent video anomaly detection and classification using faster RCNN with deep reinforcement learning model. Image Vis. Comput. 104229 (2021)
6. Finlayson, G.D., Trezzi, E.: Shades of gray and colour constancy. In: Color and Imaging Conference, vol. 2004, pp. 37–41 (2004)
7. Weijer, J.V.D., Gevers, T., Gijsenij, A.: Edge-based color constancy. IEEE Trans. Image Process. **16**(9), 2207–2214 (2007)
8. Finlayson, G.D.: Corrected-moment illuminant estimation. In: ICCV, pp. 1904–1911 (2013)
9. Barron, J.T.: Convolutional color constancy. In: ICCV (2015)
10. Bianco, S., Cusano, C., Schettini, R.: Color constancy using CNNs. In: CVPR Workshops, pp. 81–89 (2015)
11. Barron, J.T.: Tsai, Y.: Fast Fourier color constancy. In: CVPR, pp. 886–894 (2017)
12. Bianco, S., Cusano, C., Schettini, R.: Single and multiple illuminant estimation using convolutional neural networks. IEEE Trans. Image Process. **26**(9), 4347–4362 (2017)
13. Afifi, M., Brown, M.S.: Sensor-independent illumination estimation for DNN models. In: BMVC (2019)
14. Hernandez-Juarez, D., Parisot, S., Busam, B., Leonardis, A., Slabaugh, G., McDonagh, S.: A multi-hypothesis approach to color constancy. In: CVPR, pp. 2270–2280 (2020)
15. Afifi, M., Barron, J.T., LeGendre, C., Tsai, Y., Bleibel, F.: Cross-camera convolutional color constancy. arXiv preprint arXiv:2011.11890 (2020)
16. Afifi, M., Price, B., Cohen, S., Brown, M.S.: When color constancy goes wrong: correcting improperly white-balanced images. In: CVPR, pp. 1535–1544 (2019)
17. Afifi, M., Abdelhamed, A., Abuolaim, A., Punnappurath, A., Brown, M.S.: Cie xyz net: Unprocessing images for low-level computer vision tasks. IEEE Trans. Pattern Anal. Mach. Intell. 1–1 (2021)
18. Afifi, M.: Semantic white balance: Semantic color constancy using convolutional neural network. arXiv preprint arXiv:1802.00153 (2018)
19. Afifi, M., Brown, M.S.: Interactive white balancing for camera-rendered images. In: Color and Imaging Conference (2020)
20. Afifi, M., Brown, M.S.: Deep white-balance editing. In: CVPR, pp. 1397–1406 (2020)
21. O'shea, T., Hoydis, J.: An introduction to deep learning for the physical layer. IEEE Trans. Cogn. Commun. Netw. **3**(4), 563–575 (2017)
22. Aceto, G., Ciuonzo, D., Montieri, A., Pescapé, A.: Mobile encrypted traffic classification using deep learning: experimental evaluation, lessons learned, and challenges. IEEE Trans. Netw. Serv. Manag. **16**(2), 445–458 (2019)
23. Hussien, M., Ahmed, M.F.A., Dahman, G., Nguyen, K.K., Cheriet, M., Poitau, G.: Towards more reliable deep learning-based link adaptation for WiFi 6. In: ICC 2021-IEEE International Conference on Communications, pp. 1–6 (2021)
24. Afifi, M., Brubaker, M.A., Brown, M.S.: HistoGAN: Controlling colors of GAN-generated and real images via color histograms. In: CVPR (2021)
25. Finlayson, G.D., Hordley, S.D.: Color constancy at a pixel. J. Opt. Soc. Am. A (JOSA A) **18**(2), 253–264 (2001)
26. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention, pp. 234–241 (2015)
27. Ulyanov, D., Vedaldi, A., Lempitsky, V.: Instance normalization: The missing ingredient for fast stylization. arXiv preprint arXiv:1607.08022 (2016)
28. Johnson, J., Alahi, A., Li, F.: Perceptual losses for real-time style transfer and super-resolution. In: ECCV, pp. 694–711 (2016)
29. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
30. Banić, N., Koščević, K., Lončarić, S.: Unsupervised learning for color constancy. arXiv preprint arXiv:1712.00436 (2017)
31. Afifi, M., Brubaker, M.A., Brown, M.S.: Auto white-balance correction for mixed-illuminant scenes. In: WACV, pp. 1210–1219 (2022)
32. Hu, Y., Wang, B., Lin, S.: FC4: Fully convolutional color constancy with confidence-weighted pooling. In: CVPR, pp. 4085–4094 (2017)

33. Bianco, S., Cusano, C.: Quasi-unsupervised color constancy. In: CVPR, pp. 12212–12221 (2019)
34. Buchsbaum, G.: A spatial processor model for object colour perception. J. Franklin Inst. **310**(1), 1–26 (1980)
35. Sharma, G., Wu, W., Dalal, E.N.: The CIEDE2000 color-difference formula: implementation notes, supplementary test data, and mathematical observations. Color Res. Appl. **30**(1), 21–30 (2005)
36. Adadi, A., Berrada, M.: Peeking inside the black-box: a survey on explainable artificial intelligence (XAI). IEEE Access **6**, 52138–52160 (2018)
37. Nascita, A., Montieri, A., Aceto, G., Ciuonzo, D., Persico, V., Pescapé, Antonio: XAI meets mobile traffic classification: understanding and improving multimodal deep learning architectures. IEEE Trans. Netw. Serv. Manag. **18**(4), 4225–4246 (2021)