Dense Color Constancy with Effective Edge Augmentation

Yilang Zhang, Zheng Wei, Jian Wang*
Fudan University
{yilangzhang16,zwei15,jianwang}@fudan.edu.cn

Xin Yuan
Bell Lab
xyuan@bell-labs.com

Abstract

Recently, computational color constancy via convolutional neural networks (CNNs) has received much attention. In this paper, we propose a color constancy algorithm called the Dense Color Constancy (DCC), which employs a self-attention DenseNet to estimate the illuminant based on the 2D log-chrominance histograms of input images and their augmented edges. The augmented edges help to tell apart the edge and non-edge pixels in the log-histogram, which largely contribute to the feature extraction and color ambiguity elimination, thereby improving the accuracy of illuminant estimation. Experiments on benchmark datasets show that the DCC algorithm is very effective for illuminant estimation compared to the state-of-the-art methods.

1. Introduction

Color is an essential cue for studying images. The color reflected in an image is determined by the intrinsic properties of the object, surface and light source [23]. To obtain the color of an object under the standard light source (i.e., the white light), one has to eliminate the chromatic aberration caused by the light sources, which constitutes the goal of computational color constancy. Over the years, computational color constancy has been a long-standing problem in many fields, such as the visual science and computer vision. While many existing methods have been demonstrated to be effective in solving this problem, there are still challenges in both accuracy as well as computational efficiency.

Generally speaking, existing color constancy methods assume regularities for the color of a natural object observed under the white light. For example, the most simply designed Gray World [11] algorithm uses the assumption that the average reflectance in a scene is achromatic under a neutral light source. Also, the arithmetic mean has been generalized via higher order mathematical calculations [39]. These hypothesis-based algorithms are often classified as the learning-free algorithms. In recent years, with the innovative deep learning networks making a splash in computer vision, they have greatly promoted the research of learning-based methods for color constancy. Indeed, since the paper [8] debuted in 2015, which firstly used the convolutional neural network (CNN) to solve the color constancy problem, there have been many works that use deep learning frameworks for this task [5, 6, 27, 37]. Notably, some of them have achieved satisfactory performance on standard color constancy benchmark datasets [15, 22].

In this paper, we present a color constancy algorithm called the Dense Color Constancy (DCC). Our algorithm is inspired by the recent successful approaches [5, 6] that reformulated the color constancy problem as a two-dimensional (2D) spatial localization task in the log-chrominance space. Although the new formulation can largely simplify the problem by reducing the number of underlying parameters to be estimated, it also raises the issue that the spatial information of input images, which usually offers significant cues for illuminant estimation, is missing.
in the log-chrominance space.

To address the issue, we use an edge augmentation operator that can well preserve the spatial information when translated into the log-chrominance space. Specifically, the operator makes better use of the gradient information through the augmented edge while retaining the non-edge pixels of the original image. Moreover, the edge augmentation is capable of eliminating the color ambiguity of edges. Thus, we feed both the original images and their augmented edges (in the log-chrominance space) as inputs into the self-attention DenseNet for illuminant estimation. An example of an image restored by DCC is shown in Figure 1.

Not only can our network take full advantage of the spatial information of original image, it also confers many other merits, such as the end-to-end training, adaptive processing for images with arbitrary sizes, and robustness of the final estimation. Through experiments on the reprocessed Color Checker dataset [36], it is demonstrated that the proposed method has very competitive performance compared to the state-of-the-art learning-based methods. Meanwhile, our method is also flexible enough to reach satisfactory performance on the NUS 8-camera dataset [15].

2. Related Work

Roughly speaking, the computational color constancy methods can be grouped into two major categories: i) those being learning-free and ii) those relying on learning frameworks. The former typically assumes some particular statistical or physical priors of natural images, such as the unified Minkowski norm restrictions [39] and Dichromatic reflection model [33, 35]. Methods in this category are irrelevant to the dataset and camera information and also have no need of training. Thus, most of them can be performed very efficiently. However, due to the imprecision of priors and random fluctuation of statistics in varieties of practical images, which often lead to biased and noisy results, the learning-free methods may have unsatisfactory performance.

In the second category, the learning-based methods aim at constructing a model from the training data, which essentially search over the entire assumption space for the best prior. Early learning-based methods often employed simple structures and algorithms for illuminant estimation. In [13, 16, 19, 21], for example, some plain features were extracted manually to regress the illuminant prediction with linear regression or support vector machine (SVM). In [29], nearest neighbor methods were employed to solve the illuminant prediction problem. While these methods have demonstrated more or less advantages over the learning-free ones, their manufactured features still fail to characterize the whole information of images.

To better extract the information of images, Bianco et al. [8] first proposed to use the CNN for semantic feature extraction in illuminant estimation. Since then, many variants of CNNs have been employed to perform this task. For example, Shi et al. [47] proposed a two-branch structure to generate prediction from two illuminant hypotheses. [12] determined the illuminant from objects whose colors are learnt through object recognition. [7, 27] took advantage of image segmentation, which assigns different weights to a mask for different pixels of the image. In [8, 9, 37], the patch-based methods were utilized to obtain the global estimation from local candidates. In summary, by exploiting the spatial structure of images that contains significant semantic information, the CNN-based methods have achieved much improvement over the previous ones. However, it should be noted that these methods essentially perform under the multiplicative constraints [5], which might be too complex for some practical scenarios.

To address the multiplicative constraints, Barron [5, 6] reformulated the color constancy problem in a much simpler way. Specifically, images represented in RGB channels are transformed to the 2D log-chrominance histograms. Interestingly, this operation reduces the number of underlying parameters to be estimated from three to two, thereby greatly simplifying the problem. Moreover, since the varying of illuminant is equivalent to the linear shift in the space of log-chrominance, the multiplicative constraints are naturally translated into linear ones [5].

3. Dense Color Constancy

3.1. Overview

Problem formulation For an RGB image, the image value of pixel $k$ under the Lambertian assumption follows:

$$I_c^k = \int_{\omega} L^k(\lambda) \rho(\lambda) S^k(\lambda) d\lambda, \quad c \in \{r, g, b\}, \quad (1)$$

where $\lambda$ is the wavelength of the light, which belongs to the range $\omega$, $L^k(\lambda)$ is the light source to be estimated, $\rho(\lambda)$ is the sensitivity function of the camera, and $S^k(\lambda)$ is the surface reflectance of specific object at pixel $k$.

For a single light source, it is assumed that the illuminant has constant values:

$$L^k(\lambda) \equiv (L_r, L_g, L_b) \quad (2)$$

on three channels for all pixels of the image. Then, (1) reduces to a diagonal model called the von Kries Model [41]. The goal of color constancy becomes to correct the image to what it should be under the canonical illuminant:

$$L_{\text{canon}} = \left(\frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}, \frac{1}{\sqrt{3}}\right), \quad (3)$$

where $L_{\text{canon}}$ is normalized because the correction is only for the chrominance, rather than for the brightness level [24]. In a nutshell, the color constancy methods perform two steps: i) estimate the illuminant for RGB channels
from an image and ii) correct each pixel of the image by eliminating the effect caused by the estimated illuminant.

**Metric** Following previous works (e.g., [5][8]), we use the threefold cross-validation to evaluate the performance of our method. For an input image, the evaluation criteria is based on the angular error (in degree) introduced by Hornby and Finlayson [26]. In particular, the angle between the RGB triplet of the estimated illuminant \( \hat{L} \) and that of the measured ground-truth illuminant \( L \) is given by:

\[
E(L, \hat{L}) = \frac{180^\circ}{\pi} \arccos \left( \frac{\hat{L}^T \hat{L}}{\| \hat{L} \|_2 \| \hat{L} \|_2} \right). \tag{4}
\]

We consider the following five metrics, which are commonly used in the color constancy literature: i) mean, ii) median, iii) tri-mean of all the errors, iv) mean of the lowest 25% errors, and v) mean of the highest 25% errors. In addition, we provide the 95th quantile error.

**Transformation to log-chrominance** Following [5], we project the images onto the UV space of log-chrominance. The log-chrominance \( u \) and \( v \) for pixel \( k \) of the original image are defined as:

\[
I^u_k = \log \left( \frac{I^k_u}{I^k_g} \right) \text{ and } I^v_k = \log \left( \frac{I^k_v}{I^k_b} \right), \tag{5}
\]

respectively. Likewise, the illuminant in the UV space, whose scale we are not concerned about, is given by:

\[
L_u = \log \left( \frac{L_u}{L_g} \right) \text{ and } L_v = \log \left( \frac{L_v}{L_b} \right), \tag{6}
\]

respectively. With the estimation of \( L_u \) and \( L_v \), the normalized RGB illuminant can be recovered through:

\[
L_r = \frac{\exp(L_u)}{z}, \quad L_g = \frac{1}{z} \text{ and } L_b = \frac{\exp(L_v)}{z}, \tag{7}
\]

where

\[
z = \sqrt{\exp(L_u)^2 + \exp(L_v)^2 + 1}.
\]

From [5], the chrominance of all pixels in an image can be put into a histogram comprised of small bins. In this histogram, \( M_I(u, v) \) counts the number of pixels in image \( I \) whose chrominance is close to the values \( (u, v) \):

\[
M_I(u, v) = \sum_{k \in I} \left[ \left| I^u_k - u \right| \leq \frac{\epsilon}{2} \land \left| I^v_k - v \right| \leq \frac{\epsilon}{2} \right], \tag{8}
\]

where \( \epsilon \) controls the size of each bin in the histogram. Thus, the multiplicative change of illuminant in the RGB space can be translated to the additive change of illuminant in the UV space. To estimation the illuminant of an input image, we only need to detect the linear shift of the log-chrominance in the 2D UV space.

**Method** The major steps of the proposed algorithm are summarized as follows. We first perform the edge augmentation for each image. Next, by translating the images and their augmented edges to the UV space, we stack them as two channels of inputs to our network. Then, we employ a self-attention DenseNet to estimate the illuminant for each input, where the self-attention is employed to reweight the feature maps. Finally, from the network output \((L_u, L_v)\), we obtain the illuminant estimation; See [7] for details. The architecture of our DCC algorithm is shown in Figure 2.

### 3.2. Edge Augmentation and Utilization

Edge augmentation methods have been extensively used for data augmentation. For those performing in the RGB space [7][27], edges are often augmented to help segment images. Whereas for those performing in the UV space [5][6], the augmented edges can be used to measure the local gradients of images, which actually reflect their spatial statistics. In contrast to the previous researches that abandon non-edge pixels (internal color blocks) when extracting edges, we augment the edges while preserving the non-edge pixels. Later on, we shall show the advantages of retaining the non-edge pixels.

In image processing and computer vision [31][40], Sobel filter [38] has been widely used for edge extraction owing to its accurate estimation of gradients and relatively inexpensive computation. To preserve the non-edge pixels, we use a modified Sobel filter as our edge augmentation operator:

\[
f_x = \begin{bmatrix}
-1 & 0 & 1 \\
-2 & \sigma & 2 \\
-1 & 0 & 1
\end{bmatrix} \text{ and } f_y = f_x^T, \tag{9}
\]

which extract the gradient information of images in horizontal \( x \) and vertical \( y \), respectively. The intensity and angle of augmented edge for channel \( c \in \{r, g, b\} \) are given by

\[
E_c = \sqrt{(f_x \ast I_c)^2 + (f_y \ast I_c)^2} \tag{10}
\]

and

\[
\Theta_c = \arctan \left( \frac{f_y \ast I_c}{f_x \ast I_c} \right), \tag{11}
\]

respectively, where “\( \ast \)” denotes the convolution operator. To ensure that our augmented edges are robust to image rotation, only the intensity part is retained as the final augmented edge. We stress that the hyper-parameter \( \sigma \) in (9) controls the relative proportion between the gradient information and internal color block information of images to be extracted. It is easily seen that the modified operator reduces to the conventional Sobel filter [38] when \( \sigma = 0 \).

Furthermore, we can split the operator into two parts:

\[
f_x = f_x^{\text{grad}} + f_x^{\text{img}}, \tag{12}
\]
where

\[
\begin{bmatrix}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1
\end{bmatrix}
\]
and

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & \sigma & 0 \\
0 & 0 & 0
\end{bmatrix}
\]

(13)

represent the gradient-component and image-component operator, respectively. When the modified operator is applied to the pixels located at the edge in the vertical direction, it is clear that \( f_{\text{grad}} \ast I_c \) dominates \( f_{\text{img}} \ast I_c \). Hence,

\[
f_{x} \ast I_c \approx f_{x}^{\text{grad}} \ast I_c,
\]

which becomes the local gradient estimation. Similarly, when this operator works on the non-edge pixels, we have

\[
f_{x} \ast I_c \approx f_{x}^{\text{img}} \ast I_c = \sigma I_c,
\]

i.e., the preserved non-edge pixels scaled with a scalar. Since the color of edge pixels is the mixture and blur of their surroundings, it can be viewed as the noise. Whereas, the gradient of edge pixels provides us with the spatial and distributional information. The same rationale also applies to the operator \( f_{y} \).

To utilize the edge information and at the same time get rid of the noise brought in by the edge, we transform the image and its augmented edge into two log-histograms:

\[
M_{\text{img}}(u, v) = \sum_{k \in \text{img}} \left[ |I_{u}^{k} - u| \leq \frac{\epsilon}{2} \land |I_{v}^{k} - v| \leq \frac{\epsilon}{2} \right]
\]

(16)

and

\[
M_{\text{edge}}(u, v) = \sum_{k \in \text{edge}} \left[ |I_{u}^{k} - u| \leq \frac{\epsilon}{2} \land |I_{v}^{k} - v| \leq \frac{\epsilon}{2} \right]
\]

(17)

which act as two channels of the network inputs. Different from [5, 6], whose channels are parallel and irrelevant, our channels are interrelated owing to the dense connection structure of our network. Ideally, when the chrominance of edge pixels and the non-edge pixels are not overlapped in the log-histogram, the following features can be readily taken advantages of:

i) \( \min \left\{ M_{\text{img}}(u, v), M_{\text{edge}}(u, v)/(\sqrt{2}\sigma) \right\} \): The common chrominance of the image and its edge corresponds to the non-edge pixels in the RGB space, which we preserve in our augmented edge channel. The denominator \( \sqrt{2}\sigma \) is a normalization factor derived from (10) and (15).

ii) \( \max \left\{ M_{\text{edge}}(u, v)/(\sqrt{2}\sigma) - M_{\text{img}}(u, v), 0 \right\} \): After removing the chrominance of non-edge pixels from \( M_{\text{edge}} \), we obtain the chrominance of the augmented edge pixels, which reflects the statistical information of the gradients in the edge.

iii) \( \max \left\{ M_{\text{img}}(u, v) - M_{\text{edge}}(u, v)/(\sqrt{2}\sigma), 0 \right\} \): Likewise, after removing the chrominance of non-edge pixels from \( M_{\text{img}} \), what remains is the chrominance
of edge that is not augmented, i.e., that of the noisy pixels to be eliminated.

When using the hand-crafted features, we may ignore some sophisticated or effective ones. Instead, we just put $M_{\text{img}}$ and $M_{\text{edge}}$ into the network, and let the network learn and select the features by itself. Since the two chrominances are more or less overlapped, they may lead to some untrustworthy features. To ease the effect of these untrustworthy features, we introduce a self-attention module to our network, which will be discussed in Section 3.4.

3.3. Network Architecture

A sketch of the DCC network is shown in Figure 2. The network follows a similar structure as in DenseNet-121 [28], which has been widely used for solving computer vision problems. We draw inspiration from this network due to the following two reasons. First of all, according to the design of DenseNet-121, the feature-maps generated from the preceding layers can be passed through all the subsequent layers. Even if some of the features are lost during the propagation process, they can still be regenerated at the input of latter layers through the dense connections [34]. Since the shallow features play an important role in illuminant estimation, DenseNet-121 is well suited for extracting the features of images in this task. Secondly, the feature maps from different layers have various receptive fields. Since each layer can get feature maps from all preceding layers through dense connections, the final feature outputs become more effective after aggregating different sizes of receptive fields. Moreover, since DenseNet-121 implements shorter connections, and also because we set the growth rate $K = 12$, the feature maps of each layer are relatively small so that DenseNet-121 can complete this task with fewer parameters.

DenseNet-121 has four dense blocks, whose numbers of the dense layers are 6, 12, 24 and 16, respectively. For a dense layer, it contains the same sequence operations: Batch Normalization, ReLU function, and Convolution. The last layer of the network is a fully connected layer. We do not directly use the evaluation metric in (4) as the loss function, since otherwise

$$\lim_{L \to L^*} \frac{\partial}{\partial L} \left( \arccos \frac{L^T \hat{L}}{\|L\|_2 \|\hat{L}\|_2} \right) = \infty,$$

(18)

that is, the gradient value may overflow when the prediction is close to the ground-truth. Instead, we define the loss function $\mathcal{L}$ of the network in terms of the cosine of the angle between the network estimation $\hat{L}$ and the ground-truth illuminant $L$:

$$\mathcal{L} := 1 - \frac{L^T \hat{L}}{\|L\|_2 \|\hat{L}\|_2},$$

(19)

where the 1 on the right-hand side ensures $\mathcal{L}$ to be positive.

3.4. Self-attention

Attention helps to increase the representation power of images by highlighting critical features while suppressing unnecessary ones. In general, the attention module for images considers both the spatial and channel information. The spatial information of an image tells where to focus, while the channel information helps to characterize what we are interested in. Inspired by [7, 27], our network employs the Convolutional Block Attention Module (CBAM) [42], which well incorporates these two types of information.

For the color constancy problem, recall from [35] that if many pixels in an image are of nearly identical RGB values, they will appear as an impulse when translating to the UV space. The impulse, however, may have a negative effect on the illuminant estimation due to the chromatism caused. Nevertheless, CBAM alleviates the negative effect by assigning lower weights to the features of these regions. The same operation can also be applied to those untrustworthy features mentioned in Section 3.2. In our network, we put CBAM behind the last DenseBlock, because the deeper the network layer, the more effective the output feature information will be. This is somewhat similar to [27], where a confidence layer is added before the fully connected layer.

4. Experiments

4.1. Datasets

We evaluate the performance of our method on two standard color constancy datasets: i) the Color Checker dataset reprocessed by Shi and Funt [22, 36] and ii) the NUS 8-camera dataset from Cheng et al. [15]. The former contains 568 images taken from two cameras, while the latter has 1736 images (of larger size) obtained by 8 different cameras, each of which takes about 220 ones. For images from both datasets, the Macbeth Color Inspector (MCC) is utilized to capture the ground-truth, and the datasets provide the corners of the MCC. By setting $L_c = (0, 0, 0)$ to mask the MCC, we train and test the rest of areas in the image, which are not otherwise specially processed.

4.2. Preprocessing and Random Patches

**Preprocessing** The resolution of images plays an important role in many fields of computer vision, such as image recognition [25] and segmentation [2]. For illuminant estimation, however, we mostly care about the color of images, rather than the resolution. Thus, like many learning-based color constancy methods, we downsample the images taken from high quality digital single-lens reflex (DSLR) cameras to 256px × 384px in order to accelerate our training process. Besides, following the instruction in [15, 36], we subtract the black level of cameras and abandon the pixels in images that are above the saturation level of 0.98.
### Table 1: Performance of the DCC variants on the reprocessed Color Checker dataset, where the best results are highlighted with gray background.

| Method                                      | Mean | Med. | Tri. | Best 25% | Worst 25% | 95% Quant. | Avg. |
|----------------------------------------------|------|------|------|----------|-----------|------------|------|
| 1 DenseNet-121, \( p = 16 \)                | 1.99 | 1.39 | 1.52 | 0.45     | 4.54      | 5.64       | 1.54 |
| 2 DenseNet-121, CBAM, \( p = 16 \)          | 1.92 | 1.23 | 1.36 | 0.37     | 4.66      | 5.97       | 1.41 |
| 3 DenseNet-121, CBAM, \( p = 16 \), Edge in FFCC | 1.79 | 1.14 | 1.28 | 0.34     | 4.32      | 5.27       | 1.31 |
| 4 DenseNet-121, CBAM, \( p = 0, \sigma = 1/\sqrt{2} \) | 1.83 | 1.16 | 1.27 | 0.31     | 4.51      | 6.15       | 1.30 |
| 5 DenseNet-121, CBAM, \( p = 8, \sigma = 1/\sqrt{2} \) | 1.75 | 1.12 | 1.25 | 0.34     | 4.20      | 5.03       | 1.28 |
| 6 DenseNet-121, CBAM, \( p = 16, \sigma = 1/\sqrt{2} \) | 1.74 | 1.13 | 1.23 | 0.27     | 4.17      | 5.26       | 1.22 |
| 7 DenseNet-121, CBAM, \( p = 32, \sigma = 1/\sqrt{2} \) | 1.76 | 1.21 | 1.30 | 0.32     | 4.22      | 5.48       | 1.30 |
| 8 DenseNet-121, CBAM, \( p = 64, \sigma = 1/\sqrt{2} \) | 1.75 | 1.13 | 1.25 | 0.34     | 4.24      | 5.30       | 1.29 |
| 9 DenseNet-121, CBAM, \( p = 16, \sigma = 0 \) | 1.79 | 1.16 | 1.26 | 0.34     | 4.35      | 5.69       | 1.31 |
| 10 DenseNet-121, CBAM, \( p = 16, \sigma = 1 \) | 1.75 | 1.08 | 1.24 | 0.33     | 4.33      | 5.14       | 1.27 |
| 11 DenseNet-121, CBAM, \( p = 16, \sigma = \sqrt{2} \) | 1.85 | 1.24 | 1.32 | 0.32     | 4.49      | 5.79       | 1.34 |

**Patch-based improvement** To enlarge the datasets and also increase the robustness and generalization of our network, the patch-based methods are implemented. For both the training and test sets, we randomly sample multiple patches from each image as the augmented images. Specifically, we determine the height and width of the augmented images by multiplying a random number in \([0.5, 1]\) to those of the original images. Also, we randomly choose the locations for the augmented images, while ensuring that they do not cross the borders of the original image.

While the original images and their augmented patches are fed into our network with their ground-truth in the training process, they are also used to generate the local candidates of the illuminant in the test process. From these local candidates, we finally obtain the global illuminant estimation. More specifically, we compute the channel-wise median of the local predictions, followed by a normalization, to be our global prediction. According to our experiments, however, the random patches may contain some ambiguous areas, such as the yellow wall under white illuminant or the white wall under yellow illuminant \([27]\). In this case, inaccurate illuminant prediction could be made due to these misleading areas. Nevertheless, this issue can be overcome by using the median angular error, rather than the mean angular error. As shown in Figure 3, the median angular error can better handle the situation where there are many biased local candidates, whose distribution is left-right asymmetry around the ground-truth. Of course, there is still a trade-off between the accuracy and stability of the prediction, which can be controlled by the number of patches.

Apart from random sampling of patches, we also randomize the color of patches through a channel-wise scaling. Each channel of the patch is multiplied with a random variable in \([0.5, 1]\). The same multiplier is also applied to its ground-truth. As mentioned, the multiplicative change of the RGB color is equivalent to the linear shift of its log-histogram. Thus, the color randomization also forces our network to discern the shift in the log-histogram and learn the translation equivariance.

**4.3. Implementation**

The proposed DCC method is implemented by TensorFlow \([11]\). We train our model on the server with GTX 1080 Ti in an end-to-end manner. Mini-batch gradient descent is performed with a batch size of 64 and 1000 training epochs in the framework of Adam optimizer \([32]\). We set the initial learning rate to \(10^{-3}\) and the learning rate decay to 0.1, where the decay does not work until the model runs to 90%...
of the epochs, which can slightly improve the convergence speed and stability of our model. In addition, the dropout layer, which is often used to avoid overfitting, is not included in our model since there are already batch normalization layers in each dense block.

4.4. Results

**Internal comparison** To evaluate the effectiveness of edge channel and CBAM structure in our algorithm, we conduct experiments for three cases: i) neither of edge channel nor CBAM, ii) only CBAM, and iii) both of them are used. In addition, the case where the edge augmentation layers in each dense block.

Table 2: Performance comparison with previous methods on the reprocessed Color Checker and NUS 8-camera datasets. For both datasets, the mean, median, tri-mean, best 25%, worst 25%, 95% quantile angular errors are used as performance metrics. In addition, the geometric mean of the former five metrics (except the worst-case metric) is tested for comparative purpose. The results are provided in Table 2. It can be observed that for the reprocessed Color Checker dataset, DCC performs the best among all algorithms. In particular, DCC makes much improvement for the worst-case metric (i.e., the worst 25%), which clearly demonstrates the stability of the DCC algorithm. The improvement can be attributed to the following reasons. Firstly, the reweighted feature map by CBAM effectively reduces the effect of noisy pixels in the image. Secondly, the random patches can help to reduce the variance of estimation, while using the median of local candidates as the global estimation can lower the bias of estimation. For the NUS 8-camera dataset, the performance of DCC is slightly inferior to that of the state-of-the-art methods. This is because DCC employs a deep structure, which may not benefit very much from the relatively small NUS 8-camera dataset.

In Figure 4, we visualize i) the original RAW images, ii) the restored images from the ground-truth illuminant, iii) the restored images based on the DCC prediction, and iv) the corresponding chrominance. In the chrominance column, the red crosses signify the ground-truth while the blue ones represent our prediction, where a Gamma correction extractor $\sigma$ for given $p = 16$. It can be observed that the model achieves the best results when $\sigma = 1/\sqrt{2}$.

**External comparisons** We compare DCC with previous methods on both datasets. The results are provided in Table 2. It can be observed that for the reprocessed Color Checker dataset, DCC performs the best among all algorithms. In particular, DCC makes much improvement for the worst-case metric (i.e., the worst 25%), which clearly demonstrates the stability of the DCC algorithm. The improvement can be attributed to the following reasons. Firstly, the reweighted feature map by CBAM effectively reduces the effect of noisy pixels in the image. Secondly, the random patches can help to reduce the variance of estimation, while using the median of local candidates as the global estimation can lower the bias of estimation. For the NUS 8-camera dataset, the performance of DCC is slightly inferior to that of the state-of-the-art methods. This is because DCC employs a deep structure, which may not benefit very much from the relatively small NUS 8-camera dataset.
Figure 4: Examples of our results for different angular errors. From left to right: the original RAW images, images corrected with ground-truth, images corrected with predictions by DCC, and chrominance.

with $\gamma = 1/2.2$ is applied to the restored RGB images for display purpose.

5. Conclusion

In this paper, we have developed a learning-based color constancy algorithm called DCC, which has two important features. First of all, an efficient edge augmentation is used to well capture the spatial and gradient information of edges in an image. Secondly, CBAM is employed to reduce the ambiguity in the edge augmentation and feature extraction. We have demonstrated from experiments that the proposed DCC algorithm achieves the state-of-the-art illuminant estimation performance on the reprocessed Color Checker dataset.

We would like to point out a technical limitation of our algorithm. Due to the deep structure, our network may not exhibit promising performance on small datasets. Designing a lighter structure can significantly improve the generalization of our algorithm and at the same time shorten the training time. Furthermore, enhancing the physical portability of our algorithm to devices of limited computational resources can also be of great importance. Our future work will be directed towards these goals.
References

[1] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous systems. Software Availble: https://www.tensorflow.org, 2019.

[2] T. Alexandrov, S. Meding, D. Trede, J. H. Kobarg, B. Baluff, A. Walch, H. Thiele, and P. Maass. Super-resolution segmentation of imaging mass spectrometry data: Solving the issue of low lateral resolution. Journal of Proteomics, 75(1):237–245, 2011.

[3] Kobus Barnard. Improvements to gamut mapping colour constancy algorithms. In Proceedings of the European Conference on Computer Vision, pages 390–403. Springer, 2000.

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

[5] Jonathan T. Barron. Convolutional color constancy. In Proceedings of the IEEE International Conference on Computer Vision, pages 379–387. IEEE, 2015.

[6] Jonathan T. Barron and Yun-Ta Tsai. Fast fourier color constancy. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 886–894. IEEE, 2017.

[7] Simone Bianco and Claudio Cusano. Quasi-unsupervised color constancy. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 12212–12221. IEEE, 2019.

[8] Simone Bianco, Claudio Cusano, and Raimondo Schettini. Color constancy using cnns. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pages 81–89. IEEE, 2015.

[9] Simone Bianco, Claudio Cusano, and Raimondo Schettini. Single and multiple illuminant estimation using convolutional neural networks. IEEE Transactions on Image Processing, 26(9):4347–4362, 2017.

[10] David H. Brainard and Brian A. Wandell. Analysis of the retina theory of color vision. Journal of the Optical Society of America A, 10(10):1651–1661, 1986.

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

[12] Marco Buzzelli, Joost van de Weijer, and Raimondo Schettini. Learning illuminant estimation from object recognition. In Proceedings of the 25th IEEE International Conference on Image Processing, pages 3234–3238. IEEE, 2018.

[13] Vlad C. Cardei, Brian Funt, and Kobus Barnard. Estimating the scene illumination chromaticity by using a neural network. Journal of the Optical Society of America A, 19(12):2374–2386, 2002.

[14] Ayan Chakrabarti, Keigo Hirakawa, and Todd Zickler. Color constancy with spatio-spectral statistics. IEEE Transactions on Pattern Analysis and Machine Intelligence, 34(8):1509–1519, 2011.

[15] 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.

[16] Dongliang Cheng, Brian Price, Scott Cohen, and Michael S. Brown. Effective learning-based illuminant estimation using simple features. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1000–1008. IEEE, 2015.

[17] Graham D. Finlayson. Corrected-moment illuminant estimation. In Proceedings of the IEEE International Conference on Computer Vision, pages 1904–1911. IEEE, 2013.

[18] Graham D. Finlayson and Steven D. Hordley. Color constancy at a pixel. Journal of the Optical Society of America A, 18(2):253–264, 2001.

[19] Graham D. Finlayson, Steven D. Hordley, and Paul M. Hubel. Color by correlation: A simple, unifying framework for color constancy. IEEE Transactions on Pattern Analysis and Machine Intelligence, 23(11):1209–1221, 2001.

[20] Graham D. Finlayson and Elisabetta Trezzi. Shades of gray and colour constancy. In Proceedings of the Color and Imaging Conference, volume 2004, pages 37–41. Society for Imaging Science and Technology, 2004.

[21] Brian Funt and Weihua Xiong. Estimating illumination chromaticity via support vector regression. In Proceedings of the Color and Imaging Conference, volume 2004, pages 47–52. Society for Imaging Science and Technology, 2004.

[22] Peter Vincent Gehler, Carsten Rother, Andrew Blake, Tom Minka, and Toby Sharp. Bayesian color constancy revisited. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8. IEEE, 2008.

[23] Arjan Gijsenij and Theo Gevers. Color constancy using natural image statistics and scene semantics. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(4):687–698, 2010.

[24] 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.

[25] Bahadir K. Gunturk, Aziz Umit Batur, Yucel Altunbasak, Monson H. Hayes, and Russell M. Mersereau. Eigenspace-domain super-resolution for face recognition. IEEE Transactions on Image Processing, 12(5):597–606, 2003.

[26] Steven D. Hordley and Graham D. Finlayson. Re-evaluating colour constancy algorithms. In Proceedings of the 17th International Conference on Pattern Recognition, volume 1, pages 76–79. IEEE, 2004.

[27] Yuanming Hu, Baoyuan Wang, and Stephen Lin. Fc4: Fully convolutional color constancy with confidence-weighted pooling. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4085–4094. IEEE, 2017.

[28] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 4700–4708, 2017.

[29] Hamid Reza Vaezi Joze and Mark S. Drew. Exemplar-based color constancy and multiple illumination. IEEE Transactions on Pattern Analysis and Machine Intelligence, 36(5):860–873, 2013.
[30] Hamid Reza Vaezi Joze, Mark S. Drew, Graham D. Finlayson, and Perla Aurora Troncoso Rey. The role of bright pixels in illumination estimation. In Proceedings of the Color and Imaging Conference, volume 2012, pages 41–46. Society for Imaging Science and Technology, 2012.

[31] Nick Kanopoulos, Nagesh Vasanthavada, and Robert L. Baker. Design of an image edge detection filter using the sobel operator. IEEE Journal of Solid-state Circuits, 23(2):358–367, 1988.

[32] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. ArXiv:1412.6980, 2014.

[33] Yanlin Qian, Joni-Kristian Kamarainen, Jarno Nikkanen, and Jiri Matas. On finding gray pixels. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 8062–8070. IEEE, 2019.

[34] Abdul Muntakim Rafi, Uday Kamal, Rakibul Hoque, Abid Abrar, Sowmitra Das, Robert Laganière, and Md Kamrul Hasan. Application of densenet in camera model identification and post-processing detection. ArXiv:1809.00576, 2018.

[35] Steven A. Shafer. Using color to separate reflection components. Color Research and Application, 10(4):210–218, 1985.

[36] Lilong Shi and Brian Funt. Re-processed version of the gehler color constancy dataset of 568 images. Online Available: [http://www.cs.sfu.ca/~colour/data/](http://www.cs.sfu.ca/~colour/data/), 2000.

[37] Wu Shi, Chen Change Loy, and Xiaoou Tang. Deep specialized network for illuminant estimation. In Proceedings of the European Conference on Computer Vision, pages 371–387. Springer, 2016.

[38] Irwin Sobel and Gary Feldman. A $3 \times 3$ isotropic gradient operator for image processing. A Talk at the Stanford Artificial Project in, pages 271–272, 1968.

[39] Joost Van De Weijer, Theo Gevers, and Arjan Gijsenij. Edge-based color constancy. IEEE Transactions on Image Processing, 16(9):2207–2214, 2007.

[40] Rebecca O. Vincent and Olusegun Folorunso. A descriptive algorithm for sobel image edge detection. In Proceedings of Informing Science & IT Education Conference, volume 40, pages 97–107. Informing Science Institute California, 2009.

[41] Johannes Von Kries. Influence of adaptation on the effects produced by luminous stimuli. Handbuch der Physiologie des Menschen, 3:109–282, 1905.

[42] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In Proceedings of the European Conference on Computer Vision, pages 3–19, 2018.