Convolutional Color Constancy

Jonathan T. Barron
barron@google.com

Abstract

Color constancy is the problem of inferring the color of the light that illuminated a scene, usually so that the illumination color can be removed. Because this problem is underconstrained, it is often solved by modeling the statistical regularities of the colors of natural objects and illumination. In contrast, in this paper we reformulate the problem of color constancy as a 2D spatial localization task in a log-chrominance space, thereby allowing us to apply techniques from object detection and structured prediction to the color constancy problem. By directly learning how to discriminate between correctly white-balanced images and poorly white-balanced images, our model is able to improve performance on standard benchmarks by nearly 40%.

1. Intro

The color of a pixel in an image can be described as a product of two quantities: reflectance (the color of the paint of the surfaces in the scene) and illumination (the color of the light striking the surfaces in the scene). When a person stands in a room lit by a colorful light they unconsciously "discount the illuminant", in the words of Helmholtz [27], and perceive the objects in the room as though they were illuminated by a neutral, white light. Endowing a computer with the same ability is difficult, as this problem is fundamentally underconstrained — given a yellow pixel, how can one discern if it is a white object under a yellow illuminant, or a yellow object under a white illuminant? The most general characterization of this problem is the "intrinsic image" problem [6], but the specific problem of inferring and correcting the color of the illumination of an image is commonly referred to as "color constancy" or "white balance". A visualization of this problem can be seen in Figure 1.

Color constancy is a well studied in both vision science and computer vision, as it relates to the academic study of human perception as well as practical problems such as designing an object recognition algorithm or a camera. Nearly all algorithms for this task work by assuming some regularity in the colors of natural objects viewed under a white light. The simplest such algorihm is "gray world", which assumes that the illuminant color is the average color of all image pixels, thereby implicitly assuming that object reflectances are, on average, gray [12]. This simple idea can be generalized to modeling gradient information or using generalized norms instead of a simple arithmetic mean [4, 36], modeling the spatial distribution of colors with a filter bank [13], modeling the distribution of color histograms [21], or implicitly reasoning about the moments of colors using PCA [14]. Other models assume that the colors of natural objects lie with some gamut [3, 23]. Most of these models can be thought of as statistical, as they either assume some distribution of colors or they learn some distribution of colors from training data. This connection to learning and statistics is sometimes made more explicit, often in a

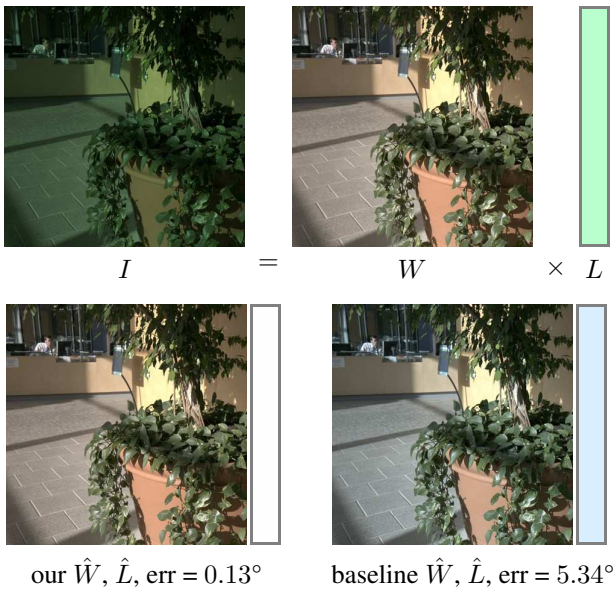

Figure 1: Here we demonstrate the color constancy problem: the input image I (taken from [24, 33]) looks green, and we want to recover a white-balanced image W and illumination L which reproduces I. Below we have our model's solution and error for this image compared to a state of the art baseline [19] (recovered illuminations are rendered with respect to ground-truth, so white is correct). More results can be seen in the supplement.

Bayesian framework [10, 24].

One thing that these statistical or learning-based models have in common is that they are all *generative* models of natural colors — a model is learned from (or assumed of) white-balanced images, and then that model is used to white-balance new images. In this paper, we will operate under the assumption that white-balancing is a *discriminative* task. That is, instead of training a generative model to assign high likelihoods to white-balanced images under the assumption that such a model will perform well at white-balancing, we will explicitly train a model to distinguish between white-balanced images and non-white-balanced images. This use of discriminative machine learning appears to be largely unexplored in context of color constancy, though similar tools have been used to augment generative color constancy models with face detection [9] or scene classification [25] information. The most related technique to our own is probably that of Finlayson [19] in which a simple "correction" to a generalized gray-world algorithm is learned using iterative least-squares, producing state-of-the-art results compared to prior art.

Let us contrast the study of color constancy algorithms with the seemingly disparate problem of object detection. Object detection has seen a tremendous amount of growth and success in the last 20 years owing in large part to standardized challenges [16] and effective machine learning techniques, with techniques evolving from simple sliding window classifiers [15, 32, 37] to sophisticated deformable models [17] or segmentation-based techniques [28]. The vast majority of this work operates under the assumption that object detection should be phrased as the problem of learning a discriminative classifier which predicts whether an image patch is, in the case of face detection for example, a "face" or a "nonface". It is common knowledge that reasoning about the "nonface" background class is nearly as important as reasoning about the object category of interest, as evidenced by the importance of "mining for hard negatives" [32] when learning an effective object detection system. In contrast, training a generative model of an object category for detection is widely considered to be ineffective, with some unusual exceptions [29]. At first glance, it may seem that all of this has little to teach us about color constancy, as most established color constancy algorithms are fundamentally incompatible with the discriminative learning techniques used in object detection. But if the color constancy problem could be reduced to the problem of localizing a template in some n-dimensional space, then presumably the lessons learned from object detection techniques could be used to design an effective color constancy algorithm.

In this paper we present CCC ("Convolutional Color Constancy"), a novel color constancy algorithm that has been designed under the assumption that color constancy is a discriminative learning task. Our algorithm is based around the observation that scaling the color channels of an image induces a translation in the log-chromaticity histogram of that image. This observation allows us to frame the color constancy problem as a discriminative learning problem, using tools similar to convolutional neural networks [31] and structured prediction [34]. Effectively, we are able to reframe the problem of color constancy as the problem of localizing a template in some two-dimensional space, thereby allowing us to borrow techniques from the well-understood problem of object detection. By discriminatively training a color constancy algorithm in this way, we produce state-of-the-art results and reduce error rates on standard benchmarks by nearly 40%.

Our paper will proceed as follows: In Section 2 we will demonstrate the relationship between image tinting and log-chrominance translation. In Section 3 we will describe how to learn a discriminative color constancy algorithm in our newly-defined log-chrominance space. In Section 4 we will explain how to perform efficient filtering in our log-chrominance space, which is required for fast training and evaluation. In Section 5 we will show how to generalize our model from individual pixel colors to spatial phenomena like edges and patches. In Section 6 we will evaluate our model on two different color constancy tasks, and in Section 7 we will conclude.

2. Image Formation

Consider a photometric linear image I taken from a camera, in which black-level correction has been performed and in which no pixel values have saturated. According to a simplified model of image formation, each RGB pixel value in I is the product of the "true" white-balanced RGB value W for that pixel and the RGB illumination L shared by all pixels, as shown in Figure 1.

$$I = W \times L \tag{1}$$

This is a severe oversimplification of the larger "intrinsic image" problem, as it ignores shading, reflectance properties, spatially-varying illumination, etc. This model also assumes that color constancy can be achieved by simply modifying the gains of each channel individually (the Von Kries coefficient law [38]) which, though certainly an approximation [11], is an effective and widespread assumption. Our goal is, given I, to estimate L and then produce $W = I/L$.

Let us define two measures of chrominance u and v from the RGB values of I and W:

$$\begin{aligned} I_u &= \log(I_g/I_r) & I_v &= \log(I_g/I_b) \\ W_u &= \log(W_g/W_r) & W_v &= \log(W_g/W_b) \end{aligned} \tag{2}$$

Additionally, it is convenient to define a luminance measure

y for I:

$$I_y = \sqrt{I_r^2 + I_g^2 + I_b^2} \tag{3}$$

Given that we do not care about the absolute scaling of W, the problem of estimating L simplifies further to just estimating the "chrominance" of L, which can just be represented as two numbers:

$$L_u = \log(L_g/L_r) \qquad L_v = \log(L_g/L_b) \tag{4}$$

Notice that by our definitions and by the properties of logarithms, we can rewrite the problem formulation in Equation 1 in this log-chrominance space:

$$W_u = I_u - L_u \qquad W_v = I_v - L_v \tag{5}$$

So, our problem reduces to recovering just two quantities: (L_u, L_v). Because of the absolute scale ambiguity, the inverse mapping from RGB to UV is undefined. So after recovering (L_u, L_v), we make the additional assumption that L is unit-norm which allows us to recover (L_r, L_g, L_b):

$$L_r = \frac{\exp(-L_u)}{z} \qquad L_g = \frac{1}{z} \qquad L_b = \frac{\exp(-L_v)}{z}$$
$$z = \sqrt{\exp(-L_u)^2 + \exp(-L_v)^2 + 1} \tag{6}$$

This log-chrominance formulation has several advantages over the RGB formulation. We have 2 unknowns instead of 3, and we just have a simple linear constraint relating W and I instead of a multiplicative constraint. Though they may seem unimportant, these properties are required to reformulate our problem as a 2D spatial localization task.

3. Learning

Let us consider an input image I and its ground-truth illumination L. We will construct a histogram M from I, where $M(u, v)$ is the the number of pixels in I whose chrominance is near (u, v), with histogram counts weighted by each pixel's luminance:

$$M(u,v) = \sum_i I_y^{(i)} \left[\left|I_u^{(i)} - u\right| \leq \frac{\epsilon}{2} \wedge \left|I_v^{(i)} - v\right| \leq \frac{\epsilon}{2}\right] \tag{7}$$

Where the square brackets are an indicator function and ϵ is the bin-width of the histogram (in all experiments, $\epsilon = 0.025$ and histograms have 256 bins). To produce our final histogram features N take the square root of the L1-normalized histogram counts, which generally improves the effectiveness of histogram features [2].

$$N(u,v) = \sqrt{\frac{M(u,v)}{\sum_{u',v'} M(u',v')}} \tag{8}$$

Any normalization or transformation is allowed at this step as long as the same operation is applied to the entire histogram, though at other stages in the algorithm care must be taken to preserve translational invariance.

(a) Input Image (b) True Image (c) Tinted Image

Figure 2: Some images and their log-chrominance histograms (with an axis overlayed for easier visualization, horizontal = u, vertical = v). The images are the same except for "tints" — scaling of red and blue. Tinting an image affects the image's histogram only by a translation in log-chrominance space. This observation enables our convolutional approach to color correction, in which our algorithm learns to localize a histogram in this 2D space.

In Figure 2 we show three tinted versions of the same image with each image's chrominance histogram. Note that each histogram is a translated version of the other histograms (ignoring sampling artifacts) and that the shape of the histogram does not change. This is a consequence of our definitions of u and v: scaling a pixel's RGB value is equivalent to translating a pixel's log-chrominance, as was noted in [20]. This equivalence between image-tinting and histogram-shifting enables the rest of our algorithm.

Our algorithm works by considering all possible tints of an image, scoring each tinted image, and then returning the highest-scoring tint as the estimated illumination of the input image. This may sound like an expensive proposition as it requires a brute-force search over all possible tints, where some scoring function is applied at each tint. However, provided that the scoring function is a linear combination of histogram bins, this brute-force search is actually just the convolution of N with some filter F, and there are many ways that convolution can be made efficient. This gives us a sketch of our algorithm: we will construct a histogram N from the input image I, convolve that histogram with some filter F, and then use the highest-scoring illumination $\hat{L}$ to produce $\hat{W} = I/\hat{L}$. More formally:

$$(\hat{L}_u, \hat{L}_v) = \arg\max_{u,v} (N * F) \tag{9}$$

A visualization of this procedure (actually, a slightly more complicated version which will be explained later) can be seen in Figure 7. Now we require a way to learn a filter

F from training data such that this convolution produces accurate output.

To learn F we use a model similar to multinomial logistic regression or structured prediction, in a convolutional framework. Formally, our optimization problem is:

$$\min_F \lambda \sum_{u,v} F(u,v)^2 + \sum_{i,u,v} P(u,v)\, C\Big(u, v, L_u^{(i)}, L_v^{(i)}\Big)$$
$$P(u,v) = \frac{\exp\big((N^{(i)} * F)(u,v)\big)}{\sum_{u',v'} \exp\big((N^{(i)} * F)(u',v')\big)} \quad (10)$$

Where F is the filter whose weights we learn, $\{N^{(i)}\}$ and $\{L^{(i)}\}$ are our training-set chrominance histograms and ground-truth illuminations, respectively, and $(N^{(i)} * F)(u,v)$ is the convolution of $N^{(i)}$ and F indexed at location (u,v). For convenience we define $P(u,v)$ which is a softmax probability for each (u,v) bin in our histogram as a function of $N^{(i)} * F$. We regularize our filter weights by minimizing the sum of squares of the elements of F, moderated by some hyperparameter λ. At a high level, minimizing our loss finds an F such that $N^{(i)} * F$ is larger at $(L_u^{(i)}, L_v^{(i)})$ than it is elsewhere, where $C(u,v,u^*,v^*)$ defines the loss incurred at mis-estimated illuminants:

$$C\left(u,v,u^*,v^*\right) = \arccos\left(\frac{\langle \ell, \ell^* \rangle}{\|\ell\|\,\|\ell^*\|}\right)$$
$$\ell = [\exp(-u),\ 1,\ \exp(-v)]^{\mathrm{T}}$$
$$\ell^* = [\exp(-u^*), 1, \exp(-v^*)]^{\mathrm{T}} \quad (11)$$

C measures the angle between the illuminations defined by (u,v) and (u^*,v^*), which is the error by which color-constancy algorithms are commonly evaluated. Visualizations of C can be seen in Figure 3. During training we initialize F to all zeros (initialization does not appear to affect accuracy) and we minimize Eq. 10 first using a variant of stochastic gradient descent (detailed in supplement) followed by batch L-BFGS until convergence. Using both optimization techniques produces lower losses and test-set error rates than using only SGD, but more quickly than only using batch L-BFGS. Though our loss function is non-convex, optimization appears to work well and our learned model performs better than other models trained with various convex approximations to our loss function.

Our problem resembles multinomial logistic regression, but where every (u,v) has a variable loss C measuring the cost of each possible (u,v) chrominance with respect to some ground-truth chrominance (u^*,v^*). The use of a softmax makes our model resemble a classification problem, and the use of a variable cost makes our model resemble structured prediction. We experimented with simply minimizing the cross-entropy of $P(u,v)$ with respect to a delta function at (u^*,v^*), and with using maximum-margin structured prediction [34] with margin rescaling and slack

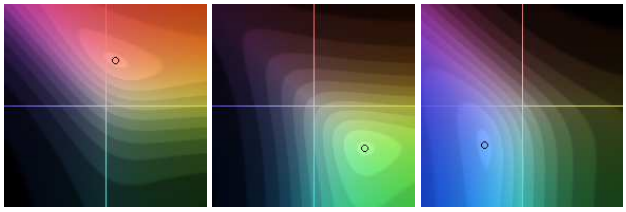

Figure 3: Visualizations of the cost function used during training $C\left(u,v,u^*,v^*\right)$ as a function of the proposed illumination color (u,v), with each plot showing a different choice of the ground-truth illumination color (u^*,v^*) (circled). Darker luminance means higher cost. These cost functions are used during training to encourage our learned filter to "fire" strongly at the true illuminant (u^*,v^*) when convolved with the input histogram.

rescaling, but found that our proposed approach produced more accurate results on the test set. We also experimented with learning a "deep" set of filters instead of a single filter F, thereby resulting in a convolutional neural network [31], but we found the amount of training data in our datasets insufficient to prevent overfitting.

A core property of our approach is that our model is trained *discriminatively*. Our structured-prediction approach means that F is learned directly in accordance with the criteria we care about — how accurately it identifies each illumination color in the training set. This is very different from the majority of color constancy algorithms which either learn or analytically construct generative models of the distributions of colors in natural images viewed under white light. To demonstrate the importance of discriminative training, we will evaluate against a generatively-trained version of our model which learns a model to maximize the likelihood of colors in natural images, while not considering that this generative model will be used for a discriminative task. Our generative model learns our filter F according to the following optimization problem:

$$\max_F \sum_i \sum_{u,v} \Big(\log\left(P(u,v)\right) N^{(i)}(u,v)\Big)$$
$$P(u,v) = \frac{\exp\big((\delta^{(i)} * F)(u,v)\big)}{\sum_{u',v'} \exp\big((\delta^{(i)} * F)(u',v')\big)}$$
$$\delta^{(i)} = \left[\left(\left|u - L_u^{(i)}\right| \le \epsilon/2\right) \wedge \left(\left|v - L_v^{(i)}\right| \le \epsilon/2\right)\right] \quad (12)$$

Minimizing this loss produces a filter F such that, when F is convolved with a delta function located at the illuminant color's chrominance, the categorical distribution produced by exponentiating that filter output maximizes the likelihood of the training set chroma histograms $\{N^{(i)}\}$. We do not regularize F, as it does not improve performance when generative training is used.

discriminative F generative F

Figure 4: Learned filters on the same training data, with the left filter learned discriminatively and the right filter learned generatively. The generative model just learns a simple "gray-world" like filter, while the discriminative model learns to do things like upweight blues that resemble the sky and downweight pale greens that resemble badly white-balanced images. One can think of the discriminatively learned filter as a histogram of colors in well white-balanced images *minus* a histogram of colors in poorly white-balanced images

A visualization of filters learned discriminatively and generatively on the same data can be seen in Figure 4. We see that discriminative training learns a much richer and more elaborate model than the generative model. This is because our discriminative training does not just learn what white-balanced images look like — it learns how to distinguish between white-balanced images and improperly white-balanced images. In Section 6 we will show that discriminative training substantially improves model accuracy.

4. Efficient Filtering

Though our algorithm revolves around a linear filter F with which we will convolve our chroma histograms, the specific parametrization of F affects the accuracy and speed of our model. For example, a filter the size of the input histogram would be likely to overfit and would be expensive to evaluate. We found that accurate filters for our task tend to have a log-polar or "retinotopic" structure, in which the filter contains a large amount of high-frequency variation near the center of the filter but only contains low-frequency variation far from the center. Intuitively, this makes sense: when localizing the illumination color of an image, the model should pay close attention to chroma variation near the predicted white point, while only broadly considering chroma variation far from the predicted white point.

With the goal of a fast retina-like filter, we chose to use the "pyramid filtering" technique of [5] for our histogram convolution. Pyramid filtering works by first constructing a Gaussian pyramid of the input signal (in this case, we construct a 7-level pyramid from $N(u, v)$ using bilinear downsampling), then filtering each scale with a small filter (we used 5×5 filters), and then collapsing the filtered pyramid down into an image (using bilinear upsampling). When collapsing the pyramid we found it necessary to apply a $[1, 2, 1]$ blur before each upsample operation to reduce sampling artifacts. This filter has several desirable properties: it is efficient to compute, there are few free parameters so optimization and regularization are easy, and the filter can describe fine detail in the center while modeling coarse context far from the center. We regularize this filter by simply minimizing the squared 2-norm of the filter coefficients at each scale, all modulated by a single hyperparameter λ, as in Eq. 10 (this is actually a slight departure from Eq. 10 as the regularization is now in a linearly transformed space). A visualization of pyramid filtering can be seen in Figure 5.

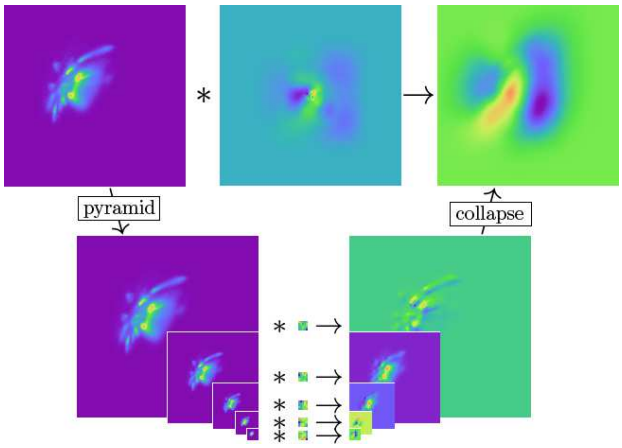

Figure 5: Here we visualize the "pyramid filter" [5] used to score chroma histograms. Above we show naive convolution of a histogram (top left) with a retina-like filter (top middle), while below we evaluate that same filter more efficiently by constructing a pyramid from the histogram, convolving each scale of the pyramid with a small filter, and then collapsing the filtered histogram. By using the latter filtering approach we simplify regularization during training and improve speed during testing.

As described in [5] pyramid filtering is equivalent to, for every pixel, computing a feature with a log-polar sampling pattern and then classifying that feature with a linear classifier. This sort of feature resembles standard features used in computer vision, like shape context [7], geometric blur [8], FREAK features [1], DAISY [35], etc. However, the pyramid approximation requires that the sampling pattern of the feature be rectangular instead of polar, that the scales of the feature be discretized to powers of 2, and that the sampling patterns of the feature at each scale overlap. This difference makes it tractable to compute and classify these features densely at every pixel in the image, which in turn

allows us to estimate the illuminant color very precisely.

5. Generalization

The previously described algorithm can estimate the illumination L from an image I by filtering a histogram N constructed from the chroma values of the pixels in I. Effectively, this model is a sophisticated kind of "gray world" algorithm, in that all spatial information is ignored and the image is treated like a "bag" of pixels. However, well-performing color constancy algorithms generally use additional sources of information, such as the color of edges [3, 19, 36] or spatial neighborhoods [13]. To that end, we present an extension of our algorithm in which instead of constructing and classifying a single histogram N from a single image I, we filter a set of histograms $\{N_j\}$ from a set of "augmented" images $\{I'_j\}$, and sum the filtered responses before computing softmax probabilities. These augmented images will reflect edge and spatial statistics of the image I, thereby enabling our model reason about multiple sources of chroma information beyond individual pixel chroma.

Naively one might attempt to construct these augmented images $\{I'_j\}$ by simply applying common image processing operations to I, such as applying a filter bank, median filters, morphological operations, etc. But remember from Section 3 that the image from which we construct chroma histograms must exactly map scaling to the channels of the input image to shifts in chroma histogram space. This means that our augmented images must also map a per-channel scaling to the same shift in histogram space, limiting the set of possible augmented images that we can use.

For our color-scaling/histogram-shifting requirement to be met, our augmented-image mappings must preserve scalar multiplication: a scaled-then-filtered version of a channel in the input image I must be equal to a filtered-then-scaled version of that channel. This problem is alluded to in [19], in which the authors limit themselves to "color moments which scale with intensity". Additionally, the output of the mappings must be non-negative as we will need to compute the logarithm of the output of each mapping (the input is assumed to be non-negative). Here are three mappings which satisfy our criteria:

$$\begin{aligned} f(I, \mathit{filt}) &= \max(0, I * \mathit{filt}) \\ g(I, \rho, w) &= \mathrm{blur}(I^\rho, w)^{1/\rho} \\ h(I, \rho, w) &= (\mathrm{blur}(I^\rho, w) - \mathrm{blur}(I, w)^\rho)^{1/\rho} \end{aligned} \tag{13}$$

Where $\mathrm{blur}(\cdot, w)$ is a box filter of width w. $f(\cdot, \mathit{filt})$ convolves each channel of the image with some filter filt and then clamps the filtered value to be at least 0. $g(\cdot, p, w)$ computes a local norm of pixel values in I such that $g(\cdot, 1, w)$ is a blur, $g(\cdot, \infty, w)$ is a "max" filter, and $g(\cdot, -\infty, w)$ is a "min" filter. $h(\cdot)$ computes a kind of normalized moment of pixel values, where $h(\cdot, 2, w)$ is the local standard deviation of pixel values — an unoriented edge/texture detector. These operations all preserve scalar multiplication:

$$\begin{aligned} f(\alpha I, \mathit{filt}) &= \alpha f(I, \mathit{filt}) \\ g(\alpha I, \rho, w) &= \alpha g(I, \rho, w) \\ h(\alpha I, \rho, w) &= \alpha h(I, \rho, w) \end{aligned} \tag{14}$$

$I'_1 = I$ I'_2 I'_3 I'_4

Figure 6: Though our model can take the pixel values of the input image I as its sole input, performance can be improved by using a set of "augmented" images $\{I'\}$. Our extended model uses three augmented images which capture local spatial information (texture, highlights, and edges, respectively) in addition to the input image.

In our extended model we use four augmented images: the input image I itself, a "sharpened" and rectified I, a "soft" max-filtered I, and a standard-deviation-filtered I.

$$\begin{aligned} I'_1 &= I \\ I'_2 &= \max\left(0, I * \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}\right) \\ I'_3 &= \mathrm{blur}(I^4, 11)^{1/4} \\ I'_4 &= \sqrt{\mathrm{blur}(I^2, 3) - \mathrm{blur}(I, 3)^2} \end{aligned} \tag{15}$$

Other similar channels or compositions of these channels could be used as well, though we use a small number of simple channels here for the sake of speed and to prevent overfitting. See Figure 6 for visualizations of the information captured by each of these channels. During training we simply learn 4 pyramid filters instead of 1 and sum the individual filter responses before computing the softmax probabilities in Eq. 10.

Now that all of our model components have been defined, we can visualize inference in our final model in Figure 7.

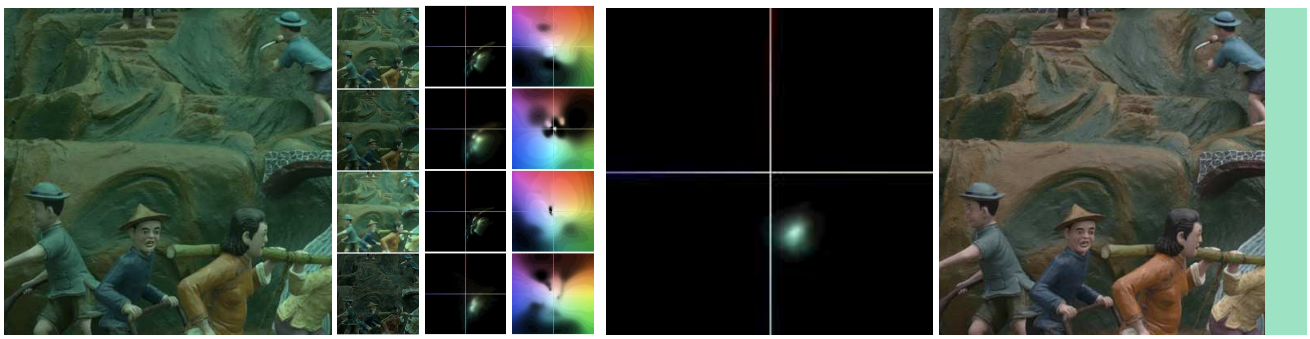

Figure 7: An overview of inference in our model for a single image. An input image I is transformed into a set of scale-preserving augmented images $\{I'_j\}$ which highlight different aspects of the image (edges, patches, etc). The set of augmented images is turned into a set of chroma histograms $\{N_j\}$, for which we have learned a set of weights in the form of pyramid filters $\{F_j\}$. The histograms are convolved with the filters and then summed, giving us a score for all bins in our chroma histogram. The highest-scoring bin is assumed to be the color of the illuminant $\hat{L}$, and the output image $\hat{W}$ is produced by dividing the input image by that illuminant.

6. Results

We evaluate our algorithm on two datasets: the Color Checker Dataset [24] reprocessed by Shi and Funt [33], and the dataset from Cheng et al. [14]. The Color Checker dataset is widely used and is reasonably large — 568 images from a single camera. The dataset from Cheng et al. is larger, with 1736 images taken from 8 different cameras, but the same scene is imaged multiple times by each of the 8 cameras. As is standard, we evaluate using three-fold cross-validation, computing the angle in degrees between our estimated illumination $\hat{L}$ and the true illumination L^* for each image. We report several statistics of these errors: the mean, the median, the tri-mean, the means of the errors in the lowest-error 25% of the data and the highest-error 25% of the data, and for the Color Checker Dataset the 95th percentile. Some baseline results on the Color Checker Dataset were taken from past papers, thereby resulting in some missing error metrics for some algorithms. For the Cheng et al. dataset we also report an average error, which is the geometric mean of the other error statistics.

Cheng et al. ran 8 different experiments with their 8 different cameras, which makes tersely summarizing performance difficult. To that end, we report the geometric mean of each error metric for each algorithm across all cameras. We computed results for our own algorithm identically: we learn a model for each camera independently, compute errors for each camera, and then report the geometric mean across all cameras.

Our results can be seen in Tables 1 and 2. On the Color Checker Dataset we see a 30% and 39% reduction in error (mean and median, respectively) from the state-of-the-art ("Corrected-Moment" [19]), and on the dataset of Cheng et al. we see a 22% reduction in average error from the state-of-the-art (Cheng et al.). This improvement is fairly consistent across different choices of error metrics. The increased improvement on the Color Checker Dataset is likely due to the larger size of the Color Checker Dataset ($\sim$379 training images as opposed to $\sim$ 144, for three-fold cross validation), which likely favors our learning-based approach. An example of our performance with respect to the state of the art can be seen in Figure 1 and in the supplement.

In our experiments we evaluated several different versions of our algorithm ("CCC"), indicated by the name of each model. Models labeled "gen" are trained in a generative fashion (Eq. 12), while "disc" models are trained discriminatively (Eq. 10). Models labeled "simp" use our simple feature set (just the input image) while "ext" models use the four augmented images from Section 5. Our results show that discriminative training is superior to generative training by a large margin ($30-40\%$ improvement), and that using our extended model produces better results than our simple model ($10-20\%$ improvement). Our generatively-trained models perform similarly to some past techniques which were also trained in a generative fashion, suggesting that the use of discriminative training is the driving force behind our algorithm's performance.

Though most of our baseline results were taken from past papers, to ensure a thorough and fair evaluation we obtained the code for the best-performing technique on the Color Checker dataset ("Corrected-Moment" [19]) and ran it ourselves on the Cheng et al. dataset. We also ran this code on the Color Checker dataset and reported our reproduced results, which differ slightly from those reported in [19] apparently due to different parameter settings or inconsistencies between the provided code and the paper [18]. Results

for the corrected moment algorithm produced by ourselves are indicated with asterisks in Tables 1 and 2.

Evaluating our trained model is reasonably fast. With our unoptimized Matlab implementation running on a 2012 HP Z420 workstation, for each image it takes about 1.2 seconds per megapixel to construct our augmented images and produce normalized log-chrominance histograms from them, and about 20 milliseconds to pyramid-filter those histograms and extract the argmax.

7. Conclusion

We have presented CCC, a novel learning-based algorithm for color constancy. Our technique builds on the observation that the per-channel scaling in an image caused by the color of the illumination produces a translation in the space of log-chroma histograms. This observation lets us leverage ideas from object detection and structured prediction to discriminatively train a convolutional classifier to perform well at white balancing, as opposed to the majority of prior work which uses generative training. Our algorithm is made more efficient by using a pyramid-based approach to image filtering, and is made more accurate by augmenting the input to our algorithm with variants of the input image that capture different kinds of spatial information. Our technique produces state-of-the-art performance on the two largest color constancy datasets, beating the best-performing techniques by $20\% - 40\%$ on various error metrics. Our experiments suggest that color constancy algorithms may benefit from much larger datasets than are currently used, as has been the case for object detection and recognition. This newly-established connection with object detection suggests that color constancy may be a fruitful domain for researchers to apply new object detection techniques. Furthermore, our results show that many of the lessons learned from discriminative machine learning are more relevant to color constancy than has been previously thought, and suggests that other core low-level vision and imaging tasks may benefit from a similar reevaluation.

References

[1] A. Alahi, R. Ortiz, and P. Vandergheynst. FREAK: Fast Retina Keypoint. *CVPR*, 2012.

[2] R. Arandjelovic and A. Zisserman. Three things everyone should know to improve object retrieval. *CVPR*, 2012.

[3] K. Barnard. Improvements to gamut mapping colour constancy algorithms. *ECCV*, 2000.

[4] K. Barnard, L. Martin, A. Coath, and B. Funt. A comparison of computational color constancy algorithms — part 2: Experiments with image data. *TIP*, 2002.

[5] J. T. Barron, P. Arbeláez, S. V. E. Keränen, M. D. Biggin, D. W. Knowles, and J. Malik. Volumetric semantic segmentation using pyramid context features. *ICCV*, 2013.

Algorithm	Mean	Med.	Tri.	Best 25%	Worst 25%	95% Quant.
White-Patch [11]	7.55	5.68	6.35	1.45	16.12	-
Edge-based Gamut [3]	6.52	5.04	5.43	1.90	13.58	-
Gray-World [12]	6.36	6.28	6.28	2.33	10.58	11.3
1st-order Gray-Edge [36]	5.33	4.52	4.73	1.86	10.03	11.0
2nd-order Gray-Edge [36]	5.13	4.44	4.62	2.11	9.26	-
Shades-of-Gray [22]	4.93	4.01	4.23	1.14	10.20	11.9
Bayesian [24]	4.82	3.46	3.88	1.26	10.49	-
General Gray-World [4]	4.66	3.48	3.81	1.00	10.09	-
Intersection-based Gamut [3]	4.20	2.39	2.93	0.51	10.70	-
Pixel-based Gamut [3]	4.20	2.33	2.91	0.50	10.72	14.1
Natural Image Statistics [25]	4.19	3.13	3.45	1.00	9.22	11.7
Bright Pixels [30]	3.98	2.61	-	-	-	-
Spatio-spectral (GenPrior) [13]	3.59	2.96	3.10	0.95	7.61	-
Cheng et al. [14]	3.52	2.14	2.47	0.50	8.74	-
Corrected-Moment (19 Color) [19]	3.5	2.6	-	-	-	8.6
Corrected-Moment (19 Edge) [19]	2.8	2.0	-	-	-	6.9
Corrected-Moment* (19 Color) [19]	2.96	2.15	2.37	0.64	6.69	8.23
Corrected-Moment* (19 Edge) [19]	3.12	2.38	2.59	0.90	6.46	7.80
CCC (gen+simp)	3.57	2.62	2.90	1.01	7.74	9.62
CCC (gen+ext)	3.24	2.33	2.61	1.02	6.88	8.42
CCC (disc+simp)	2.48	1.52	1.70	0.37	6.26	8.30
CCC (disc+ext)	1.95	1.22	1.38	0.35	4.76	5.85

Table 1: Performance on the reprocessed [33] Color Checker Dataset [24]. For each metric the best-performing technique is highlighted in red, and the second-best-performing (excluding variants of our technique) is in yellow. Some baseline numbers here were taken from past work [14, 26], which used different error measures, thereby resulting in some missing entries.

Algorithm	Mean	Med.	Tri.	Best 25%	Worst 25%	Avg.
White-Patch [11]	10.62	10.58	10.49	1.86	19.45	8.43
Edge-based Gamut [3]	8.43	7.05	7.37	2.41	16.08	7.01
Pixel-based Gamut [3]	7.70	6.71	6.90	2.51	14.05	6.60
Intersection-based Gamut [3]	7.20	5.96	6.28	2.20	13.61	6.05
Gray-World [12]	4.14	3.20	3.39	0.90	9.00	3.25
Bayesian [24]	3.67	2.73	2.91	0.82	8.21	2.88
Natural Image Statistics [25]	3.71	2.60	2.84	0.79	8.47	2.83
Shades-of-Gray [22]	3.40	2.57	2.73	0.77	7.41	2.67
Spatio-spectral (ML) [13]	3.11	2.49	2.60	0.82	6.59	2.55
General Gray-World [4]	3.21	2.38	2.53	0.71	7.10	2.49
2nd-order Gray-Edge [36]	3.20	2.26	2.44	0.75	7.27	2.49
Bright Pixels [30]	3.17	2.41	2.55	0.69	7.02	2.48
1st-order Gray-Edge [36]	3.20	2.22	2.43	0.72	7.36	2.46
Spatio-spectral (GenPrior) [13]	2.96	2.33	2.47	0.80	6.18	2.43
Corrected-Moment* (19 Edge) [19]	3.03	2.11	2.25	0.68	7.08	2.34
Corrected-Moment* (19 Color) [19]	3.05	1.90	2.13	0.65	7.41	2.26
Cheng et al.[14]	2.92	2.04	2.24	0.62	6.61	2.23
CCC (gen+simp)	3.42	2.66	2.84	0.98	7.09	2.82
CCC (gen+ext)	3.05	2.27	2.49	0.92	6.43	2.52
CCC (disc+simp)	2.61	1.70	1.87	0.52	6.28	1.94
CCC (disc+ext)	2.38	1.48	1.69	0.45	5.85	1.74

Table 2: Performance on the dataset from Cheng et al. [14]. For each metric the best-performing technique is highlighted in red, and the second-best-performing (excluding variants of our technique) is in yellow.

[6] H. G. Barrow and J. M. Tenenbaum. *Recovering Intrinsic Scene Characteristics from Images*. Academic Press, 1978.

[7] S. Belongie, J. Malik, and J. Puzicha. Shape context: A new descriptor for shape matching and object recognition. *NIPS*, 2000.

[8] A. C. Berg and J. Malik. Geometric blur for template matching. *CVPR*, 2001.

[9] S. Bianco and R. Schettini. Color constancy using faces. *CVPR*, 2012.

[10] D. H. Brainard and W. T. Freeman. Bayesian color constancy. *JOSA A*, 1997.

[11] D. H. Brainard and B. A. Wandell. Analysis of the retinex theory of color vision. *JOSA A*, 1986.

[12] G. Buchsbaum. A spatial processor model for object colour perception. *Journal of the Franklin Institute*, 1980.

[13] A. Chakrabarti, K. Hirakawa, and T. Zickler. Color constancy with spatio-spectral statistics. *TPAMI*, 2012.

[14] D. Cheng, D. K. Prasad, and M. S. Brown. Illuminant estimation for color constancy: why spatial-domain methods work and the role of the color distribution. *JOSA A*, 2014.

[15] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *CVPR*, 2005.

[16] M. Everingham, L. Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[17] P. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part based models. *TPAMI*, 2010.

[18] G. D. Finlayson. personal communication.

[19] G. D. Finlayson. Corrected-moment illuminant estimation. *ICCV*, 2013.

[20] G. D. Finlayson and S. D. Hordley. Color constancy at a pixel. *JOSA-A*, 2001.

[21] G. D. Finlayson, S. D. Hordley, and P. M. Hubel. Color by correlation: A simple, unifying framework for color constancy. *TPAMI*, 2001.

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

[23] D. A. Forsyth. A novel algorithm for color constancy. *IJCV*, 1990.

[24] P. Gehler, C. Rother, A. Blake, T. Minka, and T. Sharp. Bayesian color constancy revisited. *CVPR*, 2008.

[25] A. Gijsenij and T. Gevers. Color constancy using natural image statistics and scene semantics. *TPAMI*, 2011.

[26] A. Gijsenij, T. Gevers, and J. van de Weijer. Computational color constancy: Survey and experiments. *TIP*, 2011.

[27] A. L. Gilchrist. *Seeing Black and White*. Oxford University Press, 2006.

[28] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CVPR*, 2014.

[29] B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification. *ECCV*, 2012.

[30] H. R. V. Joze, M. S. Drew, G. D. Finlayson, and P. A. T. Rey. The role of bright pixels in illumination estimation. *Color Imaging Conference*, 2012.

[31] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1989.

[32] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *TPAMI*, 1998.

[33] L. Shi and B. Funt. Re-processed version of the gehler color constancy dataset of 568 images. http://www.cs.sfu.ca/ colour/data/.

[34] B. Taskar, V. Chatalbashev, D. Koller, and C. Guestrin. Learning structured prediction models: A large margin approach. *ICML*, 2005.

[35] E. Tola, V. Lepetit, and P. Fua. DAISY: An Efficient Dense Descriptor Applied to Wide Baseline Stereo. *TPAMI*, 2010.

[36] J. van de Weijer, T. Gevers, and A. Gijsenij. Edge-based color constancy. *TIP*, 2007.

[37] P. Viola and M. Jones. Robust real-time object detection. *IJCV*, 2001.

[38] J. von Kries. Die gesichtsempfindungen. *Handbuch der Physiologie des Menschen*, 1905.