Estimating the scene illumination chromaticity by using a neural network

Vlad C. Cardei

NextEngine Incorporated, 401 Wilshire Boulevard, Ninth Floor, Santa Monica, California 90401

Brian Funt

School of Computing Science, Simon Fraser University, 8888 University Drive, Burnaby V5A 1S6, British Columbia, Canada

Kobus Barnard

Department of Computer Science, Gould-Simpson Building, The University of Arizona, P.O. Box 210077, Tucson, Arizona 85721-0077

Received February 10, 2002; revised manuscript received July 2, 2002; accepted July 17, 2002

A neural network can learn color constancy, defined here as the ability to estimate the chromaticity of a scene's overall illumination. We describe a multilayer neural network that is able to recover the illumination chromaticity given only an image of the scene. The network is previously trained by being presented with a set of images of scenes and the chromaticities of the corresponding scene illuminants. Experiments with real images show that the network performs better than previous color constancy methods. In particular, the performance is better for images with a relatively small number of distinct colors. The method has application to machine vision problems such as object recognition, where illumination-independent color descriptors are required, and in digital photography, where uncontrolled scene illumination can create an unwanted color cast in a photograph. © 2002 Optical Society of America

OCIS codes: 330.0330, 330.1690, 330.1710, 330.1720, 100.2000.

1. INTRODUCTION

As the color of the illumination of a scene changes, the colors of the surfaces in the scene will also change. This color shift presents a problem since color descriptors will be too unstable for use in a computational vision system without something being done to compensate for it. Without color stability, most areas where color is taken into account (e.g., color-based object recognition systems[1] and digital photography) will be adversely affected even by small changes in the scene's illumination.[2] The term "color" will be used here to refer to the red–green–blue (RGB) signal recorded by a digital camera rather than what a person sees, unless the context specifically implies human color perception.

Humans exhibit some color constancy, which experiments by Brainard *et al.*[3,4] aim to quantify; however, the mechanisms behind human color constancy remain unexplained. We would like to achieve machine color constancy (i.e., automatically estimate the color of the incident illumination) as accurately as possible without regard to the process as a model of the human visual system.

In this paper we will assume that the chromaticity of the scene illumination is constant throughout the image, although its intensity may vary. The goal of a machine color constancy system will be taken to be the accurate estimation of the chromaticity of the scene illumination from a three-band, RGB digital color image of the scene. To achieve this goal, we developed a system based on a multilayer neural network. The network works with the chromaticity histogram of the input image and computes an estimate of the scene's illumination.

Calculating color-constant color descriptors is done here in two steps. The first step is to estimate the illuminant's chromaticity. The second step is to color correct the image, on the basis of the estimated illuminant chromaticity. Given an estimate of the illuminant chromaticity, the image can be color corrected[5] by using a global, von Kries type[6,7] diagonal transformation of RGB image data as shown in Eq. (1) or, equivalently, a coefficient rule scaling of the image bands. In other words, the procedure is equivalent to scaling all the camera responses on the R, G, and B channels independently by coefficients $\{k_R, k_G, k_B\}$:

$$\begin{pmatrix} R' \\ G' \\ B' \end{pmatrix} = \begin{bmatrix} k_R & 0 & 0 \\ 0 & k_G & 0 \\ 0 & 0 & k_B \end{bmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}. \tag{1}$$

The same coefficients are applied to all image pixels. The coefficients are computed so that the diagonal transformation maps colors as recorded by the camera under the scene illuminant to those that would be recorded by the camera under a standard "canonical" illuminant. The colors under the canonical illuminant then provide a color-constant representation of the scene colors. Color correction of the form given in Eq. (1) has a long history.

Although Worthey and Brill[8] have shown that broad and overlapping receptor spectral sensitivities affect the accuracy of the coefficient rule as a model of the effect of illumination change, the diagonal model is generally sufficiently accurate. It has been shown[5,9,10] that if the receptor spectral sensitivities are sharp enough, the diagonal model provides a good vehicle for color correction. If the spectral sensitivities are not sharp, they can be sharpened by using a linear transformation that converts them into a new set of spectral sensitivity functions that optimizes the diagonal model by minimizing the nondiagonal elements of the transformation matrix.

2. RELATED WORK ON COLOR CONSTANCY

Computing illuminant-independent color descriptors is an underdetermined problem, since in a three-band image (retinal or camera) with n image locations, there are $3n$ sensor measurements (three color channels times n locations), but there are $3n + 3$ unknowns (the surface descriptors plus the illuminant). All color constancy algorithms therefore impose some additional constraints to permit a solution to be obtained. The algorithms differ in the assumptions they make.

One common approach is to make some assumptions about the expected distribution of image colors. For example, Buchsbaum[11] assumes that the average of the reflected spectra corresponds to the actual illuminant. Gershon *et al.*[12] refined this idea further, counting each distinct color only once. Brainard and Freeman[13] extended beyond a simple average by constructing prior distributions that characterize illuminants and surfaces in the world. Then, given a scene, they used the Bayesian rule for *a posteriori* estimation of illuminants and surfaces. Retinex theory[14] bases color constancy on the lightness in each of the three color bands. A pixel's lightness is computed by comparing its value with other pixels in the image, generally with a bias to pixels in a localized neighborhood. In a different vein, finite-dimensional linear models for illuminants and surface reflectances have been used by several authors[15–18] in order to make the underdetermined set of equations solvable. These models make strong assumptions about the dimensionality and statistics of the surface reflectances and illuminants. For example, in the case of the Maloney–Wandell algorithm for a trichromatic system, the assumptions require that surface reflectances fall in a two-dimensional (2D) subspace. This assumption is violated so significantly in real image data that the method fails to work in practice.[19] It fails in the sense that when an image is color balanced on the basis of its estimate of the illumination, the resulting image is worse than the input image.

One of the best-performing color constancy algorithms, by Forsyth,[20] estimates color-constant descriptors for the objects in a scene under a standard canonical illuminant, on the basis of intersections of constraints given by the colors of surfaces in the scene. Finally, the "color-by-correlation" algorithm developed by Finlayson *et al.*[21,22] builds a correlation matrix that correlates the chromaticities in the image with a set of predetermined scene illuminants. The illuminant is identified as the one with the maximum correlation.

Other authors have discussed neural networks in the context of color, but none has solved the problem of estimating an unknown scene illuminant by using a neural network designed to learn the relationship between a given scene illuminant and the gamut of corresponding image colors that is likely to arise under that illuminant. For example, Hurlbert and Poggio[23,24] developed and tested a neural network that learns a version of the main stage of the Retinex algorithm, in particular the computation of lightness in a single color band. Moore *et al.*[25] developed a neural network implementation of a variant of Retinex using a VLSI analog network for speed; the network itself does not learn. Usui *et al.*[26] designed a simple three-neuron recurrent neural network that decorrelates the triplets of cone responses, thus obtaining marginally color-constant descriptors for the objects in a scene. Courtney *et al.*[27] modeled the structure of the primate visual system from the retina to the cortical area V4, with a multistage neural network. Courtney's model is not a learning model, either, but rather a neural network implementation of an existing theory. Courtney does not present any actual color constancy results with real image data, so it is unclear whether the method works.

The neural network approach[28] to color constancy that we describe below is novel in two ways. First, the network learns the connection between image colors and the color of the illuminant. Second, it works better than any previous color constancy algorithm.

3. NEURAL NETWORK APPROACH

We use a neural network to extract the relationship between a scene and the chromaticity of its illumination. To discard any intensity information, all the scene's pixels are projected into a chromaticity space. This space is then sampled and presented to a multilayer neural network. During training, the actual chromaticity of the illuminant is presented to the output of the neural network so that it can learn the relationship between the scene and its illuminant. During testing, the network produces at its two output nodes an estimate of the illuminant's chromaticity.

A. Data Representation

The neural network's input layer consists of a large number of binary inputs representing a binarized chromaticity histogram of the chromaticities of the RGBs present in the scene. In our experiments we use the *rg* chromaticity space:

$$r = R/(R + G + B),$$

$$g = G/(R + G + B). \tag{2}$$

This space has the advantage that it is bounded between 0 and 1, so it requires no additional preprocessing before being input into the neural network. If necessary, the implicit blue chromaticity component can easily be recovered:

$$b = 1 - r - g. \quad (3)$$

We also experimented with other chromaticity spaces, such as the logarithmic perspective space, where $r = \log(R/B)$ and $g = \log(G/B)$, as well as CIELAB a*, b*. In each case we obtained similar results.

Using *rg*-chromaticity space discards all spatial and intensity information, which has its pros and cons. For example, recent experiments performed on 2D versus three-dimensional gamut-mapping algorithms[29,30] showed that intensity information can help in estimating the illuminant. In the case of the neural network approach, however, a mapping from the image space into a three-dimensional space (such as RGB) would have increased the size of the neural network to the point where it would have made training impossible, both from the standpoint of training time and from the standpoint of the much larger training set that would be required.

The *rg*-chromaticity space is uniformly sampled with a step size S, so that all chromaticities within the same sampling square of size S are taken as equivalent. Each sampling square maps to a distinct network input neuron. The input neuron is set either to 0, indicating that an RGB of chromaticity *rg* is not present in the scene, or to 1, indicating that *rg* is present. The idea of a chromaticity being strictly present or absent is used for synthetic images where there is no noise, but it is modified somewhat, as discussed below in Subsection 3.B, by the preprocessing that is performed in working with real images. This quantization has the apparent disadvantage that it forgoes some of the resolution in chromaticity, and it does not represent the number of pixels having a particular chromaticity value. However, we have found that increasing the chromaticity resolution indefinitely does not improve the neural network's performance. It also appears to be the presence or absence of a given chromaticity that matters, not how often it occurs. The representation is also good in that spatial information is discarded, thereby reducing the number of possible inputs to the net, which is a major advantage for both training and testing.

A large sampling step S results in a small input layer for the neural network but loses a lot of color resolution, which when taken too far can lead to larger illumination-estimation errors. Alternatively, a small sampling step yields a very large input layer, which can make training very difficult.

Figures 1 and 2 show binarized chromaticity histograms for a natural image taken under two different illuminants, a fluorescent light (Fig. 1) and a tungsten halogen light (Fig. 2). As can be seen, the transformation between the two histograms is not simple. Moreover, as a result of to noise, filtering, and sampling errors, the number of activated bins is usually different under two different illuminants.

The output layer of the neural network produces the chromaticities r and g (in the *rg*-chromaticity space) of the illuminant. These values are real numbers in the range 0 to 1. In practice, the chromaticities of real illuminants are limited, so the neural network output values range from 0.05 to 0.9 for both the r and the g components.

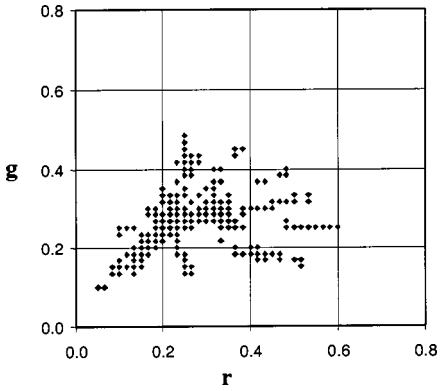

Fig. 1. Binarized histograms of a scene taken under fluorescent illuminant, as represented in the *rg*-chromaticity input space of the neural network.

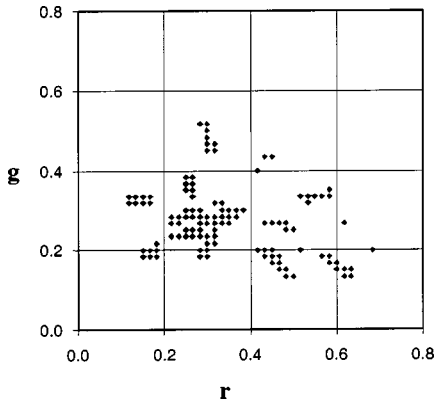

Fig. 2. Binarized histogram of the same scene as the one depicted in Fig. 1, taken under tungsten illuminant, as represented in the *rg*-chromaticity input space of the neural network.

B. Neural Network Architecture

The neural network that we used is a perceptron with two hidden layers. The first layer is large, and the input values are binarized (0 or 1), as described above. The larger the layer, the better the chromaticity resolution, but a very large layer substantially increases the training time and requires a much larger training set. Another problem with a large network is that it has a tendency to memorize the relationship between inputs and output targets and therefore has poor generalization properties. On the other hand, a small network cannot fully model the input–output mapping. This is known as the bias/variance dilemma.[31] The proper network architecture depends on the dimensionality of the function that it tries to model and on the amount and quality of training data. In our initial experiments, the neural networks with only one hidden layer yielded worse results than the ones with two hidden layers, so we focused on the networks with two hidden layers.

We experimented with different input layer sizes (512, 900, 1024, 1250, 2500, and 3600), with comparable color constancy results in all cases. The first hidden layer, H-1, contains roughly 200 neurons and the second layer, H-2, approximately 40 neurons. The output layer consists of only two neurons, corresponding to the chromaticity values of the illuminant. From our experiments we found that the size of the hidden layers can vary within a

wide range (from 25 to 400 nodes for the first hidden layer and from 5 to 50 nodes for the second one) without affecting the overall performance of the network.

All neurons have a sigmoid activation function of the form

$$y = \frac{1}{1 + \exp(-A)}, \tag{4}$$

where the activation A is the weighted sum of the inputs of the neuron, minus a threshold value. The neural network is trained with the backpropagation algorithm.[32,33] The error function used for training the network and for estimating its accuracy is the Euclidean distance between the target and the estimated illuminant in the *rg*-chromaticity space.

C. Optimizing the Neural Network

Initial tests performed with the standard neural network architecture described above showed that it took a large number of epochs to train the neural network, and consequently the training time was very long. To overcome this problem, various improvements were developed.[34]

1. Adaptive Layer

The gamut of the chromaticities encountered during training and testing is much smaller than the whole (theoretical) chromaticity space. The chromaticities are limited in part because the illuminants and surfaces are not very saturated and in part because the camera sensors overlap. To take advantage of the fact that the set of all chromaticities does not fill the whole chromaticity space, we developed an algorithm that automatically adapts the neural network's architecture to the actual chromaticity space. Thus the input layer of the network adapts itself to the chromaticity histograms such that the neural network receives input only from active nodes, where an active node is an input node that has been activated at least once during training. The inactive nodes (those input nodes that were not activated at any time during training) are purged from the neural network, together with all their links to the first hidden layer. Since all scenes are presented to the network during the first training epoch, the network's architecture, illustrated in Fig. 3, is modified only once, immediately after the first training epoch. The links from the first hidden layer, H-1, are redirected only toward the neurons in the input layer that are active (i.e., that correspond to existing chromaticities), while links to inactive nodes are eliminated. For a sampling step of 1/60 of the *rg*-chromaticity space, there are 3600 nodes, of which less than one half remain active after the first pass through the training set (the first epoch). As a direct consequence of this adaptation process, the first hidden layer, H-1, is not fully connected to the input layer.

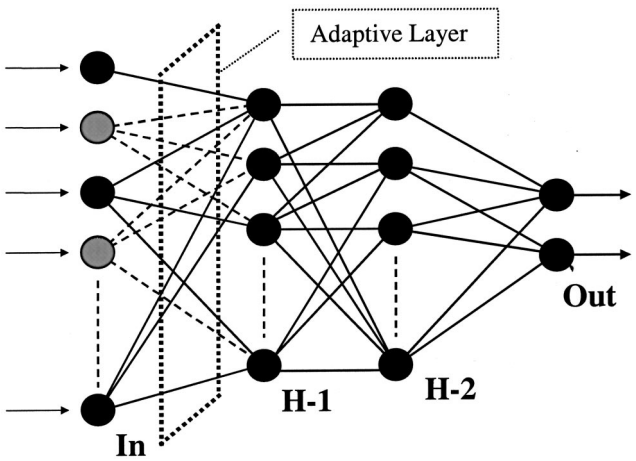

Fig. 3. Perceptron architecture. Gray input neurons denote inactive nodes as determined by the data in the training set.

Table 1. Active and Inactive Nodes versus the Total Number of Nodes in the Input Layer (*NI*)

NI	Active Nodes	Inactive Nodes
400	166	234
625	258	367
900	351	549
1600	601	999
2500	909	1591
3600	1255	2345
4900	1673	3227

Table 2. Neural Network Architectures[a]

Type	In	Links	H-1	H-2	Out
A	3600	400	200	40	2
B	3600	400	200	50	2
C	2500	200	400	30	2

[a] Neural network architectures A, B, and C described in terms of the number of nodes in each layer and the number of links between layers. In is the input layer, H-1 is the first hidden layer, H-2 is the second hidden layer, and Out is the output layer. "Links" is the number of connections between each node in the first hidden layer H-1 and the input layer In.

Table 1 shows the number of active and inactive nodes as a function of *NI*, the total number of input nodes, for typical data generated by using the sensor sensitivity functions of a SONY DXC-930 video camera. Having fewer nodes and fewer links in the network shortens the training time roughly fivefold. To shorten the training time even more, the number of links between the nodes in the first hidden layer and the input layer can actually be smaller than the total number of active nodes in the input layer. For instance, as shown in Table 2, the type C neural network has only 200 links from each node in the first hidden layer to the input layer, although the total number of active nodes is 909, as shown in Table 1.

This approach is similar in some respects to the gamut-mapping algorithms, which consider all possible RGBs that can be encountered under a set of illuminants for a given representative set of surfaces. However, whereas gamut-mapping algorithms take only the convex hull of the gamut into account, the network bases its estimate on all chromaticities from the image, including those that would be interior to the convex hull.

Of course, some chromaticities that were never encountered during training might appear in some scenes during testing; however, such previously unseen chromaticities do not present a problem. They will simply be ignored by

the neural network because there will be no link from that input node to the first hidden layer. Since there was never any information with which to train such nodes, ignoring them is better than the alternative of a fully connected input layer. The untrained weights in a fully connected input layer would only introduce error into the rest of the network.

2. Architecture-Dependent Learning Rates

The backpropagation algorithm is a gradient-descent algorithm, which changes the weights in the network until the error between the network output values and the target values falls below a threshold. The learning rate is a proportionality factor controlling how fast the network adapts its weights during training. If the learning rate is too small, the training time becomes unnecessarily large and the backpropagation algorithm might get trapped in a local minimum. On the other hand, if the learning rate is too large, the training process becomes unstable and does not converge. There is no algorithm to set exact values for the learning rate because it depends on the data set, the network architecture, and the initial random values of the network's weights and thresholds. However, there are heuristic methods to improve the training time. For example, because the sizes of the layers are so different, we used different learning rates for each layer proportional to the fan-in of the neurons in that layer.[35] Typical values for the learning rates are 0.1 for the output layer; 0.2 for the second hidden layer, H-2; and 4.0 for the first hidden layer, H-1. This shortened the training time by a factor of more than 10, to approximately five or six epochs.

Figure 4 illustrates the difference in the mean error for the standard training method with only one learning rate for all layers, as well as for the improved method with multiple learning rates. The training set was composed of 4900 scenes; 50 scenes were generated for each of the 98 illuminants in a set described in more detail in Subsection 2.D. Each scene contained from 5 to 50 colors, generated (with use of that scene's illuminant) from a database of 260 reflectance spectra including those provided by Vhrel *et al.*[36] plus additional ones that we measured with a PhotoResearch PR650 spectroradiometer. For this test, we used the neural network architecture A, as described in Table 2. When different learning rates are used for each layer, the average error drops to 0.03 after one training epoch and attains the target error of 0.01 after only eight or nine epochs.

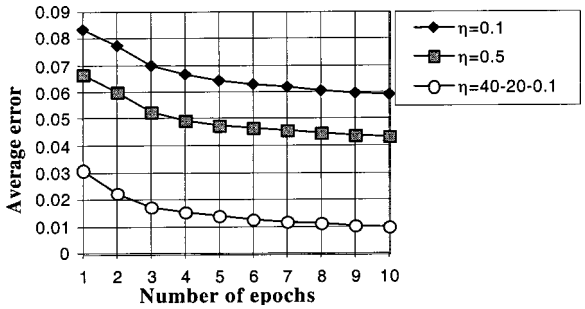

Fig. 4. Average error during the ten training epochs for three different learning-rate (η) configurations.

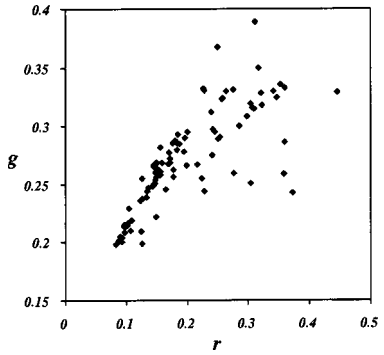

Fig. 5. Chromaticities of the 98 illuminants in our database, reflected from a surface of ideal 100% reflectance.

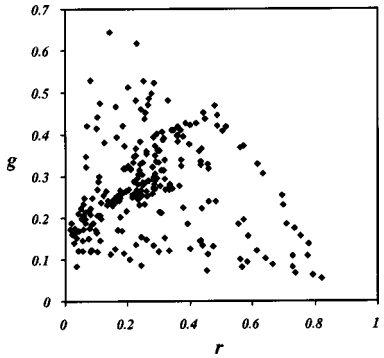

Fig. 6. Chromaticities of the 260 surfaces in our database, illuminated with equal-energy white light.

D. Databases Used for Generating Synthetic Data

If testing is done on data generated from the same surface and illuminant databases and by using the same sensor sensitivities, then any database and sensors can be used. However, our final goal is to test the neural networks on real image data of natural scenes taken with a digital camera. If a neural network that is trained on synthetic data is to be tested on real images, the sensor sensitivity functions used to train it must be as close as possible to the real sensors. Any deviation of the real camera from its model leads to differences in the RGBs observed by it and, consequently, to errors in the neural network's illuminant estimate. In this context, a SONY DCX-930 camera was calibrated,[37] and we used the calibrated sensor sensitivity functions for training and testing the networks. The illuminants in the database were measured with a PhotoResearch PR650 spectroradiometer and covered a wide range from bluish fluorescent lights to reddish tungsten ones. Colored filters were also used to create new illuminants. A blue filter was used in conjunction with four illuminants to create additional sources similar to various phases of daylight. However, strongly colored, theater-style lighting was avoided. Figure 5 shows the *rg* chromaticities of the 98 illuminants in the database, and Fig. 6 depicts the *rg* chromaticities of the 260 surfaces under equal-energy white light.

E. Training and Testing the Network

Table 2 specifies the three different network architectures for which we report results. For instance, neural network A has 3600 nodes in the input layer, 200 nodes in the

first hidden layer (H-1), 40 nodes in the second hidden layer (H-2), and 2 nodes in the output layer. Each node in the first hidden layer has 400 links to the input. All other layers are fully connected to the preceding ones, so in these layers the number of links connecting a neuron to its preceding layer is equal to the size of that layer.

In the first series of experiments, the neural network was trained on synthesized data. Each scene, representing a flat Mondrian, is composed of a variable number of surface patches seen under one illuminant. The patches correspond to matte reflectances and therefore have only one *rg* chromaticity. Of course, the same patch will have different chromaticities under different illuminants, but it will have only one chromaticity when seen under a particular illuminant. This model is a simplification of the real-world case, where, owing to noise, a flat matte patch will yield many more chromaticities scattered around the theoretical chromaticity.

Training on artificial data instead of natural scenes has the advantage that the environment can be carefully controlled, and it is easy to generate very large training sets. Each training set is composed of a large number of artificially generated scenes. For synthesized data the user can set the number of patches constituting a scene, whereas for real images (used for testing), the number of patches depends on the input image. This representation disregards any spatial information in the original image and takes into consideration only the chromaticities present in the scene.

The RGB color of a patch is computed from its randomly selected surface reflectance S^j and the spectral distribution of the illuminant E^k (selected at random, but the same for all patches in a scene) and by the spectral sensitivities of camera sensors ρ according to

$$R = \sum_i E_i^k S_i^j \rho_i^R, \quad G = \sum_i E_i^k S_i^j \rho_i^G,$$

$$B = \sum_i E_i^k S_i^j \rho_i^B. \tag{5}$$

The index i is over the wavelength domain corresponding to wavelengths in the range 380 to 780 nm.

4. EXPERIMENTS

Tests were performed on synthesized scenes as well as on real images taken with a Sony DXC-930 camera. The synthesized scenes used for testing were generated in a way similar to that for the training sets. A large number of scenes, each containing a variable number of surfaces, were synthesized from the same spectral databases and with the same sensor sensitivity functions as in training. The neural network estimates are compared with those of other color constancy algorithms.

A. Testing on Synthetic Data

Testing on synthetic data offers the advantage that the tests are not affected by noise or other artifacts. Moreover, tests can be performed on a very large data set, thus achieving reliable statistics on the performance of various color constancy algorithms. After the neural network training, the average error in estimating the illumination chromaticity for the training set data ranged from 0.0083 to 0.011, depending on the neural network architecture and the test set. When tests were done on scenes that were not part of the training set, the average error was slightly higher, ranging from 0.01 to 0.02. These average errors are also a function of the distribution of the number of patches in each scene, since scenes containing a smaller number of patches generally lead to larger errors.

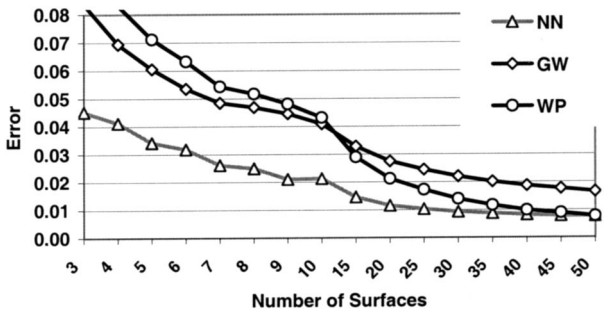

Fig. 7. Comparative results on synthesized scenes. The graph shows the average error as a function of the number of distinct colors in the scene.

In the example given in Fig. 7, the test set contains 100 random scenes for each of the 98 illuminants. The number of patches in each test scene ranges from 3 to 50, distributed uniformly. Each patch can appear only once in a test scene. The performance of the neural network (NN) algorithm is compared with the white-patch (WP) algorithm and the gray-world (GW) algorithm, described below.

The WP algorithm estimates the color of the illuminant as being the color given by the maxima taken from each of the R, G, and B channels. Since there are no "clipped" pixels (i.e., pixels for which the sensor response on a channel is saturated) in synthesized scenes, the WP algorithm performs much better on synthetic data than on real-world images.

The GW algorithm is based on the assumption that the average of the tristimulus values of the surfaces in the reflectance database illuminated by a particular light source will be the same as spatial average of the tristimulus values from a real scene under the same light. The algorithm averages the pixel values of the test image on each of the three color channels and assumes that any deviation of these average values from the database averages is caused by the color of the illuminant. Because the GW algorithm uses *a priori* knowledge about the statistical properties of the surface reflectances used for creating the test sets, it will eventually converge to zero error when tested on scenes with a very large number of patches. On real images, GW performs more poorly, because the distribution of the colors in real world images is not known *a priori*.

The superior performance of the NN algorithm is clearly apparent in Fig. 7 for scenes with a small number of patches, especially below 20. For scenes with a large number of patches, the error converges to a very small value. The good performance of the NN algorithm might allow local image processing, which could help solve the color constancy problem for scenes with multiple illuminants.[19,38]

It should be noted that both GW and WP algorithms are at an advantage relative to the NN owing to the design of the testing scenario. Statistically, the estimation errors for both WP and GW algorithms will converge to zero as the number of surfaces in the scene approaches the size of the database. In the case of the WP algorithm, this happens because the probability of a surface with a constant 100% spectral reflectance (i.e., a white surface) being present in the scene increases. There is, in fact, a reference white surface in the database. Similarly, in the case of the GW algorithm the scene average converges to the database average.

B. Testing on Real Images

The network was also tested on 48 images (of size 637 $\times$ 468 pixels) taken with the Sony DXC-930 camera under controlled conditions. The chromaticity of the illuminant was assumed to be the same as the chromaticity of a reference white patch under the same illuminant. The illuminants varied from fluorescents with added blue filters to tungsten illuminants.

The images were preprocessed before being passed to the network. The clipped and the very dark pixels were eliminated. A threshold value of 7 (on a 0–255 scale) in any of the three RGB color channels was used to select the dark pixels. The images were also smoothed by using 5×5 local averaging to eliminate noise. After preprocessing, approximately 10,000 valid image pixels were passed to the network. Owing to the sampling size of the chromaticity histogram, the number of distinct binarized histogram bins and, consequently, active inputs to the neural network representing the set of *rg*-chromaticities occurring in the image, varied from 60 to 120.

Table 3 shows the results on real images. The mean distance error represents the average Euclidean distance in *rg*-chromaticity space between the estimated and the actual illuminants. The standard deviation is also given. To relate the results to a perceptual measure of the color difference between the estimated and the actual illuminants, the mean CIE L*a*b* ΔE errors[39] are also given. The ΔE error is taken between the color of the estimated illuminant and the color of the actual one, under the following assumptions. We assume first that the RGB space is that of an sRGB-compliant device[40] and second that the two illuminants have the same luminance so that Y is equal to 100 in CIE *XYZ* coordinates. The cameras that we used are not calibrated to sRGB space, so the first assumption is violated to some extent; however, this should not have much effect since we are computing only the difference in color between the two illuminants, not either one's true color. Converting from the RGB space to the CIE L*a*b* color space involves first converting the RGB values to the CIE *XYZ* space, on the basis of the sRGB model. The tristimulus values X_n, Y_n, Z_n of the nominal white involved in the conversion from *XYZ* to CIE L*a*b* are equal to the values of the CIE D_{65} standard illuminant, with Y_n equal to 100. The conversion from *XYZ* to CIE L*a*b* was done by using the formulas in Ref. 39.

In Table 3 the illumination-chromaticity variation listed in the first row shows the average shift in the *rg*-chromaticity space between the canonical illuminant and the true illuminant of each of the test scenes. This can be considered a worst-case estimation algorithm that simply outputs the chosen canonical illuminant as the "answer" in all cases. In our experiments the canonical illuminant was selected to be the one for which the CCD camera was best color balanced. For this illuminant, the image of a white patch records identical values on all three color channels.

In every case, the errors are higher for real images (Table 3) than for synthesized ones (see Figs. 4 and 7). The average errors, larger than 0.05 for all algorithms, were almost five times higher than the average errors obtained for synthesized scenes. Noise, specularities, clipped pixels, and errors in camera calibration are some of the factors that might have affected the performance of the algorithms. The gray-world algorithm (second row) had to rely only on a model based on *a priori* knowledge gathered from the surface database. The results show that the particular distributions found in the databases from which the artificial scenes were synthesized do not match the real-world distributions of surfaces and illuminants. The white-patch algorithm (third row) suffered because of clipped pixels, noise, and the fact that the "whitest" patch may not in fact have been white but some other color.

The results for the neural network (fourth row) were obtained by using the neural network architecture B (described in Table 2) and trained with synthesized data.

Table 3. Tests on Real Images[a]

Method of Illumination Estimation	Mean Error	Standard Deviation	Mean ΔE_{Lab}
Illumination-chromaticity variation	0.090	0.062	22.38
Gray-world with average R, G, and B	0.071	0.051	15.27
White-patch with maximum R, G, and B	0.075	0.049	16.36
Neural network trained on synthetic data	0.059	0.043	15.03
2D gamut-mapping method with surface constraints only	0.054	0.047	12.90
2D gamut-mapping method with surface and illumination constraints	0.047	0.039	12.67
Neural network B with 25% specularity model	0.044	0.032	12.13

[a] Comparison of performance of the various color constancy algorithms when tested on real images. Distances are measured between the actual and the estimated illuminant in terms of Euclidean distance in *rg*-chromaticity space and CIE L*a*b* ΔE.

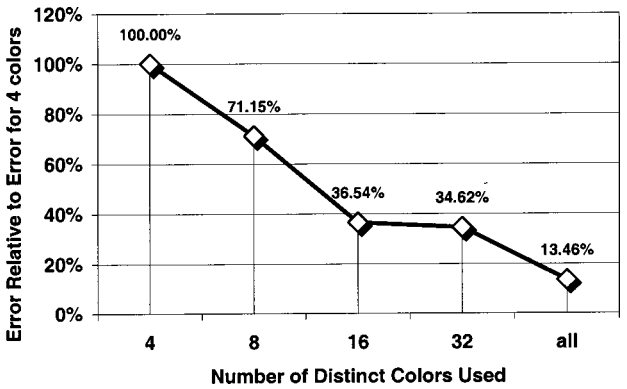

Fig. 8. Error as a function of the number of colors used. Colors were randomly selected from a single image. All values are relative to the base case of using only four distinct colors. Error drops noticeably as the number of colors increases.

The 2D gamut-mapping algorithm that uses only surface constraints (fifth row) is Finlayson's "perspective" variation[41] on Forsyth's algorithm,[20] and the extended method (sixth row) adds illumination constraints.[41] The neural network results are improved (seventh row) by modeling specular reflections in the training set, as will be discussed in more detail below. As a general rule, the more distinct colors there are in a scene, the better most color constancy algorithms are likely to perform, since having more colors implies more information to exploit. To determine the accuracy of the neural network as a function of the number of distinct colors in an image, we created new "images" by taking random subsets of the colors found in a single real image. As the test image, we took the Macbeth Colorchecker under a relatively blue light created by a fluorescent tube behind a blue filter. After the initial preprocessing was applied to the image as described above, 4, 8, 16, or 32 colors were selected at random. Fifty new sampled "images" were made for each number of colors to be selected. As well, the original image with all its initial colors was included in the testing. The relative error as a function of the number of colors, plotted in Fig. 8, clearly shows that the neural network's performance improves with the number of colors. Although Fig. 8 is based on a single image so as to factor out the effect of scene content, the results are consistent, nonetheless, with those based on all scenes.

C. Modeling Specular Reflections

The accuracy of the neural network's illumination-chromaticity estimate generally was similar to or surpassed that of the GW and WP algorithms. However, as seen above, the errors obtained with real images were significantly larger than those for the synthetic ones. After experiments with adding noise to the synthetic data, we concluded that there was a more fundamental problem requiring explanation than simply the influence of noise. We hypothesized that specular reflection was partially causing the problem, so we modeled the specular reflection in the training set.[42]

Most color constancy algorithms assume matte surface-reflection properties for the objects appearing in images. However, some algorithms[43,44] exploit specularities explicitly when calculating the illuminant's chromaticity and will fail if there are no specularities. Those algorithms use the dichromatic model of specular reflection[45] and depend on the fact that the spectrum of the specularly reflected component—that which is reflected directly from the surface of the object rather than entering the object—is approximately the same as that of the incident illumination. These algorithms detect a specularity based on its spatial structure. In contrast, the neural network's histograms contain no spatial image structure, and the network does not explicitly identify specularities in the image.

To incorporate specularities into the neural network approach, we modified the training set to include random amounts of specularity calculated by using the dichromatic reflection model, which states that the reflected light is an additive mixture of a specular and a body component. The body component describes the light that enters the object's surface before being reemitted. Therefore specularities were added to the training set simply by adding random amounts of the scene illumination's RGB to the matte component of the synthesized surface RGBs.

Two different neural network architectures, B and C from Table 2, were tested. The networks were trained with training sets containing 9800 artificially generated scenes (100 scenes for each of 98 illuminants). Each scene contained 10 to 100 randomly selected surfaces. To each of the generated RGB values we added a random amount w of the scene's illumination. The value of w for a scene was computed as the product, $w = Sp$, of a user-controlled maximum value for the specular component S and a random subunitary coefficient p. Since surface specularity is not uniformly distributed in a real image, we created a nonuniform distribution by squaring a uniformly distributed random function: $p = \text{rand}(\cdot)^2$. This model has an expected value for the specular coefficient p of 33.3% and a standard deviation of 29.81%. This ensures that generally, only a few surfaces in the scene will be highly specular while a large variance of specularity is retained. A random amount of white noise, to a maximum $\pm 5\%$ of the RGB values, was then also added to the data.

We generated training sets with different amounts of maximum specularity and trained the networks for ten epochs on each training set. All networks of the same architecture were trained by starting from a network initialized with identical random weights, which ensures that the results depend only on the training sets and not on the network's starting state. When finished, we have a separate neural network for each training set.

For these networks, the average error in estimating the illumination chromaticity for the images in the training set ranged from 0.008 to 0.011. When the networks were tested on synthesized scenes that were not part of the training set, the average error ranged from 0.012 to 0.022. More important, on the test set of real images, the specularity modeling improved the neural network's performance significantly. The results are summarized in Table 4, Table 5, and row 7 of Table 3.

The results in Tables 4 and 5 show that there is a significant improvement in the network performance for networks trained on images with a specular component. The error drops from an average of 0.059 to 0.044 mea-

Table 4. Results with Network C with Use of Specularity Modeling[a]

Specularity (%)	Mean Error	Standard Deviation	Improvement (%)
0	0.058	0.047	—
5	0.051	0.037	11.8
10	0.056	0.038	3.4
25	0.045	0.036	22.4
50	0.047	0.032	18.9

[a] Results for the C network trained for different amounts of specularity and then tested on images of real scenes. The error is reported in terms of Euclidean distance in *rg*-chromaticity space between the actual and the estimated illuminant chromaticities.

Table 5. Results with Network B with Use of Specularity Modeling[a]

Specularity (%)	Mean Error	Standard Deviation	Improvement (%)
0	0.059	0.043	—
5	0.051	0.035	13.5
10	0.044	0.026	25.4
25	0.044	0.030	25.4
>50	≈0.044	≈0.035	25.4

[a] Results for the B network trained for different amounts of specularity and then tested on images of real scenes. The error is reported in terms of Euclidean distance in *rg*-chromaticity space between the actual and the estimated illuminant chromaticities.

sured in the *rg*-chromaticity space. As can be seen from Table 3, the neural network's estimates are more accurate than those of any of the other methods tested. Nonetheless, the error (Table 3, row 7) for real images is still four times larger than the average error obtained with synthetic images (~0.01, as can be seen from the NN curve in Fig. 7). This discrepancy leads to the question of whether training on real image data will improve the results and the accompanying problem of how to obtain a large enough training set of real images.

D. Training and Testing on Real Images

As shown in the previous subsections, the neural network does not work as well with real images as with synthetic ones. It is possible that training on real image data will improve the network's performance on real images. Another benefit of training on real images is that it eliminates the need for camera calibration.

The main problem in training on real images is how to obtain a sufficiently large number of images. Training sets need to contain 10,000 or more images in which the illumination conditions are known. Since obtaining thousands of images under controlled conditions is not practical, we had to take a different approach. As an alternative, we created new image histograms from subsets of the pixels found in a modest set of controlled real images. In essence, this method synthesizes new scenes from real data. The training sets were generated from only 44 images. The images used for training and testing the neural network were taken with a Kodak DCS460 digital camera. This camera has the advantage over the Sony DXC-930 camera in that it is portable and has a wider dynamic range (8 to 9 bits). It also has greater spatial resolution, but the extra resolution is not necessary, so the images were reduced to a resolution of 1000 × 600 to speed up the preprocessing.

The images contain outdoor scenes, taken in daylight at different times of day, as well as indoor scenes, taken under a variety of tungsten and fluorescent light sources both with and without colored filters. The chromaticity of the light source in each scene was determined by taking an image of a reference white reflectance standard in the same environment. The average distance ΔE in the CIE L*a*b* space between the chromaticity of one of the light sources and the chromaticity of the reference light source (i.e., the source for which the camera produces R = G = B for a white reflectance standard) was 17.05 with a standard deviation of 9.51.

To obtain even more training and test data, all images were downloaded from the camera by using two different camera-driver color-balance settings ("Daylight" and "Tungsten"). These settings performed a predefined color adjustment; however, this did not mean that the images were correctly color balanced, since the actual illumination under which any particular image was taken was usually different from that anticipated by either of the two possible camera settings. We made no assumptions regarding the camera sensors nor about the two color-balance settings of the camera driver. We measured the gamma of the camera, which we found to be the same for both color-balance settings, and linearized the images accordingly.

The neural network was trained for five epochs on data derived from the 44 real images. Each image was preprocessed in the same way as described above. The set of chromaticities appearing in each of the 44 preprocessed images was then randomly sampled to derive a much larger number of training "images." A total of 50,000 images containing between 10 and 100 distinct chromaticities were generated in this way.

Table 6 compares the performance of the neural network relative with other color constancy algorithms on a test set of 42 real images not included in the neural network's training set. To make the comparisons, we also trained a neural network on 123,000 synthetic scenes based on the spectral sensitivity functions of the Kodak DCS460 camera, using the same databases of illuminants and surface reflectances as before. As well, we generated gamuts for the gamut-mapping algorithms based on the DCS460 sensors. As in the other tables, the mean error and standard deviation are computed in the *rg*-chromaticity space and as CIE L*a*b* ΔE.

The network trained on real data clearly outperforms the network trained on synthetic data as well as the other color constancy algorithms. The accuracy of all the gamut-mapping algorithms was not as good as we initially expected. One possible reason is that the sensors of the Kodak camera are rather broad, which makes them less suitable for diagonal transformations[9] and gamut-mapping algorithms.[30] Inaccuracies in the calibration of the camera's spectral sensitivity functions also reduce the effectiveness of both the gamut-mapping algorithms and

Table 6. Estimation Errors of Color Constancy algorithms (*I*)[a]

Illumination-Estimation Algorithm	Mean Error	Standard Deviation	Mean ΔE_{Lab}
Illumination-chromaticity variation	0.0956	0.0789	21.22
Database gray-world	0.0553	0.0295	19.61
White-patch	0.0716	0.0464	19.47
Gamut-mapping algorithms			
2D Hull average with surfaces only	0.0861	0.0420	21.43
2D Hull average with surfaces and illumination	0.0839	0.0436	20.68
2D Constrained-illumination hull average	0.0782	0.0427	20.61
2D Surface constrained-illumination average	0.0821	0.0437	22.22
2D Surface constrained-chromaticity average	0.0824	0.0439	22.23
Neural networks			
RG neural net trained on synthetic data with specularity	0.0748	0.0493	14.84
RG neural net trained on real images	0.0207	0.0231	5.67

[a] Comparison of the performance of the various color constancy algorithms when tested on Kodak DCS 460 images. The last two rows show the performance improvement obtained by training on real image data instead of synthetic image data. Training the network on real image data reduces the error by more than half.

the neural network trained on synthetic data. Training the neural network on real images reduces the average illumination-estimation distance in *rg*-chromaticity space to only 5.67 in CIE L*a*b* space.

E. Example of Color Correction

Figure 9 shows an example of color correction based on the illuminant estimate provided by various color constancy algorithms. Given an estimate of illuminant chromaticity, the image is then corrected using the diagonal model.[5] After application of the diagonal transformation, the intensity of the pixels is adjusted such that the average intensity of the image remains constant: The average image intensity is computed before and after the diagonal transformation, and then the corrected image is scaled globally such that its average intensity becomes equal to the average intensity of the original image.

In Fig. 9, the top-left panel shows the original image, taken under an unknown illuminant with the Sony camera. The top-right panel shows the target image, taken under the canonical illuminant. Given only the image in the top-left panel, our color-correction goal is to produce an image that matches the top-right image as closely as possible. The middle-left image is calculated by first using the neural network to estimate the illuminant of the top-left image followed by the appropriate scaling of each of the RGB channels on the basis of the estimated illuminant. Similarly, the middle-right image shows the result of the gamut-mapping algorithm that uses both surface and illuminant constraints. The bottom-left panel gives the WP algorithm result, and the bottom-right panel shows the GW result.

5. TRAINING AND TESTING ON UNCALIBRATED IMAGES

The experiments described above were done by using images taken with calibrated cameras (i.e., cameras for which the sensor sensitivity functions, white balance, and amount of gamma correction were known). In dealing with uncalibrated images, such as images downloaded from the Internet or taken with an unknown camera (a common case for photo-finishing labs), the problem becomes more difficult.

First, there is the issue of estimating the illuminant. The camera's white balance, its gamma value (gamma values other than 1.0 result in the image intensity becoming a nonlinear function of scene intensity), and its sensor sensitivity functions are unknown. Each of these factors can have an effect on the illuminant estimate. Consumer digital cameras produce an image that is intended for CRT monitors, so the expected variation in gamma is relatively small; but, on the other hand, the white balance and sensor sensitivity functions of these cameras vary significantly.

Second, even if the color of the illuminant is estimated correctly, there remains the problem of how to correct all the nonwhite colors in an image of unknown gamma. In previous work[46] we have shown that as in the linear case, a diagonal transformation can be used for color correction of uncalibrated nonlinear images. Although for nonlinear images the off-diagonal terms of the full 3×3 transformation matrix are larger relative to the diagonal terms, the perceptual error of the transformation induced by ignoring the off-diagonal terms remains small, and therefore the diagonal transformation remains a good model of illumination change.

For this experiment on uncalibrated images, we used a database of 900 images, collected with a variety of digital cameras: Kodak DCS 460, DC 210 and DC 280, Olympus C2020Z and C820L, Hewlett-Packard PhotoSmart C30 and 912xi, Fuji 600, 2400Z and MX-1200, Polaroid PDC 640 and PDC 2300, Canon Powershot S10, Ricoh RDC-5000, and Toshiba PDR-M60. The actual illuminant chromaticity was determined for each image by measuring the RGB of a gray card in the image.

The images were taken over a long period of time (over one year), under very diverse lighting conditions (indoors with and without flash, outdoors under natural light, outdoors with fill-in flash, etc.). All images were down-

sampled to a fixed size such that the larger of the width or height has 150 pixels.

We trained numerous networks of different architectures to find the one yielding the best illumination estimates. The best network designed for the case of a 1024-bin *rg*-chromaticity binary histogram contains 206 nodes in the input layer (corresponding to a total of 206 active bins in the chromaticity histogram), one hidden layer composed of ten neurons, and two output neurons representing the *r* and *g* chromaticity of the illuminant. This network turns out to be smaller than the ones used in our earlier experiments, especially those done on synthetic data, but is optimized for the actual training data.[31]

For this experiment we employed the leave-one-out cross-validation approach[47,48]: We excluded one image at a time from the image set, trained a neural network on the remaining 899 images (as described in Subsection 4.D), and then tested it only on the one excluded image. This process was repeated 900 times, resulting in 900 different neural networks. This process is computationally intensive but nonetheless feasible, since the training time for a single network is approximately 30 s. The leave-one-out cross validation allows us to test the network approach on a large number of images, none of which the network was trained on.

The estimation errors were quite small, with an average of 0.0226, a maximum of 0.0774, and a root-mean-square (RMS) error of 0.0276. In terms of CIE L*a*b*, the average $\Delta E_{\mathrm{Lab}} = 6.70$. The CIE L*a*b* errors were computed by representing the estimated illuminant chromaticity in terms of their colors on an sRGB-compliant monitor. The nominal white tristimulus values involved in the conversion[39] from *XYZ* to CIE L*a*b* were derived from CIE D_{65}, as discussed in Subsection 4.B. The CIE L* value was set to 50 for all illuminants to ensure that the CIE L*a*b* errors reflected differences only in a* and b*.

All these results are compared in Table 7 with the competing color constancy algorithms that are described below. In each case, there are some difficulties in making a fair comparison. For example, the gamut-mapping algorithm, against which we benchmarked the neural network in Subsection 4.D, requires camera calibration, but the test set is composed of 900 uncalibrated images from a

Fig. 9. Color correction of real images: Top left, original image; top right, target image; middle left, neural network estimate; middle right, gamut-mapping algorithm; bottom left, WP algorithm; bottom right, GW algorithm.

Table 7. Estimation Errors of Color Constancy Algorithms (II)[a]

Algorithm	Mean Error	RMS Error	Mean ΔE_{Lab}
Illumination chromaticity Variation	0.0403	0.0576	11.60
Database gray-world	0.0292	0.0381	8.26
White-patch	0.0311	0.0438	8.76
Color by correlation	0.0292	0.0389	8.45
Neural network trained on 900 images	0.0226	0.0276	6.70

[a] Comparison of the accuracy of illuminant estimation errors of different algorithms, evaluated on a collection of 900 uncalibrated images.

range of cameras. Therefore in this experiment we did not compare the network with the gamut-mapping algorithm. Instead, we adapted the related color-by-correlation algorithm developed by Hubel and Finlayson and colleagues.[21,22] To implement this algorithm, we used the 1024-bin chromaticity histogram to bin all illuminants encountered in the image database. After binning, we obtained 46 distinct illuminants.

In order to have unbiased results, we employed the same leave-one-out cross-validation method that we used for testing the neural network. We excluded one image at a time and computed the correlation matrix by using actual chromaticity data from the other 899 images. For any given test image, the histogram vector (obtained from the 2D *rg*-chromaticity histogram, rearranged as a row vector) containing values of either 0 or 1 is multiplied by the correlation matrix. If H_i is the image histogram of image i that was not used for computing the correlation matrix and $\mathbf{C}$ is the correlation matrix, the algorithm computes a vector L_i, where L_{ij}, the jth component of L_i, is the likelihood of illuminant j in image i:

$$L_i = H_i \cdot \mathbf{C}. \tag{6}$$

Given the properties of our image database, the correlation matrix $\mathbf{C}$ has 206 rows (the number of active bins in the histogram) and 46 columns (the number of illuminants). Finally, we used a weighted average over the vector L_i to compute the best estimate for the scene illuminant. We repeated this procedure for all 900 images.

The database GW algorithm estimates the chromaticity of the illuminant by averaging the RGBs of all pixels in the image and then converting this average RGB triplet $\{R_{\mathrm{av}}, G_{\mathrm{av}}, B_{\mathrm{av}}\}$ into the *rg*-chromaticity domain:

$$\begin{cases} r_\mu = R_{\mathrm{av}}/(R_{\mathrm{av}} + G_{\mathrm{av}} + B_{\mathrm{av}}), \\ g_\mu = G_{\mathrm{av}}/(R_{\mathrm{av}} + G_{\mathrm{av}} + B_{\mathrm{av}}). \end{cases} \tag{7}$$

The final estimation is obtained by normalizing this *rg* value by the average chromaticity of the illuminants from all images in the image database $\{r_{\mathrm{ill}}, g_{\mathrm{ill}}\}$ relative to the value of the chromaticity of white $\{r_{\mathrm{wh}} = 1/3, g_{\mathrm{wh}} = 1/3\}$:

$$\begin{cases} r_{\mathrm{GW}} = r_{\mathrm{wh}} r_\mu r_{\mathrm{ill}} \\ g_{\mathrm{GW}} = g_{\mathrm{wh}} g_\mu / g_{\mathrm{ill}} \end{cases} \Leftrightarrow \begin{cases} r_{\mathrm{GW}} = 1/3 r_\mu / r_{\mathrm{ill}}, \\ g_{\mathrm{GW}} = 1/3 g_\mu / g_{\mathrm{ill}}. \end{cases} \tag{8}$$

The WP algorithm is the same as in our previous experiments and is based on the values of the brightest R, G, and B pixels in the image. Since most images contain relatively large numbers of clipped and near-clipped pixels, this method is not very accurate.

The illumination-chromaticity variation shows the distribution of the actual illuminants around camera white and is described in more detail in Subsection 4.B. Since the images are of natural scenes under normal lighting conditions (we did not use colored filters as we did in the earlier tests) with use of the cameras' built-in color correction algorithms, the average chromaticity variation is smaller than before, but it perhaps reflects much better the range of chromaticity variation and estimation errors than we can expect from consumer cameras under typical conditions.

6. CONCLUSIONS

A novel neural network approach to achieving color constancy has been developed and tested. The neural network learns to estimate the chromaticity of the scene illumination on the basis of the colors present in the image. The method is novel in that although neural networks have been discussed in the context of color, none has solved the problem of estimating an unknown scene illuminant by learning the relationship between the colors appearing in an image and the scene illuminant. When trained on synthesized images, the network outperforms other color constancy algorithms on tests done on synthesized images. When tested on real images, it performs well, although the errors are larger than for synthetic data. Introducing specular reflections into the model reduced the error for tests performed on real images, but not to the point of rivaling the results on synthetic test data. Much better results were obtained on real images by a network trained on real images. A sufficiently large training set for real-data training was created by taking subsets of the colors found in a much smaller set of real images. The resulting average ΔE CIE L*a*b* error is small and compares favorably with that obtained by human subjects in the experiments of Brainard and co-workers.[3,4] Since the error is also lower than that obtained by the gray-world and the white-patch methods, which have been previously used for color-cast removal, it could also lead to improved color balance in applications such as digital photography.

ACKNOWLEDGMENTS

This work was conducted primarily at Simon Fraser University. Funding was provided by the Natural Sciences and Engineering Research Council of Canada, Simon Fraser University, and Hewlett Packard Corporation.

REFERENCES

1. M. J. Swain and D. Ballard, "Color indexing," Int. J. Comput. Vision **7**, 11–32 (1991).
2. B. Funt, K. Barnard, and L. Martin, "Is color constancy good enough?" in *Proceedings of the Fifth European Conference on Computer Vision*, H. Burkhardt and B. Neumann, eds. (Springer, Berlin, 1998), pp. 445–459.
3. D. H. Brainard and B. A. Wandell, "Asymmetric color matching: how color appearance depends on the illuminant," J. Opt. Soc. Am. A **9**, 1433–1448 (1992).

4. D. H. Brainard, W. A. Brunt, and J. M. Speigle, "Color constancy in the nearly natural image. 1. Asymmetric matches," J. Opt. Soc. Am. A **14**, 2091–2110 (1997).
5. G. Finlayson, M. Drew, and B. Funt, "Color constancy: generalized diagonal transforms suffice," J. Opt. Soc. Am. A **11**, 3011–3020 (1994).
6. J. von Kries, "Chromatic adaptation," in *Sources of Color Vision*, D. L. MacAdam, ed. (MIT Press, Cambridge, Mass., 1970). Originally published in *Festschrift der Albrecht-Ludwigs-Universitat*, 1902), pp. 145–148.
7. J. von Kries, "Influence of adaptation on the effects produced by luminous stimuli," in *Sources of Color Vision*, D. L. MacAdam, ed. (MIT Press, Cambridge, Mass., 1970). Originally published in *Handbuch der Physiologie des Menschen*, 1905), Vol. 3, pp. 109–282.
8. J. A. Worthey and M. H. Brill, "Heuristic Analysis of von Kries color constancy," J. Opt. Soc. Am. A **3**, 1708–1712 (1986).
9. G. Finlayson, M. Drew, and B. Funt, "Spectral sharpening: sensor transformations for improved color constancy," J. Opt. Soc. Am. A **11**, 1553–1563 (1994).
10. K. Barnard, F. Ciurea, and B. Funt, "Sensor sharpening for computational color constancy," J. Opt. Soc. Am. A **18**, 2728–2743 (2001).
11. G. Buchsbaum, "A spatial processor model for object colour perception," J. Franklin Inst. **310**, 1–26 (1980).
12. R. Gershon, A. D. Jepson, and J. K. Tsotsos, "From [R, G, B] to surface reflectance: computing color constant descriptors in images," Perception **17**, 755–758 (1988).
13. D. H. Brainard and W. T. Freeman, "Bayesian color constancy," J. Opt. Soc. Am. A **14**, 1393–1411 (1997).
14. E. H. Land and J. J. McCann, "Lightness and Retinex theory," J. Opt. Soc. Am. **61**, 1–11 (1971).
15. J. Cohen, "Dependency of the spectral reflectance curves of the Munsell color chips," Psychon. Sci. **1**, 369–370 (1964).
16. D. B. Judd, D. L. MacAdam, and G. W. Wyszecki, "Spectral distribution of typical daylight as a function of correlated color temperature," J. Opt. Soc. Am. **54**, 1031–1040 (1964).
17. B. A. Wandell, "The synthesis and analysis of color images," IEEE Trans. Pattern Anal. Mach. Intell. **PAMI9**, 2–13 (1987).
18. L. Maloney and B. A. Wandell, "Color constancy: a method for recovering surface spectral reflectance," J. Opt. Soc. Am. A **3**, 29–33 (1986).
19. G. Finlayson, B. Funt, and K. Barnard, "Color constancy under varying illumination," in *Proceedings of the Fifth International Conference on Computer Vision*, W. E. L. Grimson, ed. (IEEE Computer Society Press, Los Alamitos, Calif., 1995), pp. 720–725.
20. D. A. Forsyth, "A novel algorithm for color constancy," Int. J. Comput. Vision **5**, 5–36 (1990).
21. G. Finlayson, P. Hubel, and S. Hordley, "Color by correlation," in *Proceedings of the IS&T/SID Fifth Color Imaging Conference: Color Science, Systems and Applications*, (Society for Imaging Science and Technology, Springfield, Va., 1997), pp. 6–11.
22. G. Finlayson, S. Hordley, and P. Hubel, "Color by correlation: a simple unifying framework for color constancy," IEEE Trans. Pattern Anal. Mach. Intell. **23**, 1209–1221 (2001).
23. A. C. Hurlbert and T. A. Poggio, "Synthesizing a color algorithm from examples," Science **239**, 482–485 (1988).
24. A. C. Hurlbert, "Neural network approaches to color vision," in *Neural Networks for Perception: Vol. 1: Human and Machine Perception*, H. Wechsler, ed. (Academic, San Diego, Calif., state, 1991), pp. 266–284.
25. A. Moore, J. Allman, and R. M. Goodman, "A real-time neural system for color constancy," IEEE Trans. Neural Netw. **2**, 237–247 (1991).
26. S. Usui, S. Nakauchi, and Y. Miyamoto, "A neural network model for color constancy based on the minimally redundant color representation," in *Proceedings of the International Joint Conference on Neural Networks* (International Neural Network Society, Mt. Royal, N.J., 1992), Vol. 2, pp. 696–701.
27. S. M. Courtney, L. F. Finkel, and G. Buchsbaum, "A multistage neural network for color constancy and color induction," IEEE Trans. Neural Netw. **6**, 972–985 (1995).
28. B. Funt, V. Cardei, and K. Barnard, "Learning Color Constancy," in *Proceedings of the IS&T/SID Fourth Color Imaging Conference: Color Science, Systems and Applications* (Society for Imaging Science and Technology, Springfield, Va., 1996), pp. 58–60.
29. K. Barnard, "Practical colour constancy," Ph.D. thesis (Simon Fraser University, Burnaby, B.C., Canada, 1999).
30. K. Barnard, V. Cardei, and B. Funt, "A comparison of computational color constancy algorithms; part one: methodology and experiments with synthesized data," IEEE Trans. Image Process. (to be published).
31. S. Geman, E. Bienenstock, and R. Doursat, "Neural networks and the bias/variance dilemma," Neural Comput. **4**, 1–58 (1992).
32. D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Vol. I: Foundations*, D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, eds. (MIT Press, Cambridge, Mass., 1986), pp. 318–362.
33. J. Hertz, A. Krogh, and R. G. Palmer, *Introduction to the Theory of Neural Computation* (Addison-Wesley, Reading, Mass., 1991).
34. V. Cardei, B. Funt, and K. Barnard, "Modeling color constancy with neural networks," in *Proceedings of the International Conference on Vision, Recognition, and Action: Neural Models of Mind and Machine* (Center for Adaptive Systems, Boston University, Boston, Mass., 1997).
35. D. Plaut, S. Nowlan, and G. Hinton, "Experiments on learning by back propagation," Tech. Rep., CMU-CS-86-126 (Carnegie-Mellon University, Pittsburgh, Pa., 1986).
36. M. J. Vrhel, R. Gershon, and L. S. Iwan, "Measurement and analysis of object reflectance spectra," Color Res. Appl. **19**, 4–9 (1994).
37. K. Barnard and B. Funt, "Camera characterization for color research," Color Res. Appl. **27**, 153–164 (2002).
38. K. Barnard, G. Finlayson, and B. Funt, "Color constancy for scenes with varying illumination," Comput. Vision Image Understand. **65**, 311–321 (1997).
39. G. Wyszecki and W. Stiles, *Color Science: Concepts and Methods, Quantitative data and Formulae*, 2nd ed. (Wiley, New York, 1982).
40. M. Anderson, R. Motta, S. Chandrasekar, and M. Stokes, "Proposal for a standard default color space for the Internet–sRGB," in *Proceedings of the IS&T/SID Fourth Color Imaging Conference: Color Science, Systems and Applications* (Society for Imaging Science and Technology, Springfield, Va., 1996), pp. 238–246.
41. G. Finlayson, "Color in perspective," IEEE Trans. Pattern Anal. Mach. Intell. **18**, 1034–1038 (1996).
42. B. Funt, V. Cardei, and K. Barnard, "Neural network color constancy and specularly reflecting surfaces," in *AIC Color 97 Proceedings of the 8th Congress of the International Colour Association* (Color Science Association of Japan, Tokyo, 1997), Vol. II, pp. 523–526.
43. H. Lee, "Method for computing the scene-illuminant chromaticity from specular highlights," J. Opt. Soc. Am. A **3**, 1694–1699 (1986).
44. W. M. Richard, *Automatic Detection of Effective Scene Illuminant Chromaticity from Specular Highlights in Digital Images*, M.Sc. thesis (Rochester Institute of Technology, Rochester, N.Y., 1995).
45. S. A. Shafer, "Using color to separate reflection components," Color Res. Appl. **10**, 210–218 (1985).
46. V. Cardei and B. Funt, "Color correcting uncalibrated digital images," J. Imaging Sci. Technol. **44**, 288–294 (2000).
47. R. L. Eubank, *Spline Smoothing and Nonparametric Regression* (Marcel Dekker, New York, 1988).
48. J. Moody, "Prediction risk and architecture selection for neural networks," in *From Statistics to Neural Networks: Theory and Pattern Recognition Applications*, V. Cherkassky, J. H. Friedman, and H. Wechsler, eds., NATO ASI Series F (Springer-Verlag, Berlin, 1994), pp. 147–165.