Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®

J. Parallel Distrib. Comput. 64 (2004) 79–88

Journal of Parallel and Distributed Computing

http://www.elsevier.com/locate/jpdc

A parallel algorithm for color constancy

Marc Ebner

Universität Würzburg, Lehrstuhl für Informatik II, Am Hubland, 97074 Würzburg, Germany

Received 21 April 2002; revised 3 June 2003

Abstract

Objects retain their color in spite of changes in the wavelength and energy composition of the light they reflect. This phenomenon is called color constancy and plays an important role in computer vision research. We have devised a parallel algorithm for color constancy. The algorithm runs on a two-dimensional grid of processors each of which can exchange information with its four neighboring processors. Each processor calculates local average color. This information is then used to estimate the reflectances of the object. The algorithm was tested on several images of everyday objects. The algorithm also works for scenes where the illuminant changes smoothly over the image.

© 2003 Elsevier Inc. All rights reserved.

Keywords: Color constancy; Artificial retina; Gray-world assumption

1. Introduction

The human visual system is able to correctly perceive the color of objects irrespective of the light which illuminates the scene. That is, the leaves of a tree still look green to a human observer even if the tree is illuminated with red light and the leaves actually reflect more red than green light. One is somehow able to discount the illuminant and extract a measure of the object's reflectance properties [38]. The same ability would also be useful for a robot which has to work under different lighting conditions. Suppose a robot's task is to distinguish a red from a green apple. Then it should be possible for the robot to determine the color of the apple irrespective of the light which illuminates the apple. The task of computing color constant descriptors for an image is known as the problem of color constancy. If we are able to correctly determine the reflectances of an object viewed with a digital camera it makes the task of object recognition much easier. Thus, the problem of color constancy is of particular importance for the task of object recognition [18,21]. Determining the reflectances of an object may also be used in the field of consumer photography. The colors appearing in a photograph should be the same as the colors which were experienced by the person who took the photograph. An introduction to image processing techniques may be found in [26,29,30].

Numerous solutions to the problem of color constancy have been proposed, i.e. Land's retinex theory [31,32], variants of the retinex theory [6,7,23,29], gamut-constraint methods [3,19,20], perspective color constancy [15], color by correlation [4,17], the gray-world assumption [8,25], recovery of basis function coefficients [24,28,34], mechanisms of light adaptation coupled with eye movements [12], neural networks [10,11,22,27,35,36], minimization of an energy function [37], comprehensive color normalization [18], committee-based methods which combine the output of several different color constancy algorithms [9] or use of genetic programming [13]. We now summarize some background on color image formation and discuss the two most widely known algorithms for color constancy: white-patch and the gray-world assumption.

2. Color image formation

The response of a sensor at position $\mathbf{x}_s$ measuring the light reflected from a Lambertian surface at position $\mathbf{x}_o$ is given by

$$\mathbf{I}(\mathbf{x}_s) = \mathbf{n}_l \cdot \mathbf{n}_o \int_\lambda R(\lambda, \mathbf{x}_o) L(\lambda) \mathbf{S}(\lambda)\, d\lambda,$$

where $\mathbf{I}(\mathbf{x}_s)$ is a vector of sensor responses, $\mathbf{n}_l$ is the unit vector pointing in the direction of the light source, $\mathbf{n}_o$ is

E-mail address: ebner@informatik.uni-wuerzburg.de.
URL: http://www2.informatik.uni-wuerzburg.de/staff/ebner/welcome.html.

0743-7315/$ - see front matter © 2003 Elsevier Inc. All rights reserved.
doi:10.1016/j.jpdc.2003.06.004

the unit vector corresponding to the surface normal, $R(\lambda, \mathbf{x}_o)$ specifies the percentage of light with wavelength λ reflected by the surface at position $\mathbf{x}_o$, $L(\lambda)$ is the intensity of light hitting the surface and $\mathbf{S}(\lambda)$ specifies the sensor's response functions [18]. The integration is done over all wavelengths to which the sensors respond. Assuming ideal sensors for red, green and blue light $(S_i(\lambda) = \delta(\lambda - \lambda_i))$ with $i \in \{r, g, b\}$ and a light source which illuminates the surface at a right angle the equation simplifies to

$$I_i(\mathbf{x}_s) = R(\lambda_i, \mathbf{x}_o) L(\lambda_i)$$

where $I_i(\mathbf{x}_s)$ denotes the ith component of the vector $\mathbf{I}(\mathbf{x}_s)$. Thus, if we assume that the camera's sensors are sufficiently close to ideal sensors, color constancy may be achieved by independently scaling the three color bands. Little may be gained by using a more general transform [2,16].

In this case the light illuminating the scene simply scales the reflectances. If there exists at least one pixel for each band which reflects all light for this particular band, one could simply rescale all color bands to the range $[0, 1]$.

$$R(\lambda_i, \mathbf{x}_o) = \frac{I_i(\mathbf{x}_s)}{L_{\max}(\lambda_i)}$$

with $L_{\max}(\lambda_i) = \max_{\mathbf{x}}\{I_i(\mathbf{x})\}$. This algorithm is called the white-patch retinex algorithm [21].

Another possibility would be to calculate space average color of the image and use this information to estimate the intensities of the light illuminating the scene. If one assumes that the reflectances of the surface are uniformly distributed over the interval $[0, 1]$, one gets [13]:

$$\begin{aligned}\frac{1}{N}\sum_{\mathbf{x}}^{N} I_i(\mathbf{x}) &= \frac{1}{N}\sum_{\mathbf{x}}^{N} R(\lambda_i, \mathbf{x}) L(\lambda_i) \\ &= L(\lambda_i)\frac{1}{N}\sum_{\mathbf{x}}^{N} R(\lambda_i, \mathbf{x}) \\ &= L(\lambda_i)\frac{1}{2}.\end{aligned}$$

This is the so-called gray-world assumption. Thus, for a sufficiently complex image one can estimate the intensities of the light illuminating the scene as twice the space average color.

$$L(\lambda_i) = \frac{2}{N}\sum_{\mathbf{x}}^{N} I_i(\mathbf{x}).$$

With this information the reflectances can be calculated as follows:

$$R(\lambda_i, \mathbf{x}_o) = \frac{I_i(\mathbf{x}_s)}{L(\lambda_i)}.$$

Both cues, space-average scene color as well as the color of the highest luminance patch might be used by the human visual system to estimate the color of the light illuminating the scene [33].

3. A parallel algorithm for color constancy

We have devised a parallel algorithm for the problem of color constancy [14]. The algorithm assumes a two-dimensional mesh of processing elements [1]. It is constrained to obtain and exchange information only locally but not globally. This property is very important in order for the algorithm to be scalable to arbitrary sizes. Each processing element is able to exchange information with its four neighboring elements as shown in Fig. 1. The elements receive an image as input and their task is to compute the reflectances of the objects shown in the image. We have one element per pixel.

Each element consists of three sub-elements, one for each color band. We assume that we have three different color bands red, green and blue (Fig. 2). Colors are adjusted for each band independently. Each sub-element has access to four data paths (left, right, up and down) to and from neighboring elements, one temporary storage (tmp), and to the intensity value of the current pixel for its color band (pixel). The algorithm basically calculates the local average scene color. The element's estimate of the average scene color is distributed to the neighboring elements. The data received from the neighboring elements are used to update the current estimate. Each sub-element runs the following algorithm:

```
1.) $\text{avg} = (\text{left} + \text{right} + \text{up} + \text{down})/4.0$
2.) $\text{tmp} = \text{pixel} \cdot p + \text{avg} \cdot (1 - p)$
3.) $\text{tmp} \rightarrow \text{left, right, up, down}$
4.) $\text{out} = \text{pixel}/(2 \cdot \text{tmp})$
```

where out is the output of the sub-element. The output of each sub-element as well as the estimate of the intensities of the ambient light are restricted to the range $[0, 1]$. The percentage p was set to 0.0005 for the experiments which are described below. Step 1 of the algorithm averages the data received from the neighboring elements. Step 2 adds a small percentage p from the current pixel intensity. Step 3 distributes the new average pixel value to neighboring elements. And finally step 4 calculates the output intensity. Steps 1–4 are iterated indefinitely.

For a static image the estimate will converge to local space average color. The current intensity is faded into the average with a small percentage p. The existing average is reduced by the percentage $(1 - p)$. The initialization can be arbitrary. If the estimate of an element is lower than the estimate of its neighbors, it will be pulled up by the data received from the neighboring elements. If the estimate of an element is higher than the estimate of its neighbors it will be pulled down by the

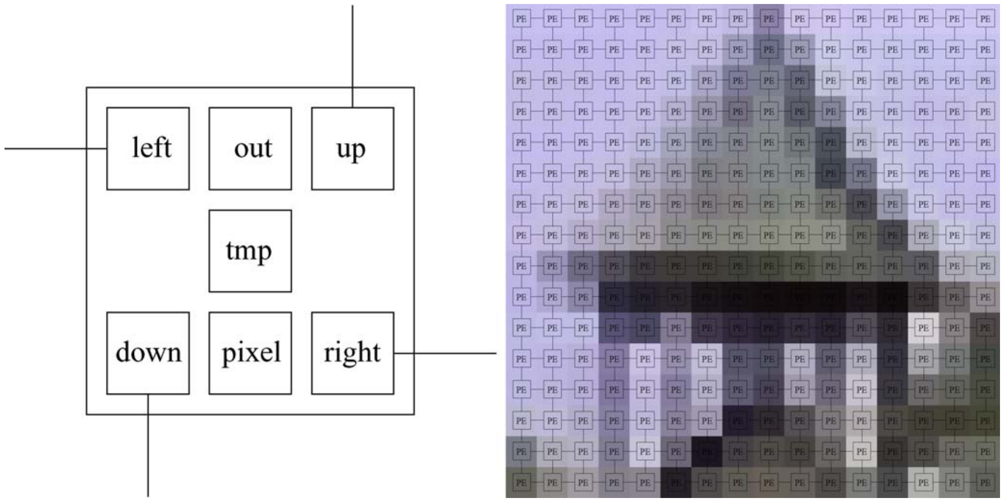

Fig. 1. A single processing element is shown on the left. The current average of pixel values is stored inside the element (tmp) and is also distributed to the left, right, up and down. This knowledge is continually updated. Each element has access to the red, green and blue channels of the viewed image (pixel). Using its knowledge about average pixel values, each element calculates the reflectances of its pixel (out). Each element only exchanges information locally, thus the individual elements may be combined easily to form a large $n \times n$ array (shown on the right).

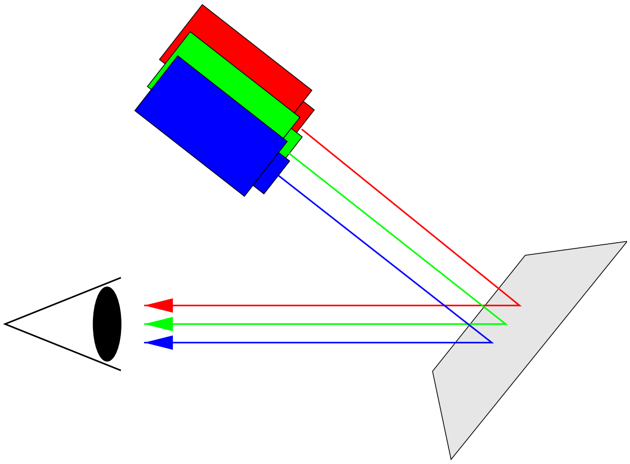

Fig. 2. Our algorithm assumes that we have three independent color bands red, green and blue.

data received from the neighboring elements. As the estimate is passed from one element to the next it will decrease by a factor p. Thus this factor determines the extent of the area for which local color constancy is calculated. Nearby elements have a larger impact on the estimate than elements which are further away.

The radius of influence can be calculated as follows. As the estimate is passed from one element to the next it will be reduced by a factor of $(1-p)$. After n steps it will have been reduced by $(1-p)^n$. We define the radius of influence as the number of steps after which the estimate will have been reduced to $\frac{1}{2}$. Then the radius of influence is defined by $r = \frac{-1}{\log_2(1-p)}$. Thus, the color stored at a distance r from the current element will only influence the current element by $\frac{p}{2}$. In practice one chooses p small enough such that space average at any point of the image receives information from all other points of the image. The actual value is arbitrary as long as p is small enough. For simulation purposes, one wants to use large values of p which reduces the time needed for the simulation. For our experiments we have used $p = 0.0005$. If we assume a maximum image size of 256 pixels, this tells us that the influence of a pixel at the border will be limited to $(1-p)^{256} = 88\%$ at the other side of the image.

The algorithm has a very simple structure and as such can be realized easily in hardware. All that is needed is to give one element access to the temporary storage of neighboring elements. Each sub-element only needs to store a single value, its estimate of the illuminant. Thus, the main advantage of this algorithm is that it uses only very little memory, and requires only a simple interconnection network. Each element has to be connected to its four immediate neighbors. Given the simple structure, the algorithm can be hardwired easily. Other algorithms are computationally much more expensive and often require global information. Of course it takes some time until the information has propagated through the network. But the time is only linear in the width (or height) of the image and not the size (the number of pixels) of the image. If we enlarge the image, the only parameter that needs adjustment is the parameter p, which determines the area for which local color constancy will be computed. We assume that propagation along the network is performed sufficiently fast in comparison to the time needed to grab an image.

Horn also proposed a parallel algorithm based on Land's Retinex theory [29]. He took the logarithm of the reflected intensity thereby splitting the product of reflectance and illuminant into a sum. Using an edge detection and thresholding operation he discarded slow changes of the illuminant. The original signal is restored

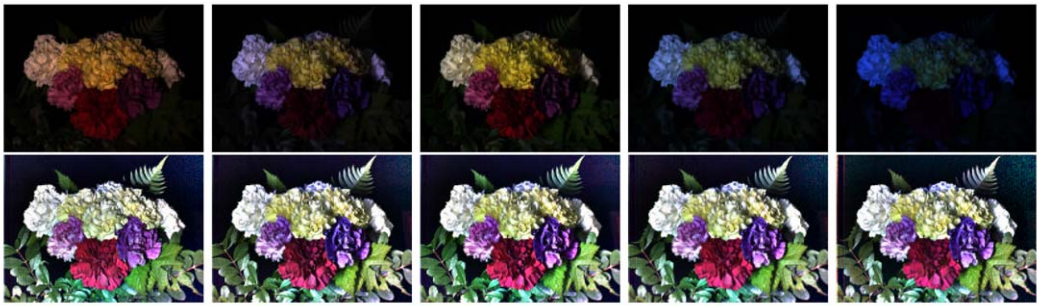

Fig. 3. A bouquet of flowers shown under five different illuminants: Sylvania 75 W halogen bulb, Sylvania Cool White fluorescent tube, Philips Ultralume fluorescent, Macbeth 5000K fluorescent, Macbeth 5000K fluorescent with a Roscolux 3202 full blue filter. The input images were taken from a library which was created by Funt et al. [5,21] to test color constancy algorithms. The bottom row shows the output of the algorithm.

using a resistive grid algorithm. A similar algorithm was proposed by Moore et al. [35] who computed a blurred image from the input image using a resistive grid. The blurred version was subtracted from the input image and then the output was rescaled. Moore et al. were able to implement this version of the retinex algorithm in VLSI.

4. Results

We tested our algorithm on a set of images[1] created by Barnard et al. [5] to test color constancy algorithms. From the available data sets we have used the one which was also used in the paper by Funt et al. [21][2]. This data set has the advantage that all objects are upright and not rotated. The newer data sets are especially suited for object recognition. Fig. 3 shows an image of a flower bouquet taken from the data set viewed under different illuminants. The top row shows the images which were used as input for our algorithm. The images are very dark, because they are linear and they were purposely under-exposed such that the number of clipped pixels was small. The bottom row of images shows the output images of our algorithm. The algorithm automatically performs a brightness correction for dark input images. For dark input images the average intensity is below 0.5 for the three bands red, green and blue. After gray-world color constancy has been applied this average is restored back to 0.5. Since we only calculate local space average color, brightness is also increased for very dark regions of the image.

Funt et al. [21] provide detailed response curves of the camera's sensors which were used to take the image. They also measured the spectral power distribution of the illuminants used. This data can be used to calculate the chromaticity of the illuminant. We compare the results of our algorithm to this data for the flower bouquet which was illuminated with a Macbeth 5000K fluorescent with a Roscolux 3202 full blue filter. Our algorithm estimates the illuminant using local space average color for every pixel. Even if a single illuminant is used, the estimate deviates slightly from pixel to pixel. Thus, we compare the estimated chromaticity of our algorithm to the correct chromaticity along a horizontal line of the image. This comparison is shown in Fig. 4.

Fig. 5 shows how the algorithm iteratively refines its estimate of the average pixel values. During the first iteration, each element uses the color of its pixel as an estimate of the illuminant. This estimate is then passed to neighboring elements where it will be updated. During the second iteration each element will have four estimates which are received from the neighboring elements. The local color is slowly faded into this value. As the estimate is again passed to its neighbors it will have decreased by a factor of $(1-p)$. From the snapshots we can see how the estimate is propagated through our network. With each iteration each element receives estimates from further away. This process continues until convergence. The number of steps until convergence for different values of p is shown in Fig. 6. Of course, if p is set to a large value, the radius of influence will be small and the algorithm will converge earlier. However, if the radius of influence is small then the assumption that the world on average is gray does not hold. In practice we have to set p such that this assumption is correct.

Results for additional objects from the same data set are shown in Fig. 7. Note that the algorithm is based on the gray-world assumption and works only if this assumption is valid. The assumption is valid as long as the scene is sufficiently complex which is the case for most of the images tested. If an object contains only a limited number of colors then this algorithm will not work. In this case, the average color in the image will not be gray. In this respect, our algorithm is similar to the performance of a human observer. A human observer is also unable to determine the correct color of a patch if the patch is viewed in isolation [38].

[1] http://www.cs.sfu.ca/~colour/data/index.html.

[2] http://www.cs.sfu.ca/~colour/image_db/index.html.

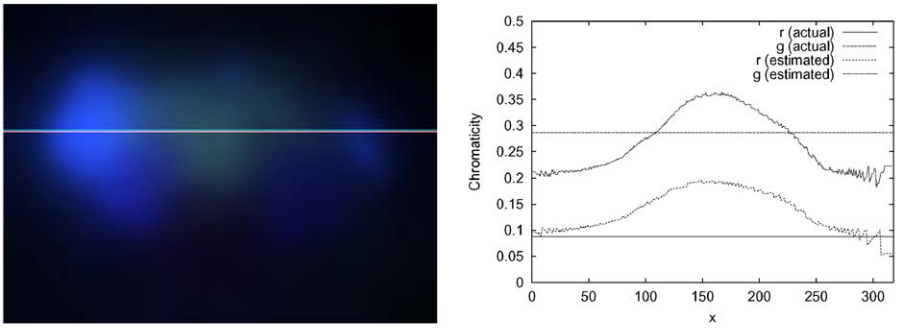

Fig. 4. The image on the left shows the illuminant estimated by our algorithm. From these data we extracted the chromaticities along a vertical line (marked in white). On the right the chromaticities are compared to the data provided by Funt et al. [21]. It is clear that the black background pixels lower the estimate.

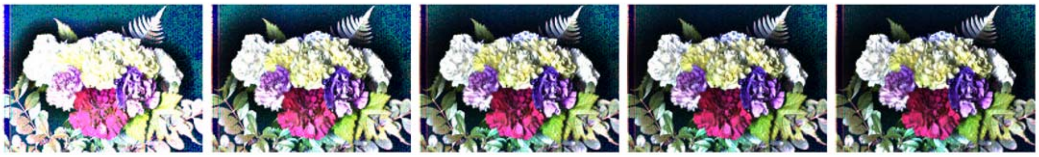

Fig. 5. The images show the output image after 1000, 2000, 3000, 4000, and 5000 iterations of the update rule. One can clearly see how the algorithm continually refines its estimate of the average pixel values and improves its output image. After approximately 5000 iterations, the output changes very little.

Our goal is to correctly restore the original colors of an image. The algorithm performs very well for most of the images from this data set. All images shown in this article can be viewed in color on the author's web page.[3]

5. Color constancy in the presence of multiple illuminants

We also tested our algorithm on images with two or more illuminants. Most research on color constancy assumes only a single illuminant. Algorithms which can also handle varying illumination include the retinex algorithm and variants [6,23,29,31,32,35]. Color constancy for scenes with varying illumination was also addressed by Barnard et al. [3]. Compared to existing algorithms our algorithm is much simpler to implement.

We have used the set of images created by Funt et al. [5,21] to artificially create images with varying illumination. Two images were taken for each object. One image was taken when the object was illuminated with a Sylvania 75 W halogen bulb, the other image was taken when the object was illuminated with a Macbeth 5000K fluorescent with a Roscolux 3202 full blue filter. Both images were merged to create a single image where the object is illuminated by both illuminants. A sigmoidal function was used to merge both images.

$$\alpha(x) = \frac{1}{1 + e^{-x/\sigma}}.$$

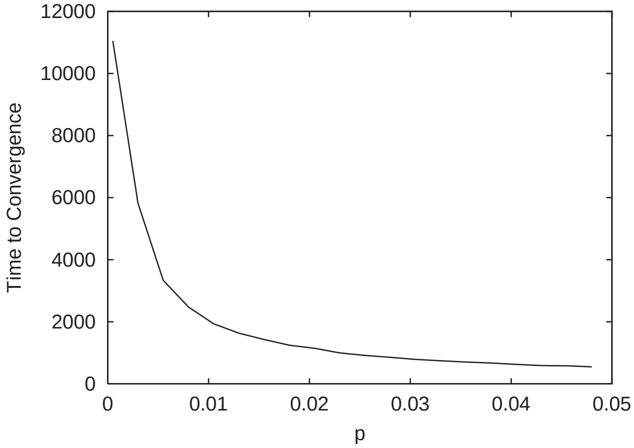

Fig. 6. Number of time steps until convergence for different values of p. Time to convergence was measured for the 318 × 234 image of the flower bouquet.

The shape of this function is shown for $\sigma = 0.1$ and 0.3 in Fig. 8. The parameter σ controls the steepness of the curve at the origin. The resulting image is calculated as

$$I_{\mathrm{C}}(x,y) = I_1(x,y)\alpha(x - x_0) + I_2(x,y)(1 - \alpha(x - x_0)),$$

where I_{C} is the combined image, I_1 is the first image (illuminant 1), I_2 is the second image (illuminant 2). This creates an image where pixels left of x_0 are mainly illuminated by illuminant 2 and pixels right of x_0 are mainly illuminated by illuminant 1. By artificially merging two images we have complete control over

[3] http://www2.informatik.uni-wuerzburg.de/staff/ebner/research/colorConstancy/colorConstancy.html..

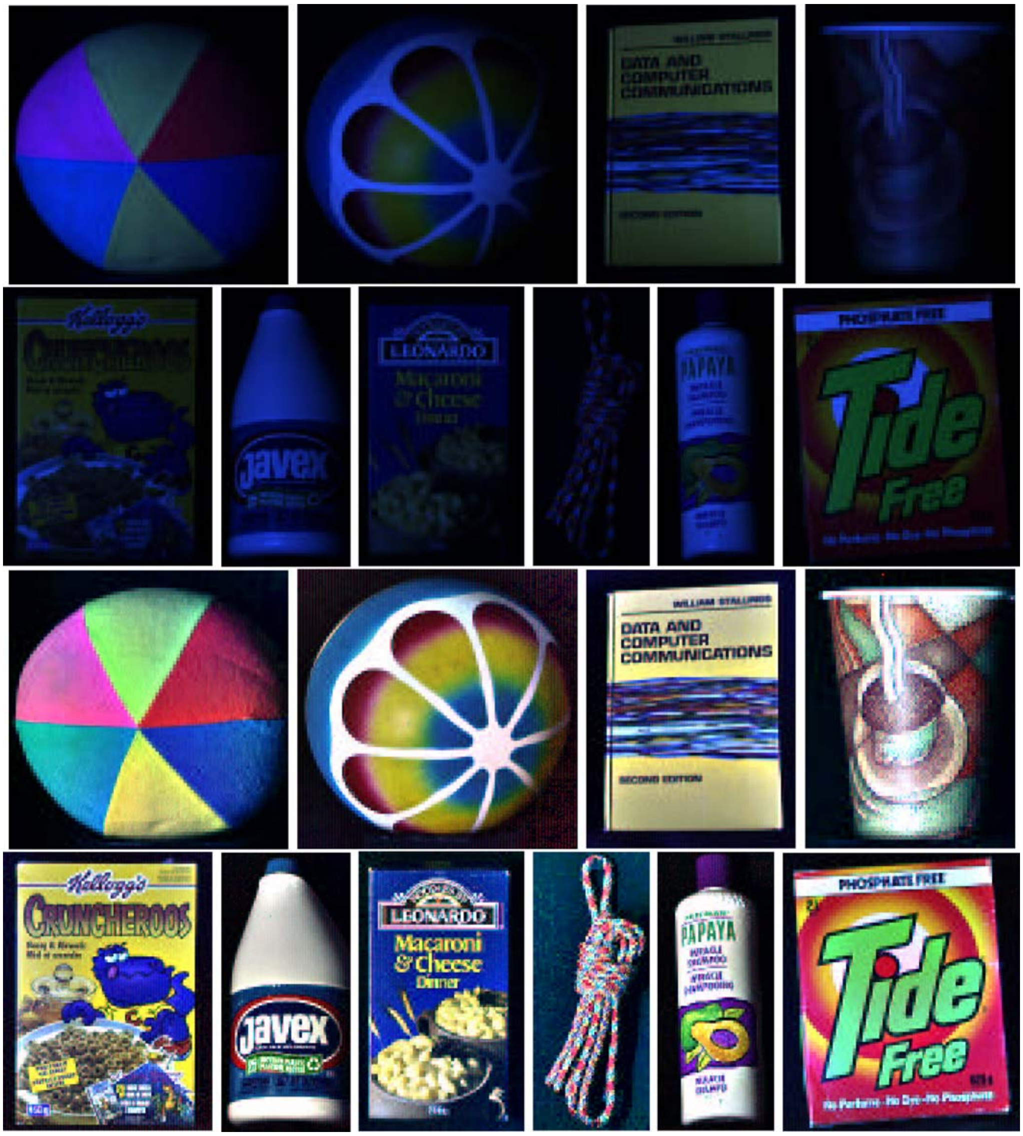

Fig. 7. Results for 10 additional objects taken from a library created by Funt et al. [5,21] which is used to test color constancy algorithms. The illuminant is a Macbeth 5000K fluorescent with a Roscolux 3202 full blue filter. The top two rows show the input images. The resulting images are shown in the two bottom rows.

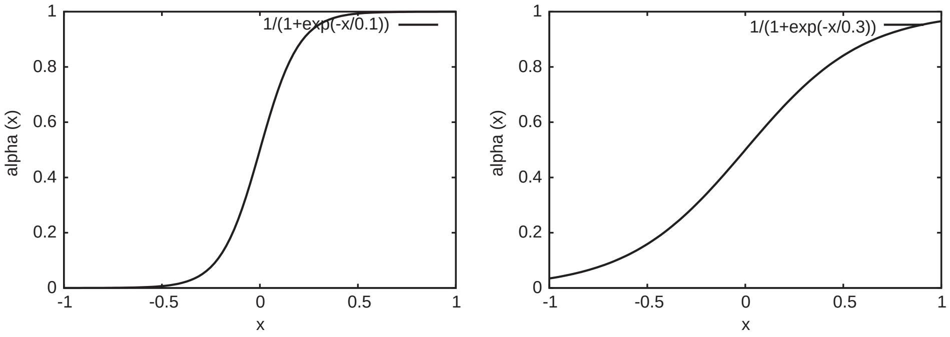

Fig. 8. Sigmoidal function for $\sigma = 0.1$ and 0.3.

how the illuminant varies over the image. We can create sharp changes of the illuminant as well as smooth changes.

Of course it is possible to also vary the shape of the change. We used an additional function f which models the shape of the transition from one illuminant to the next. This allows us to experiment with horizontal, vertical or diagonal transitions or even spotlight effects. The merged image is calculated as

$$I_C(x,y) = I_1(x,y)\alpha(f(x,y)) + I_2(x,y)(1-\alpha(f(x,y))).$$

If we choose $f(x,y) = x - x_0$ we get a vertical transition of the illuminant where x_0 defines the position of the transition. If we chose

$$f(x,y) = \sqrt{(x-x_1)^2 + (y-y_1)^2} - r$$

we get a spotlight effect. The object seems to be illuminated by a spotlight. The parameters (x_1, y_1)

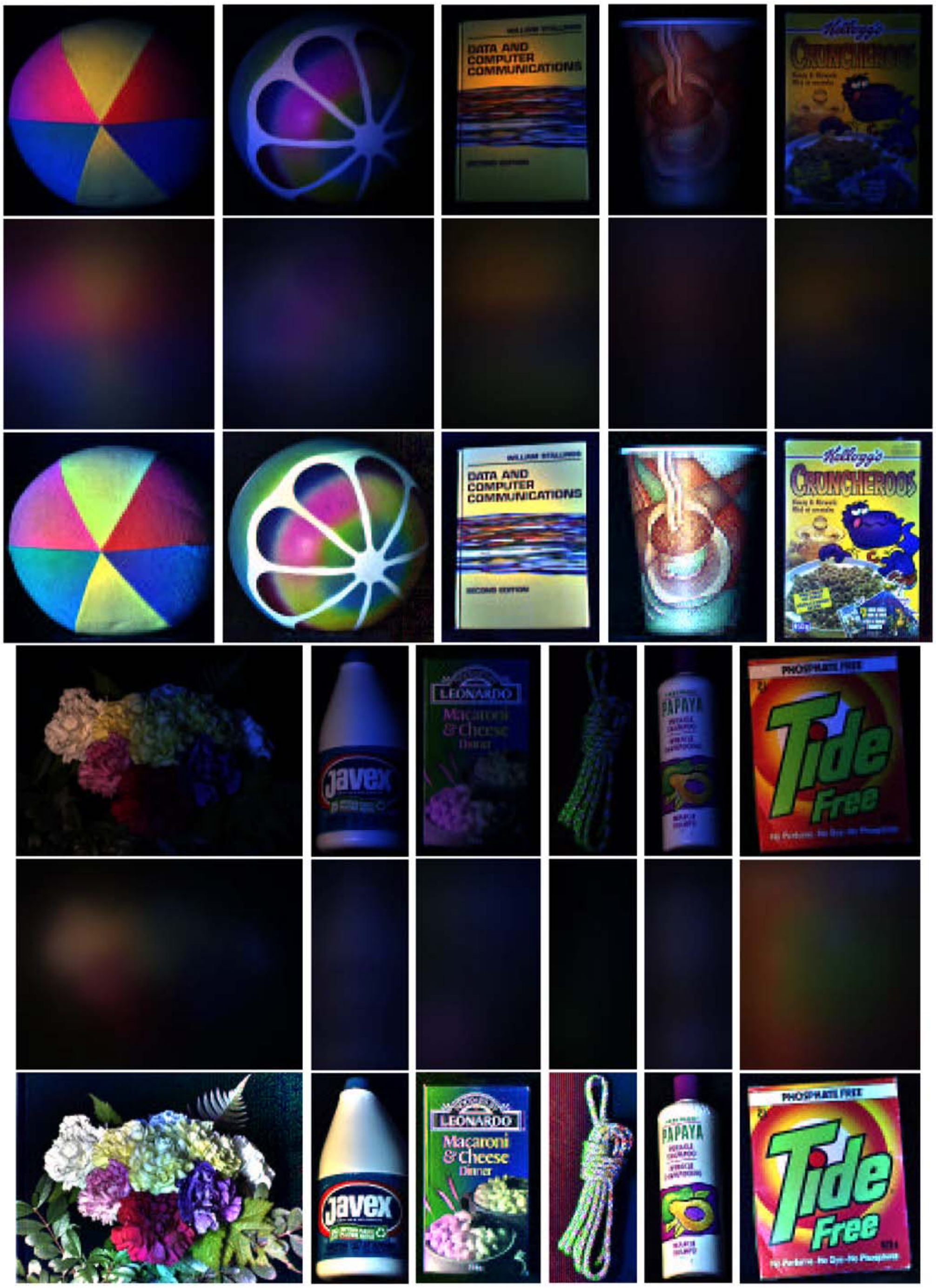

Fig. 9. Results for 11 objects taken from a library created by Funt et al. [5,21] which is used to test color constancy algorithms. The input image was created by merging two images. One was taken with a Sylvania 75 W halogen bulb, the other was taken with a Macbeth 5000K fluorescent with a Roscolux 3202 full blue filter. The upper three rows show the results for the simulated spotlight. The lower three rows show the results for the vertical transition function. The first and fourth rows show the input images. The second and fifth rows show space average color and the third and sixth rows show the output of our algorithm.

determine the position of the spotlight. The parameter r defines the size of the spot.

For our experiments we have set $x_0 = 0.4 \cdot \text{width}$, $x_1 = 0.4 \cdot \text{width}$, $y_1 = 0.4 \cdot \text{height}$, and $r = 0.3 \cdot \min\{\text{width}, \text{height}\}$ where width and height are the width and height of the input image. The results are shown in Fig. 9 for $\sigma = 10$. A circular boundary function was used for objects 1–5, and a vertical boundary function was used for objects 6–11. For each object the input image, the estimated illuminant and the output of our algorithm are shown. The algorithm is able to restore the original colors of the image even though a non-uniform illuminant is used. A comparison between the chromaticities of the actual illuminant and estimated illuminant for the flower bouquet is shown in Fig. 10.

Suppose, for the moment, that we have a vertical transition of the illuminant and an infinite image. Let the transition be linear. In our experiments the transition is approximately linear for large values of σ. Consider a single processing element. This processing element calculates local average color from four input values. One value from above, one from below, one from the left and one from the right. If we have a linear vertical transition of the illuminant, then the average color calculated by processing elements above and below will be equivalent to the average color which was calculated by the current processing element. The average color calculated by the processing element on the left will be a little smaller (or larger, depending on the order of the illuminants) than the average color calculated by the processing element on the right (Fig. 11). If we have a linear change of the illuminant then the differences will cancel each other out. Thus, after convergence local space average color will consist of vertical bands.

So far, we have only considered artificially merged images. Let us now look at the results for a real image with multiple illuminants. Fig. 12 shows an office with a large table. Some utensils were placed on the table. For the first image, the room was darkened and the left half was illuminated with a red light bulb. The right half was illuminated with a green light bulb. For the second image, the darkened room was illuminated with a blue light bulb on the left side and a red light bulb on the right side of the room. For the third image, blue office curtains were closed and the desk lamp was turned on. The image was taken on a sunny day. The sun falling through the curtains produced the blue background illumination. The desk lamp produced a spotlight on the table. All images were taken using a conventional camera. After development the images were digitized using a scanner. Images were gamma corrected with a factor of 2.2 before our algorithm was applied. A gamma correction of 1/2.2 was applied to the output images. The algorithm works best on the first two images. Results are not so good for the third image. This

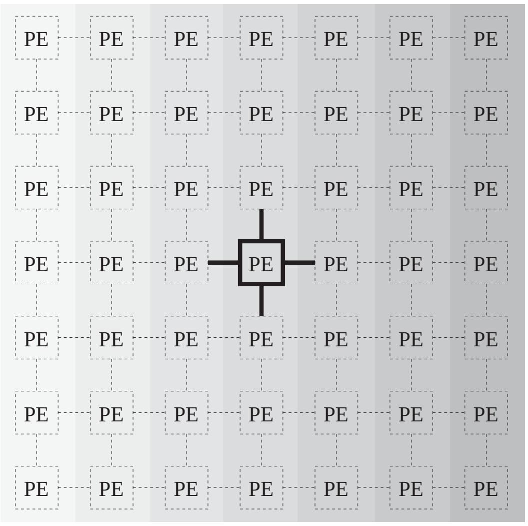

Fig. 11. Consider an element which is located at a position where the illuminant varies horizontally. If we have a linear change of the illuminant, then the values received from above and below will be correct. The value received from the left will be a little higher than its own estimate, while the value received from the right will be a little lower. However, both differences cancel each other out. This will produce vertical bands in the estimate of the illuminant.

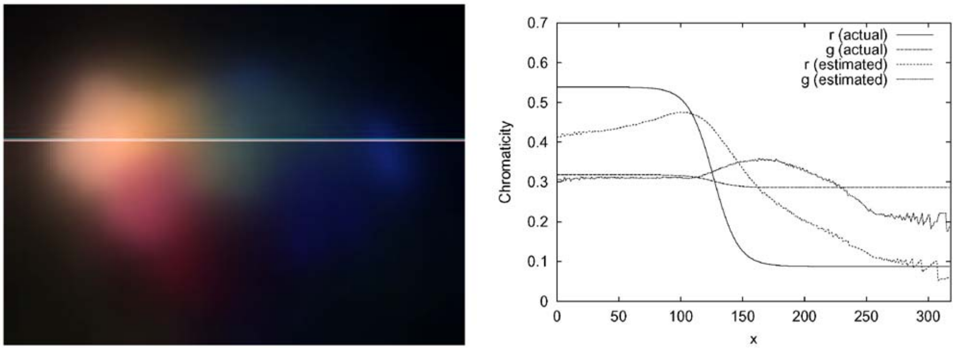

Fig. 10. The image on the left shows the illuminant estimated by our algorithm. From these data we extracted the chromaticities along a vertical line (marked in white). On the right, the chromaticities are compared to the actual chromaticities. The data provided by Funt et al. [21] were merged using the same sigmoidal function which was used to produce the input image.

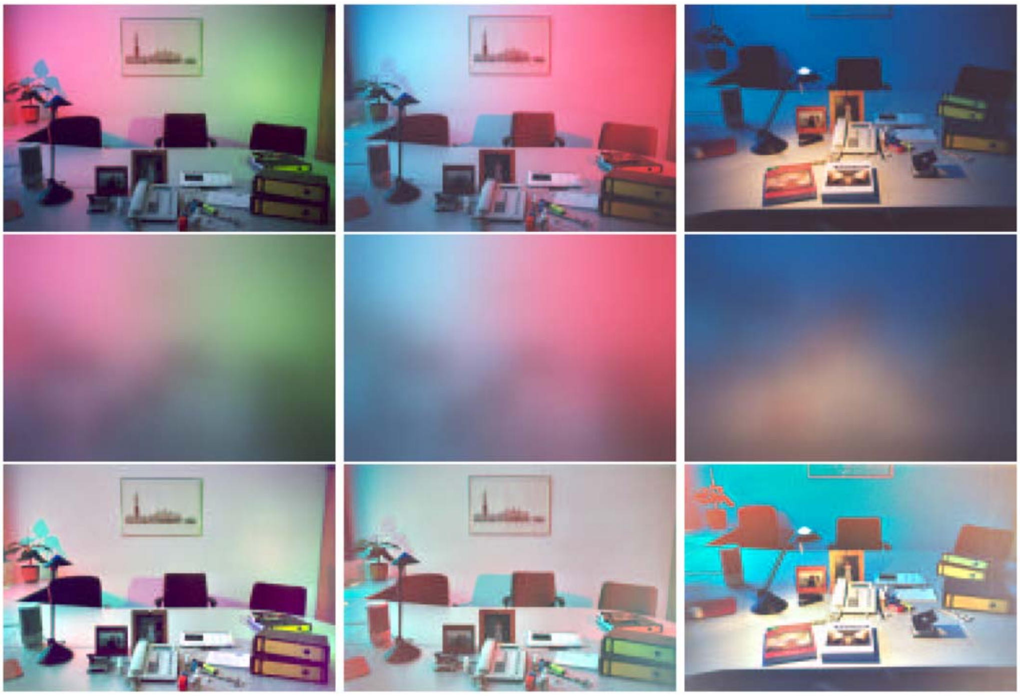

Fig. 12. An office scene illuminated by several illuminants. The first image shows an office where the left part is illuminated with a red light bulb and the right part is illuminated with a green light bulb. The second image shows the same office but this time the left is illuminated with a blue and the right half is illuminated with a red light bulb. The third image shows the office again, but this time the sun was shining through blue curtains (blue background illumination) and the desk lamp was turned on (spotlight on table). Below each image the estimate of local space average color is shown. The output image of our algorithm is shown in the last row.

image highlights the limitations of the algorithm. The algorithm works well for smooth transitions of the illuminant but fails for sharp transitions.

Why is this the case? For the gray-world assumption to hold, we must use small values for p. This means that local average color will be calculated over a large area. However, if we calculate local average color over a large area, then the assumption, that the transition of the illuminant is linear, no longer holds. We assume that the transition is relatively sharp then processing elements located at positions, where the transition is either linear or a single illuminant illuminates the scene, will calculate correct local average color. However, as we move closer to the transition, there will come a point where a processing element will also receive information from locations where the illuminant changes sharply.

At present we are working on the problem of color constancy for sharp transitions of the illuminant. Here the main problem is how to distinguish a sharp transition of the illuminant from a change of the object's color.

6. Conclusion

We have developed a parallel algorithm to solve the problem of color constancy. The algorithm was designed such that information is only exchanged locally, not globally. The algorithm is thus scalable to arbitrary sizes. The algorithm was tested on several images taken from a publicly available database to test color constancy algorithms.

The algorithm calculates local space average color and therefore works similar to the gray-world assumption. Due to its parallel mode of operation it is not equivalent to the gray-world assumption. The gray-world assumption works only for a single illuminant whereas our algorithm works also in the presence of multiple illuminants provided that the transition from one illuminant to the next is approximately linear.

Results are shown for a set of images where the amount of illumination coming from the illuminants was carefully controlled. Input images were created by merging two images which were identical except for the illuminant. The algorithm was also tested on a real-world example which also contains sharp transitions of the illuminant. This example shows the limitations of the current algorithm. At present we are working on an algorithm for color constancy which also works for sharp transitions of the illuminant.

References

[1] G.S. Almasi, A. Gottlieb, Highly Parallel Computing, The Benjamin/Cummings Publishing Company, Redwood City, CA, 1994.

[2] K. Barnard, F. Ciurea, B. Funt, Sensor sharpening for computational color constancy, J. Opt. Soc. Amer. A 18 (11) (November 2001) 2728–2743.

[3] K. Barnard, G. Finlayson, B. Funt, Color constancy for scenes with varying illumination, Comput. Vision Image Understand. 65 (2) (February 1997) 311–321.

[4] K. Barnard, L. Martin, B. Funt, Colour by correlation in a three dimensional colour space, in: Proceedings of the Sixth European Conference on Computer Vision, Dublin, Ireland, 2000, pp. 275–289.

[5] K. Barnard, L. Martin, B. Funt, A. Coath, A data set for color research, in: Color Research and Application 27 (3) (2002) 147–151.

[6] D.H. Brainard, B.A. Wandell, Analysis of the retinex theory of color vision, in: G.E. Healey, S.A. Shafer, L.B. Wolff (Eds.), Color, Jones and Bartlett Publishers, Boston, 1992, pp. 208–218.

[7] M. Brill, G. West, Contributions to the theory of invariance of color under the condition of varying illumination, J. Math. Biol. 11 (1981) 337–350.

[8] G. Buchsbaum, A spatial processor model for object colour perception, J. Franklin Inst. 310 (1) (July 1980) 337–350.

[9] V.C. Cardei, B. Funt, Committee-based color constancy, in: Proceedings of the IS&T/SID Seventh Color Imaging Conference: Color Science, Systems and Applications, Scottsdale, AZ, 1999, pp. 311–313.

[10] S.M. Courtney, L.H. Finkel, G. Buchsbaum, A multistage neural network for color constancy and color induction, IEEE Trans. Neural Networks 6 (4) (July 1995) 972–985.

[11] P.A. Dufort, C.J. Lumsden, Color categorization and color constancy in a neural network model of v4, Biol. Cybernet. 65 (1991) 293–303.

[12] M. D'Zmura, P. Lennie, Mechanisms of color constancy, in: G.E. Healey, S.A. Shafer, L.B. Wolff (Eds.), Color, Jones and Bartlett Publishers, Boston, 1992, pp. 224–234.

[13] M. Ebner, Evolving color constancy for an artificial retina, in: J. Miller, M. Tomassini, P.L. Lanzi, C. Ryan, A.G.B. Tettamanzi, W.B. Langdon (Eds.), Genetic Programming: Proceedings of the Fourth European Conference, EuroGP 2001, Lake Como, Italy, April 18–20, Springer, Berlin, 2001, pp. 11–22.

[14] M. Ebner, A parallel algorithm for color constancy, Technical Report 296, Universität Würzburg, Lehrstuhl für Informatik II, Am Hubland, Würzburg 97074, Germany, April 2002.

[15] G.D. Finlayson, Color in perspective, IEEE Trans. Pattern Anal. Mach. Intell. 18 (10) (October 1996) 1034–1038.

[16] G.D. Finlayson, M.S. Drew, B.V. Funt, Spectral sharpening: sensor transformations for improved color constancy, J. Opt. Soc. Amer. A 11 (4) (April 1994) 1553–1563.

[17] G.D. Finlayson, P.M. Hubel, S. Hordley, Color by correlation, in: Proceedings of IS&T/SID. The Fifth Color Imaging Conference: Color Science, Systems, and Applications, November 17–20, The Radisson Resort, Scottsdale, AZ, 1997, pp. 6–11.

[18] G.D. Finlayson, B. Schiele, J.L. Crowley, Comprehensive colour image normalization, in: Fifth European Conference on Computer Vision (ECCV '98), Freiburg, Germany, Springer, Berlin, 1998, 445–459.

[19] D.A. Forsyth, A novel approach to colour constancy, in: Second International Conference on Computer Vision, Tampa, FL, December 5–8, IEEE Press, New York, 1988, pp. 9–18.

[20] D.A. Forsyth, A novel algorithm for color constancy, in: G.E. Healey, S.A. Shafer, L.B. Wolff (Eds.), Color, Jones and Bartlett Publishers, Boston, 1992, pp. 241–271.

[21] B. Funt, K. Barnard, L. Martin, Is colour constancy good enough? in: Fifth European Conference on Computer Vision (ECCV '98), Freiburg, Germany, 1998, pp. 445–459.

[22] B. Funt, V. Cardei, K. Barnard, Learning color constancy, in: Proceedings of the IS&T/SID Fourth Color Imaging Conference, Scottsdale, 19–22 November 1996, pp. 58–60.

[23] B.V. Funt, M.S. Drew, Color constancy computation in near-mondrian scenes using a finite dimensional linear model, in: Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition, Computer Society Press, Ann Arbor, MD, 5–9 June 1988, pp. 544–549.

[24] B.V. Funt, M.S. Drew, J. Ho, Color constancy from mutual reflection, Internat. J. Comput. Vision 6 (1) (1991) 5–24.

[25] R. Gershon, A.D. Jepson, J.K. Tsotsos, From [r,g,b] to surface reflectance: computing color constant descriptors in images, in: Proceedings of the Tenth International Joint Conference on Artificial Intelligence, Milan, Italy, Vol. 2, 1987, pp. 755–758.

[26] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Addison-Wesley, Reading, MA, 1992.

[27] J. Herault, A model of colour processing in the retina of vertebrates: from photoreceptors to colour opposition and colour constancy phenomena, Neurocomputing 12 (1996) 113–129.

[28] J. Ho, B.V. Funt, M.S. Drew, Separating a color signal into illumination and surface reflectance components: theory and applications, in: G.E. Healey, S.A. Shafer, L.B. Wolff (Eds.), Color, Jones and Bartlett Publishers, Boston, 1992, pp. 272–283.

[29] B.K.P. Horn, Robot Vision, The MIT Press, Cambridge, MA, 1986.

[30] R. Jain, R. Kasturi, B.G. Schunck, Machine Vision, McGraw-Hill, New York, 1995.

[31] E.H. Land, The retinex theory of colour vision, Proc. Roy. Inst. Great Britain 47 (1974) 23–58.

[32] E.H. Land, An alternative technique for the computation of the designator in the retinex theory of color vision, Proc. Natl. Acad. Sci. USA 83 (May 1986) 3078–3080.

[33] K.J. Linnell, D.H. Foster, Space-average scene colour used to extract illuminant information, in: C. Dickinson, I. Murray, D. Carden (Eds.), John Dalton's Colour Vision Legacy. Selected Proceedings of the International Conference, Taylor & Francis, London, 1997, pp. 501–509.

[34] L.T. Maloney, B.A. Wandell, Color constancy: a method for recovering surface spectral reflectance, J. Opt. Soc. Amer. A3 3 (1) (January 1986) 29–33.

[35] A. Moore, J. Allman, R.M. Goodman, A real-time neural system for color constancy, IEEE Trans. Neural Networks 2 (2) (March 1991) 237–247.

[36] C.L. Novak, S.A. Shafer, Supervised color constancy for machine vision, in: G.E. Healey, S.A. Shafer, L.B. Wolff (Eds.), Color, Jones and Bartlett Publishers, Boston, 1992, pp. 284–299.

[37] S. Usui, S. Nakauchi, A neurocomputational model for colour constancy, in: C. Dickinson, I. Murray, D. Carden (Eds.), John Dalton's Colour Vision Legacy. Selected Proceedings of the International Conference, Taylor & Francis, London, 1997, pp. 475–482.

[38] S. Zeki, A Vision of the Brain, Blackwell Science, Oxford, 1993.

Marc Ebner was born in Stuttgart, Germany, in 1969. He received the M.S. degree in computer science from New York University, NY, in 1994, the degree Dipl.-Inform. from the Universität Stuttgart, Germany, in 1996 and the degree Dr. rer. nat. from the Universität Tübingen, Germany, in 1999. At present he is a research assistant at the Universität Würzburg, Germany. His research interests include computer vision, biologically inspired systems, evolutionary algorithms, evolutionary robotics, and artificial life.