6

Algorithms for Color Constancy under Uniform Illumination

Numerous algorithms have been proposed for color constancy. We now describe in detail how these algorithms work. We also see how the algorithms perform in practice. The performance of the algorithms will be shown on two sample images. One image shows a scene with a uniform illumination and the other shows a scene with a nonuniform illumination.

The two sample images are shown in Figure 6.1. The images were taken with a Canon 10D using the sRGB color space. The first image (a) shows a table with plates, coffee cups, spoons, and so on. The image looks very yellow because the room was illuminated by a yellow illuminant. The second image (b) shows an office scene with a desk and several utensils on top of the desk. The image was taken on a bright sunny day. The blue window curtains were closed and the desk lamp was switched on. The blue window curtains created the blue background illumination. The spotlight effect on the table was created by the desk lamp.

In this chapter, we will assume that the color is uniform across the scene. The intensity $\mathbf{I}$ measured by a sensor at position $\mathbf{x}_I$ can be modeled as

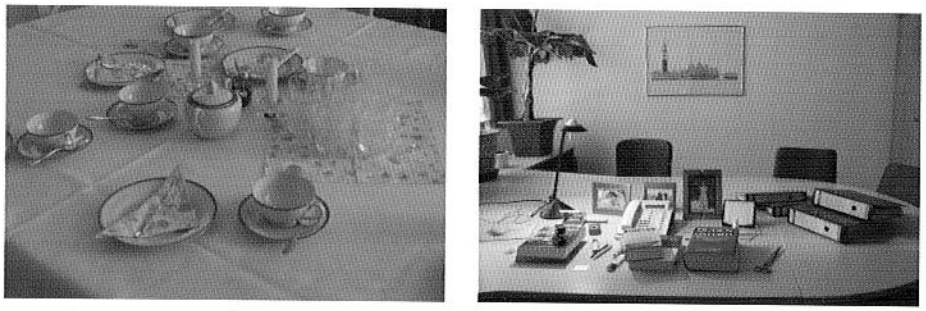
$$\mathbf{I}(\mathbf{x}_I) = G(\mathbf{x}_{\text{Obj}}) \int R(\lambda, \mathbf{x}_{\text{Obj}}) L(\lambda) \mathbf{S}(\lambda)\, d\lambda \tag{6.1}$$

where $G(\mathbf{x}_{\text{Obj}})$ is a scaling factor due to the geometry of the patch at position $\mathbf{x}_{\text{Obj}}$, $R(\lambda, \mathbf{x}_{\text{Obj}})$ denotes the reflectance at position $\mathbf{x}_{\text{Obj}}$, $L(\lambda)$ is the radiance given off by the light source, and $\mathbf{S}(\lambda)$ describes the sensitivity of the sensors.

For all of the following algorithms, we will assume that the response functions of the sensors are very narrow-band, i.e. they can be approximated by delta functions. Let λ_i with $i \in \{r, g, b\}$ be the wavelengths to which the sensors respond. We will now denote the sensor coordinates by (x, y). The intensity measured by the sensor at position (x, y) is then given by

$$I_i(x, y) = G(x, y) R_i(x, y) L_i \tag{6.2}$$

where $G(x, y)$ is a factor that depends on the scene geometry at the corresponding object position, $R_i(x, y)$ is the reflectance for wavelength λ_i, and L_i is the irradiance at wavelength



(a) (b)

Figure 6.1 Two sample images. The scene in (a) shows a table illuminated by a yellow illuminant. The scene in (b) shows a desk illuminated by sunlight falling through a blue curtain. This creates a blue background illumination. The lamp on the desk was also switched on.

λ_i. We have already derived this equation in Section 3.6. The algorithms discussed in this chapter assume that the illuminant is uniform across the entire image, i.e. the irradiance does not depend on the coordinates (x, y). Algorithms that do not make these assumptions are discussed in the next chapter.

6.1 White Patch Retinex

The white patch retinex algorithm is basically just a simplified version of the retinex algorithm (Cardei and Funt 1999; Funt et al. 1998, 1996; Land and McCann 1971), which is described in the next chapter. The retinex algorithm relies on having a bright patch somewhere in the image. The idea is that, if there is a white patch in the scene, then this patch reflects the maximum light possible for each band. This will be the color of the illuminant, i.e. if $R_i(x, y) = 1$ for all $i \in \{r, g, b\}$ and $G(x, y) = 1$, then

$$I_i(x, y) = L_i. \tag{6.3}$$

If one assumes a linear relationship between the response of the sensor and pixel colors, i.e. $c_i(x, y) = I_i(x, y)$, and one also assumes that the sensor's response characteristic is similar to delta functions, then the light illuminating the scene simply scales the product of the geometry term G and the reflectance R_i of the object.

$$c_i(x, y) = G(x, y)R_i(x, y)L_i \tag{6.4}$$

Therefore, we can rescale all color bands once we have located such a bright patch. In practice, one does not look for a white patch but looks for the maximum intensity of each color channel. Let $L_{i,\max}$ be the maximum of each band over all pixels.

$$L_{i,\max} = \max_{x,y}\{c_i(x, y)\} \tag{6.5}$$

(a)

(b)

Figure 6.2 The white patch retinex algorithm is able to perform some color adjustments for the image shown in (a) but the image still looks very yellow. The white patch retinex algorithm does not work very well for the image shown in (b) because of the nonuniform illuminant.

This maximum is then used to scale each color band of the pixels back to the range [0, max]

$$o_i(x, y) = \frac{c_i(x, y)}{L_{i,\max}} = G(x, y)R_i(x, y) \tag{6.6}$$

where $\mathbf{o}(x, y) = [o_r(x, y), o_g(x, y), o_b(x, y)]^T$ is the color of the output pixel.

Figure 6.2 shows the results achieved with the white patch retinex algorithm. The problem with the white patch retinex algorithm is that it assumes that a single uniform illuminant illuminates the scene. Therefore, it does not work if we have a nonuniform illumination as is the case for the image shown in (b). It also performs poorly on the image shown in (a). This image has some bright specularity in the glassware on the table. The resulting image is still too yellow.

A drawback of this simple version of the white patch retinex algorithm is that a single bright pixel can lead to a bad estimate of the illuminant. If we have a highlight in the image caused by an object that does not reflect the color of the illuminant uniformly, then the estimate will not be equivalent to the actual color of the illuminant. Noise in the image will also be a problem. The white patch retinex algorithm is also highly susceptible to clipped pixels (Funt et al. 1998). If one or more color channels are clipped, then the color of the illuminant cannot be reliably estimated from the brightest pixel.

The white patch retinex algorithm can be made more robust by computing a histogram H_i for each color channel i. The histogram tells us how many image pixels have a particular intensity for a given color channel. Let n_b be the number of buckets of the histogram and let $H_i(j)$ be the number of pixels of color channel i that have intensity j. Instead of choosing the pixel with the maximum intensity for each color channel, one can choose the intensity such that all pixels with intensity higher than the one chosen account for some percentage of the total number of pixels. This method is also used as an automatic white balance by some scanning software. Finlayson et al. (2002) also use it in their method for shadow removal. Let p be the percentage, i.e. 1% or a similar small value, and n be the total number of pixels in the image. Let $c_i(j)$ be the intensity of color channel i represented by

bucket j of the histogram H_i. Then the estimate of the illuminant is given by

$$L_i = c_i(j_i) \tag{6.7}$$

with j_i chosen such that

$$pn \leq \sum_{k=j_i}^{n_b} H_i(k) \qquad \text{and} \qquad pn \geq \sum_{k=j_i+1}^{n_b} H_i(k). \tag{6.8}$$

Results for this algorithm are shown in Figure 6.3. The image of the coffee table looks much better now.

6.2 The Gray World Assumption

The gray world assumption was proposed by Buchsbaum (1980). It estimates the illuminant using the average color of the pixels. That the illuminant could be estimated by computing some kind of average of the light received by the observer was known for a long time and was also suggested by Land (see Judd 1960). However, Buchsbaum was the first to formalize the method. The gray world assumption is probably one of the best-known algorithms for color constancy. Many algorithms have been proposed, which use the gray world assumption in one way or another (Ebner 2002, 2003a,c, 2004c,d; Finlayson et al. 1998; Gershon et al. 1987; Moore et al. 1991; Paulus et al. 1998; Pomierski and Groß 1995; Rahman et al. 1999; Tominaga 1991). These algorithms are all based on the assumption that, on average, the world is gray. Buchsbaum's algorithm estimates the illuminant by assuming that a certain standard spatial spectral average exists for the total visual field. This average is used to estimate the illuminant, which is then used to estimate the reflectances. Results were only shown for simulated data. The derivation of the gray world assumption given here differs from the one given by Buchsbaum. Buchsbaum considers overlapping response characteristics of the sensor array. We assume nonoverlapping response characteristics because little may be gained by using a more general transform (Barnard et al. 2001; Finlayson et al. 1994b). In addition, we also include geometry information in the reflection model whereas Buchsbaum used a simple reflection model without any geometry information.

From the theory of color image formation, we have seen that the intensity $I_i(x, y)$ measured by a sensor i with $i \in \{r, g, b\}$ at position (x, y) on the sensor array can be approximated by

$$I_i(x, y) = G(x, y)R_i(x, y)L_i(x, y) \tag{6.9}$$

if we assume that the sensor sees a surface that reflects light equally in all directions, i.e. it is a Lambertian surface. Each sensor determines the intensity of the light of a particular wavelength λ_i. Here, $G(x, y)$ is a factor that depends on scene geometry at the corresponding object point that is shown at position (x, y) and the type of lighting model used, $R_i(x, y)$ is the amount of light reflected at the corresponding object position (x, y) for wavelength λ_i, and $L_i(x, y)$ is the intensity of the illuminant at the corresponding object position.

For the derivation of this equation, we have also assumed ideal sensors that only respond to light of a single wavelength, i.e. the sensor can be described by a delta function. This is an

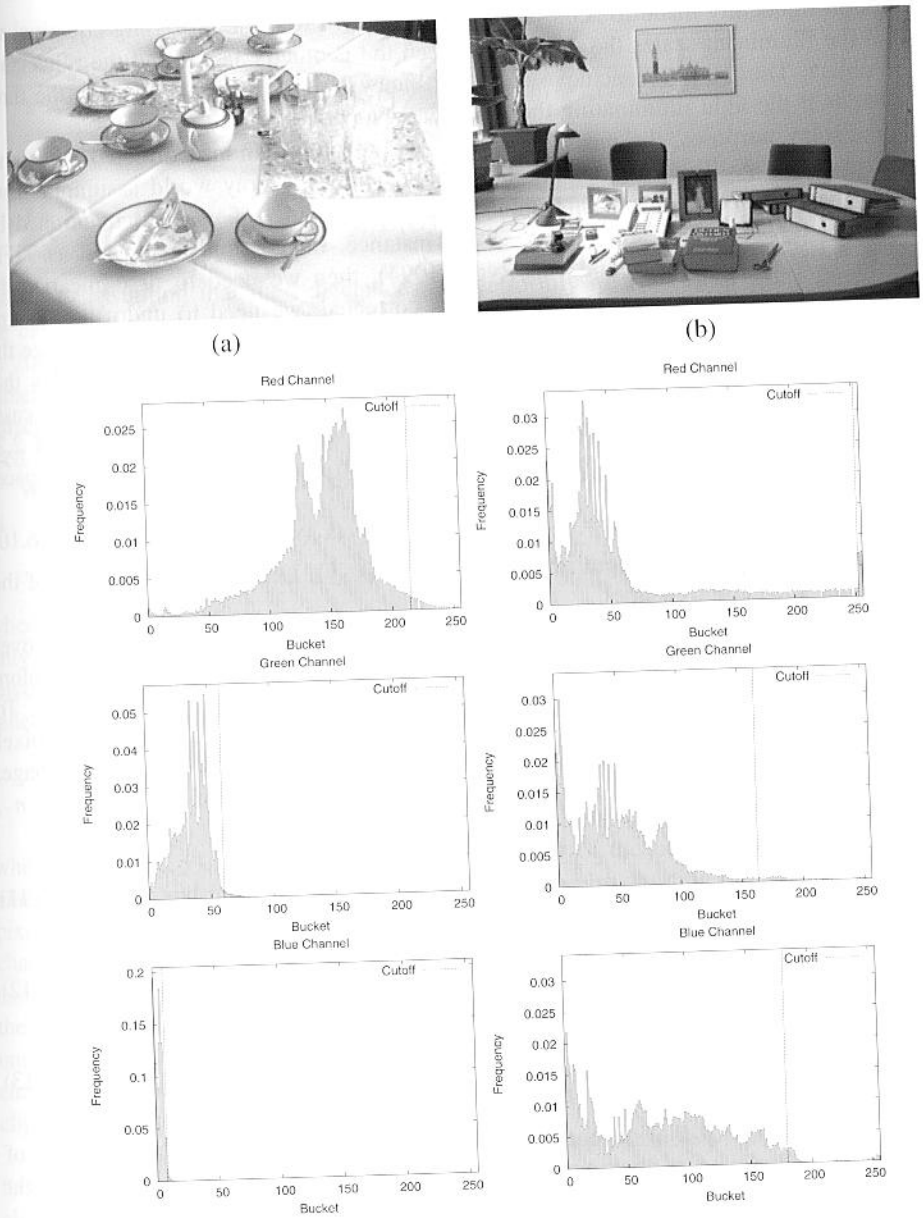


Figure 6.3 Results for the white patch retinex algorithm where histograms were used to find a white patch. The histograms for all the three color bands of the input image are also shown. The cutoff value is marked with a vertical line. All the three cutoff values represent an estimate of the illuminant.

assumption frequently made to achieve color constancy. From the preceding equation, we see that the illuminant scales the product between the geometry factor and the reflectance of the object. Thus, one can achieve color constancy by independently scaling the three color bands if the light illuminating the scene was known.

For display purposes, the measured intensities are often gamma corrected and transformed to the range [0, 1] or [0, 255]. In order to apply the gray world assumption, we need to linearize pixel colors. This point is very important and was not mentioned by Buchsbaum. If we process stored images, for instance, images that are stored as TIFF, JPEG, or PPM files (Murray and van Ryper 1994), then we need to first linearize the intensity values. If the pixel colors are gamma corrected, we need to undo this gamma correction. If the colors are gamma corrected with a factor of 1/2.2, then we linearize the pixel colors by applying a gamma correction with a factor of 2.2. The problem is that quite often the gamma factor of the original gamma correction is not known. In this case, it is assumed that images use the sRGB color space, which assumes a gamma of 2.2. See Section 4.5 for details on how to transform a given image to linear intensity space. Let

$$\mathbf{c}(x, y) = [c_r(x, y), c_g(x, y), c_b(x, y)]^T \tag{6.10}$$

be the linearized color of the input image at position (x, y). Let $[0, 1]$ be the range of the intensities for each color channel.

If we assume that the colors of the objects in view are uniformly distributed over the entire color range and we have a sufficient number of objects with different colors in the scene, then the average color computed for each channel will be close to $\frac{1}{2}$. To show this, we assume a linear mapping between sensor measurements and image pixel colors, i.e. $c_i(x, y) = I_i(x, y)$, and an illuminant that is uniform across the entire image, i.e. $L_i(x, y) = L_i$. In this case, space average color $\mathbf{a}$ of an image of size $n = n_x \times n_y$, where n_x is the width and n_y is the height of the image, is given by

$$a_i = \frac{1}{n} \sum_{x,y} c_i(x, y) \tag{6.11}$$

$$= \frac{1}{n} \sum_{x,y} G(x, y) R_i(x, y) L_i \tag{6.12}$$

$$= L_i \frac{1}{n} \sum_{x,y} G(x, y) R_i(x, y) \tag{6.13}$$

where $G(x, y)$ is a factor that depends on scene geometry, $R_i(x, y)$ is the reflectance of the object point displayed at position (x, y) in the image, L_i is the intensity of the light that illuminates the scene, and the index i denotes the corresponding color channel.

Let $E[GR_i]$ be the expected value of the geometry factor G multiplied by the reflectance R_i. Both can be considered as independent random variables, as there is no correlation between the shape and the color of an object. Let us assume that the reflectances are uniformly distributed, i.e. many different colors are present in the scene and each color is equally likely. Therefore, the reflectance can be considered to be a random variable drawn from the range $[0, 1]$. We obtain (Johnson and Bhattacharyya 2001)

$$E[GR_i] = E[G]E[R_i] = E[G]\left(\int_0^1 x\, dx\right) = E[G]\frac{1}{2}. \tag{6.14}$$

For large n, we have

$$a_i = L_i \frac{1}{n} \sum_{x,y} G(x, y) R_i(x, y) \tag{6.15}$$

$$\approx L_i E[G R_i] \tag{6.16}$$

$$= L_i E[G] \frac{1}{2}. \tag{6.17}$$

Instead of assuming that the reflectances are uniformly distributed over the range [0, 1], we can also use the actual distribution of reflectances to compute $E[R_i]$ (Barnard et al. 2002a). In this case, we also need to know the shape of the response curves of the camera to compute the expected value. This value would then depend on the set of reflectances chosen for the surrounding and would also depend on the type of camera used.

We now see that we can use space average color to estimate the color of the illuminant as

$$L_i \approx \frac{2}{E[G]} a_i = f a_i \tag{6.18}$$

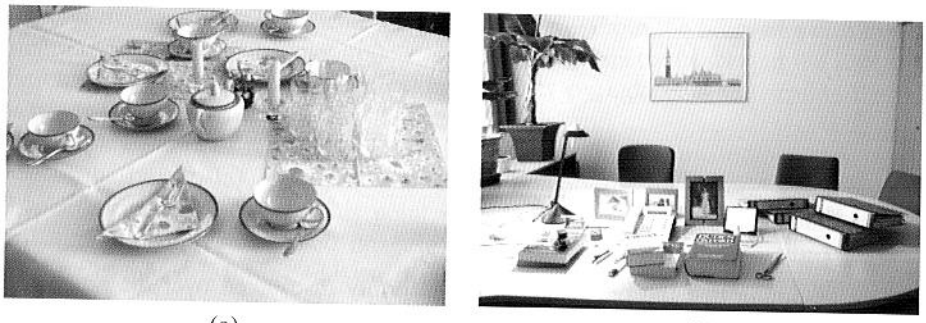
where $f = \frac{2}{E[G]}$ is a factor that depends on the scene viewed. Note that this derivation is only valid if the relationship between measured sensor values and pixel colors is linear. Given the color of the illuminant, we can estimate the combined geometry factor and the reflectance of the object. With $c_i(x, y) = G(x, y) R_i(x, y) L_i$, we have,

$$o_i(x, y) = \frac{c_i(x, y)}{L_i} \approx \frac{c_i(x, y)}{f a_i} = G(x, y) R_i(x, y) \tag{6.19}$$

where $\mathbf{o} = [o_r(x, y), o_g(x, y), o_b(x, y)]^T$ is the color of the output pixel. Thus, the combined geometry and reflectance factor can be estimated by dividing the color of the current pixel by the product of f and space average color. The factor f only scales all color channels equally and affects only the intensity of the colors. It can be set as $f = 2$. This assumes that $E[G] = 1$, i.e. there is a perpendicular orientation between the object and the camera. The factor f can also be estimated directly from the image. In this case, one first rescales each channel by dividing the intensity by the average value of the channel. Next all channels are rescaled equally such that, say, only 1% of all pixels are clipped.

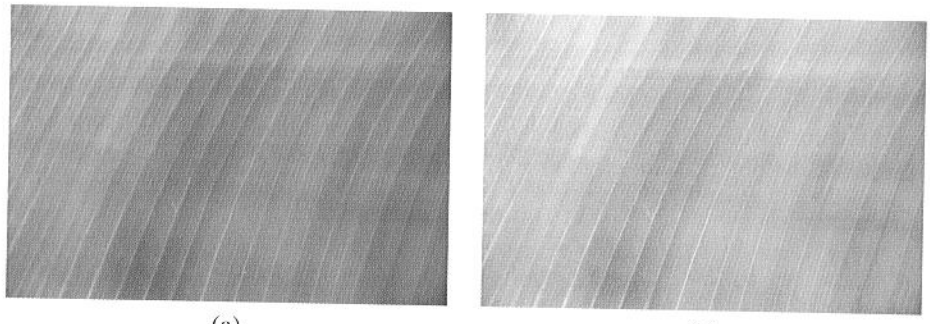
Figure 6.4 shows the results achieved with the gray world assumption. In practice, we have found the gray world assumption to produce better results than the white patch retinex algorithm. The gray world assumption is based on the average of a large number of pixels. In this respect, it is much more robust than the white patch retinex, which is only based on the maximum pixel value. If one only has a single pixel that is very bright, then the white patch retinex algorithm will probably produce incorrect results. This holds especially if shiny objects are in the scene, which reflect all of the incident light, which may result in clipped pixels.

The gray world assumption works nicely if we only have a single illuminant. However, if we have multiple illuminants, then the gray world assumption does not work. We can see this in image (b) of Figure 6.4. This is not surprising, as one of the assumptions was that we



(a) (b)

Figure 6.4 Gray world assumption. The gray world assumption produces nice results if we only have a single illuminant (a). If we have multiple illuminants, then the gray world assumption does not work very well (b).



(a) (b)

Figure 6.5 The gray world assumption will fail to produce correct colors if sufficiently large numbers of colors are not present in the scene. A leaf from a banana plant is shown in (a). The image in (b) shows the output image.

have a uniform illuminant. The gray world assumption requires that there be a sufficiently large number of different colors in the image. If this is not the case, then the gray world assumption will not work. Figure 6.5(a) shows a close-up of a leaf from a banana plant. The average of the channels for this image is [0.227827, 0.339494, 0.049392]. Thus, the gray world assumption will increase the influence of the blue channel to a great extent. The output image is shown in (b). Clearly, this is not what is desired.

The gray world assumption as well as the white patch retinex algorithm are used frequently for automatic white balance. The popular draw utility written by Coffin (2004) scales each color channel using the average as an automatic white-balance option. After rescaling, the white point is set at the 99th percentile. In other words, all channels are scaled equally such that only the top 1% of all pixels are clipped. This assumes that there are only a few highlights.

Another assumption is that the reflectances of the image are uniformly distributed over the range [0,1]. This assumption may not be correct in practice. Suppose that we have one uniformly colored object in the image and that the object covers most of the image pixels. In this case, space average color will be close to the color of the object irrespective of the background. Therefore, it may make sense to segment the image before performing the averaging operation. Gershon et al. (1987) suggest that the input image may be segmented into different regions. Let n_r be the number of different regions and let $\mathbf{a}(R_j) = [a_r(R_j), a_g(R_j), a_b(R_j)]^T$ be the average color of region $j \in \{1, ..., n_r\}$. We now calculate the average color by looping over the unique regions:

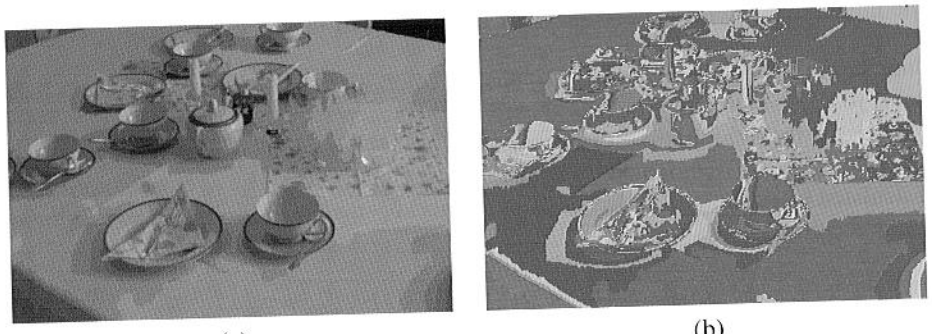
$$a_i = \frac{1}{n_r} \sum_{j=1}^{n_r} a_i(R_j) \tag{6.20}$$

This space average color is then used to calculate the reflectances.

$$L_i \approx f a_i \tag{6.21}$$

Obviously, if the input image is segmented before calculating the average, then each region contributes only once to the average. The result is independent of the number of pixels the different areas cover. Figure 6.6 shows a segmented image. For the image shown in (a), each region was colored using the average color of the pixels that belong to the particular region. To show the different regions of the segmented image better, the regions are shown with random colors in (b). Figure 6.7 shows the result when segmentation is used to compute the average.

Suppose that we have one large object that is covered by a second object in front of it. If both objects have a different color, a segmentation algorithm may segment the image into three regions even though there are only two unique colors. In order to solve this problem, instead of segmenting the image we could compute a color histogram. Such a color histogram is shown in Figure 6.8. Figure 6.8 (a) shows the input image. The plot in (b) shows the color histogram. To visualize the histogram better, each color channel was quantized into 10 different intensities. Since we have three color channels, the histogram has 1000 buckets. For each bucket a cube is drawn. The size of the cube is proportional to



(a) (b)

Figure 6.6 Segmented input image. For the image shown in (a), each region was assigned the average color of the pixels that belong to that region. For the image in (b), random colors were used for each region.

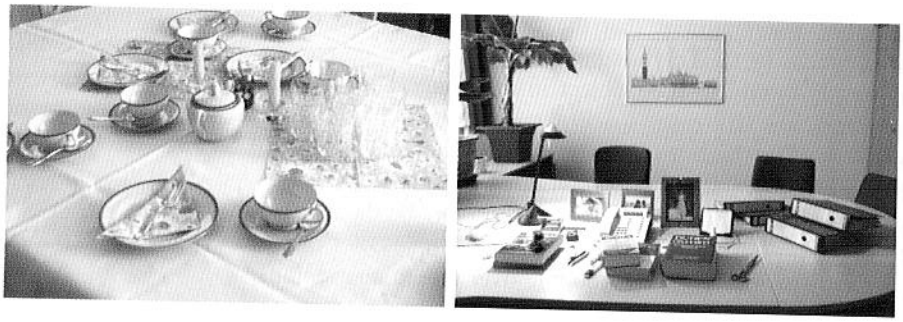
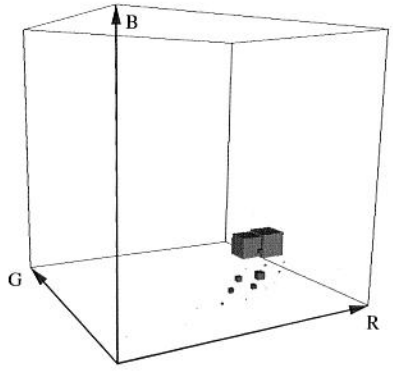


Figure 6.7 Output images produced by the algorithm of Gershon et al. (1987).

(a)



(b)

Figure 6.8 The input image is shown in (a). The graph in (b) shows the color histogram. A quantization of 10 was used for each color channel. Therefore, there are 1000 buckets in the histogram. Each bucket is represented by a cube where the size of the cube is proportional to the number of pixels of the same color in the original image.

the number of pixels having the corresponding color. The peak of the histogram is located at color [0.85, 0.45, 0.15], which is caused by the yellowish table cloth.

Let n_b be the total number of buckets. Let n_{nz} be the number of nonzero buckets and let $\mathbf{c}(j)$ be the color represented by bucket j of the histogram. The average color can then be calculated by summing up the colors of the buckets that are nonzero, i.e. averaging over all unique colors in the image.

$$a_i = \frac{1}{n_{\text{nz}}} \sum_{j=1}^{n_b} c_i(j) \tag{6.22}$$

This space average color can then be used to estimate the illuminant, which in turn can be used to calculate the reflectances. Figure 6.9 shows the result of this algorithm.

Irrespective of the exact method used to estimate the illuminant, i.e. whether we estimate the illuminant from all pixels, segment the image, or compute the histogram, the gray world

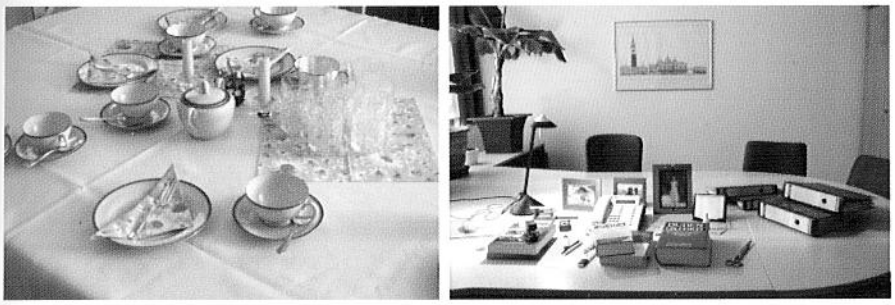


Figure 6.9 Output images produced by estimating the illuminant from the histogram of the input image.

assumption relies on the assumption that there is only a single illuminant present. However, in practice this assumption does not hold.

6.3 Variant of Horn's Algorithm

Horn (1974, 1986) has developed an algorithm for color constancy under varying illumination. We discuss this algorithm in detail in Section 7.2. Horn's algorithm assumes that the illuminant is nonuniform across the image. He suggests that the logarithm of the input signal be first taken. Next, the Laplacian is applied. Then, a threshold operation is used to separate a change in reflectance from a change of the illuminant. Changes due to a change of the illuminant are removed. Finally, the output of the thresholded data is reintegrated to obtain the log reflectances. Let us now consider what happens if the illumination is constant over the entire image. If the illuminant is constant, then we can omit taking the Laplacian, thresholding, and reintegrating.

It is assumed that the image color $c_i(x, y)$ is basically the product of the reflectance $R_i(x, y)$ and the illuminant $L_i(x, y)$ at the corresponding object location for color channel i.

$$c_i(x, y) = R_i(x, y)L_i(x, y) \tag{6.23}$$

For a constant illumination $L_i(x, y) = L_i$, we have

$$c_i(x, y) = R_i(x, y)L_i. \tag{6.24}$$

If we now apply the logarithm, the product of reflectance and illuminant turns into a sum of logarithms.

$$\log(c_i(x, y)) = \log(R_i(x, y)) + \log L_i \tag{6.25}$$

The unknown constant $\log L_i$ can be removed by transforming the logarithm of the pixel colors independently for each color channel to the range [0, 1]. This constant does not depend on the coordinates (x, y). By transforming each channel to the range [0, 1], this constant will be subtracted from each channel and the result will be an image independent of the illuminant. After transforming the log-pixel values to the range [0, 1], each channel is a color constant descriptor, given by the log reflectances. In order to undo the logarithm, we can now exponentiate and then transform the result a second time to the range